Carlos Eduardo da Costa Rodrigues

Caracterização e Modelação Semântica de Serviços em Redes Multiserviço

Outubro de 2010

Universidade do Minho
Escola de Engenharia

Carlos Eduardo da Costa Rodrigues **Caracterização e Modelação Semântica de Serviços em Redes Multiserviço**

UMinho | 2010

Universidade do Minho
Escola de Engenharia

Carlos Eduardo da Costa Rodrigues

Caracterização e Modelação Semântica de Serviços em Redes Multiserviço

Dissertação de Mestrado
Mestrado em Informática

Trabalho efectuado sob a orientação do
Professor Doutor Paulo Carvalho
e do
Professor Doutor Luis M. Álvarez-Sabucedo

Outubro de 2010

Agradecimentos

Agradeço ao Prof. Dr. Paulo Carvalho, Prof. Dr. Luis Sabucedo, Prof. Dr. Solange Lima, família e amigos.

Caracterização e Modelação Semântica de Serviços em Redes Multiserviço

Resumo

A Internet foi projectada com base no modelo de serviço melhor-esforço ("Best Effort"). Contudo, com o aumento de popularidade de aplicações tempo-real como a telefonia IP ou a Vídeo-Conferência cresce a necessidade de oferta de serviços diferenciados com garantias de Qualidade de Serviço (QoS) por parte dos ISPs. Essas garantias, que variam com o tipo de serviço, são providenciadas através da articulação de políticas e mecanismos de controlo de tráfego e de escalonamento, de acordo com os níveis de qualidade e desempenho pré-definidos. Os níveis de serviço são previamente negociados através do estabelecimento de contactos de serviço (SLAs).

Neste contexto, a caracterização e modelação semântica dos serviços de rede diferenciados assume um papel fundamental para potenciar a automatização da gestão desses mesmos serviços. Além disso, a descrição formal e semântica dos serviços potencia também a interoperabilidade e negociação entre o fornecedor de serviços e os seus clientes.

Esta dissertação centra-se, portanto, na definição de uma ontologia para serviços de rede, por forma a sistematizar e facilitar os processos de decisão sobre a implementação de QoS na infra-estrutura de rede, por forma a satisfazer os níveis de serviço pré-acordados nos SLAs. A ontologia é desenvolvida nas linguagens OWL e SWRL relacionadas com a emergente Web Semântica. A utilização aplicacional do modelo é possibilitada através da QoSModel API, conferindo-lhe aplicabilidade num contexto real.

Characterization and Semantic Modeling of Services in MultiService Networks

Abstract

The Internet is based on the best-effort service model. With the increasing of real-time applications popularity such as IP telephone and Video Conferencing, there is a growing need for ISPs providing differentiated services and QoS guaranties implementation. These guaranties, depending on the type of service, are provided through the articulation of traffic control and scheduling policies and mechanisms, according to the service level and quality established. Services levels are negotiated in a previous stage through service level agreements (SLAs).

In this context, the semantic characterization and modeling of differentiated network services undertakes an essential role in increasing network service management automation. Besides that, the semantic and formal description of services allows enhancement of negotiation and interoperability between clients and service providers.

This dissertation concentrates in the definition of a network services ontology that eases and systemizes decision support processes of QoS implementations in networks infrastructures, that satisfies services levels established in SLAs.The ontology is developed in the OWL and SWRL languages related to the emerging Semantic Web. The ontology applicational use is enabled through the API QoSModel, giving it applicability in a real world context.

Conteúdo

Lista de Figuras vi

Lista de Tabelas vii

1 Introdução 1
- 1.1 Enquadramento e Motivação . . . 1
- 1.2 Objectivos . . . 2
- 1.3 Organização da dissertação . . . 2

2 Modelos e Conceitos QoS 4
- 2.1 Modelos QoS . . . 4
  - 2.1.1 Serviços Integrados . . . 4
  - 2.1.2 Serviços Diferenciados . . . 7
- 2.2 Conceitos QoS . . . 10
  - 2.2.1 SLS . . . 10
  - 2.2.2 DSCP . . . 11
  - 2.2.3 Flow Label . . . 12
  - 2.2.4 Métricas QoS . . . 12
  - 2.2.5 Classificação de tráfego . . . 14
  - 2.2.6 Condicionamento de Tráfego . . . 14
- 2.3 Sumário do Capítulo . . . 16

3 Conceito de Ontologia e Trabalho Relacionado 17
- 3.1 Conceito de Ontologia . . . 17
- 3.2 Trabalho Relacionado - Ontologias QoS . . . 19
- 3.3 Sumário do Capítulo . . . 22

4 Ontologia para Redes Multiserviço **23**
4.1 Tecnologias Utilizadas e Planeamento . . . 23
4.1.1 OWL . . . 23
4.1.2 SWRL . . . 25
4.1.3 Protégé . . . 25
4.1.4 METHONTOLOGY . . . 26
4.2 Descrição da ontologia . . . 31
4.2.1 Módulo de Rede . . . 31
4.2.2 Módulo de Gestão de Serviços . . . 54
4.2.3 Descrição de regras . . . 70
4.3 Sumário do Capítulo . . . 76

5 QoSModel API **77**
5.1 Tecnologias utilizadas no desenvolvimento da API . . . 77
5.1.1 Jena . . . 78
5.1.2 Persistência de informação . . . 79
5.1.3 JenaBeans . . . 79
5.1.4 SPARQL . . . 79
5.1.5 RDFa . . . 80
5.2 QoSModel API . . . 80
5.2.1 Estrutura da API . . . 81
5.3 Sumário de Capítulo . . . 83

6 Conclusão **84**
6.1 Principais Contribuições . . . 84
6.2 Trabalho Futuro . . . 85

Bibliografia **86**

A Informação de apoio ao planeamento **90**
A.1 Fase 1 - Glossário de termos . . . 90
A.2 Fase2 - Taxonomia de conceitos . . . 95
A.3 Fase 3 - Relações binárias ad hoc . . . 97
A.4 Fase 4 - Dicionário de conceitos . . . 99
A.5 Fase 5 - Descrição de relações ad-hoc . . . 101
A.6 Fase 6 - Descrição de atributos . . . 103
A.7 Fase 7 - Descrição de regras . . . 104

B Exemplo RDFa **112**

Lista de Figuras

2.1 Componentes IntServ 5

4.1 Aplicação Protégé 26
4.2 Modelo METHONTOLOGY 27
4.3 Diagrama Rede 32
4.4 Diagrama Gestão 55

5.1 Diagrama da QoSModelAPI 77
5.2 Diagrama da Jena API 78

A.1 Taxonomia Gestão 95
A.2 Taxonomia Rede 96
A.3 Relações Gestão 97
A.4 Relações Rede 98

Lista de Tabelas

4.1 Conceito Network 33
4.2 Conceito Node 34
4.3 Conceito Interface 35
4.4 Conceito Policy 37
4.5 Conceito BA 38
4.6 Conceito MF 40
4.7 Conceito LogicOperator 41
4.8 Conceito DSCP 42
4.9 Conceito FlowLabel 42
4.10 Conceito Marker 43
4.11 Conceito TokenBucketPolicer 45
4.12 Conceito SingleRateThreeColorMarker 46
4.13 Conceito TwoRateThreeColorMarker 48
4.14 Conceito LeakyBucket 50
4.15 Conceito TokenBucketShaper 51
4.16 Conceito TRANSMIT 52
4.17 Conceito MARK 52
4.18 Conceito DROP 53
4.19 Conceito Client 56
4.20 Conceito SLA 57
4.21 Conceito SLS 58
4.22 Conceito StandardServiceSchedule 60
4.23 Conceito ReservedServiceSchedule 61
4.24 Conceito Metric 62
4.25 Conceito QualitativeMetricValue 64
4.26 Conceito MonitorSLS 65
4.27 Conceito MonitorScope 66
4.28 Conceito Event 67
4.29 Conceito ServicePack 68

4.30 Conceito Service 69

A.1 Fase 1 - Gestão de Serviços 90
A.2 Fase 1 - Rede Multiserviço 92
A.3 Fase 4 - Gestão e Contractos de Serviço 99
A.4 Fase 4 - Rede Multiserviço 100
A.5 Fase 5 - Gestão de Serviços 101
A.6 Fase 5 - Rede Multiserviço 102
A.7 Fase 6 - Rede Multiserviço 103
A.8 Fase 6 - Gestão de Serviços 103
A.9 Regra 1 104
A.10 Regra 2.a 105
A.11 Regra 2.b 105
A.12 Regra 2.c 107
A.13 Regra 2.d 107
A.14 Regra 3 108
A.15 Regra 4.a 108
A.16 Regra 4.b 109
A.17 Regra 4.c 110
A.18 Regra 4.d 110
A.19 Regra 5 111

Capítulo 1

Introdução

1.1 Enquadramento e Motivação

Redes de telecomunicação informática, como a Internet, foram essencialmente desenhadas para a transmissão de dados baseada no modelo de Melhor Esforço, ou seja, por defeito não existem garantias de entrega final da informação, de níveis de desempenho da rede na transmissão de dados, reserva de recursos ou segurança.

Hoje em dia, cada vez mais aplicações requerem algum nível de garantia de qualidade por parte da rede. Aplicações de tempo real, como Voz sobre IP (**VoIP**), Videoconferência, jogos em rede, são bastante sensíveis a latências, variações de atraso e perda de pacotes e requerem a reserva de largura de banda na rede. Estes factores influenciam a Qualidade de Experiência (**QoE**) final do utilizador, ou seja, a percepção final da qualidade de serviço fornecida. Transferências comerciais de informação de grande importância têm exigências especiais, particularmente acerca de largura de banda reservada, perda de pacotes, segurança em alguns casos da capacidade de resposta do serviço. Com aumento de tráfego a altas taxas de transmissão de informação, com crescente número de clientes e cada vez mais elevadas exigências quanto ao fornecimento de serviços obrigam as companhias fornecedores de serviços de rede (**ISPs**) a implementar modelos para aumentar as garantias de qualidade de serviço (**QoS**) sem serem baseados apenas no sobredimensionamento da rede.

As políticas de QoS, implementadas através de mecanismos de controlo e escalonamento de tráfego, permitem o tratamento e controlo diferenciado de serviços nas infra-estruturas da rede. Essa diferenciação pode ser executada através de várias estratégias com diferentes níveis de granularidade quanto à acumulação de fluxos de tráfego, como os modelos IntServ e DiffServ. A gestão de QoS é su-

portada através da definição de contractos de nível de serviço (**SLA**). Nos SLAs várias questões administrativas são definidas, deixando as questões tecnológicas na definição de especificações de nível de serviço (**SLS**). Os SLSs fornecem orientação à implementação de mecanismos de QoS na rede e à conformidade contratual de desempenho da rede do serviço contratado. A gestão e implementação de políticas QoS são tarefas exigentes para serem executadas manualmente, podendo levar à implementação deficiente dos serviços e a uma pobre gestão de recursos. Devido ao grande número de clientes e à heterogeneidade das redes hoje em dia, é importante a investigação de novos processos para automatização da implementação e gestão de serviços em redes multiserviço.

1.2 Objectivos

Este trabalho propõe uma modelação semântica de serviços em redes multiserviço. A modelação semântica assenta em estruturas de dados baseadas em conceitos, o que permite a definição de relações lógicas mais poderosas entre componentes de informação. Este modelo tem em vista fornecer suporte a uma plataforma que facilite a interoperabilidade entre cliente e fornecedor de serviço, gestão de contractos e inquirição de dados por parte do fornecedor e em certos aspectos por parte do cliente. Também visa facilitar o mapeamento entre SLSs e rede através do estabelecimento de relação entre os vários elementos comuns às duas perspectivas. O modelo semântico de redes multiserviço foi desenvolvido em OWL DL com a ajuda da ferramenta Protégé.

Os objectivos propostos para esta dissertação são:

- Identificação dos conceitos envolvidos na caracterização de serviços de rede.
- Construção de um modelo semântico sobre o domínio de redes multiserviço.
- Desenvolvimento de uma API, que forneça um conjunto de ferramentas, possibilitando a construção e gestão de uma base de conhecimento, suportada pelo modelo semântico desenvolvido.

1.3 Organização da dissertação

No capítulo 2, descrevem-se os dois modelos mais importantes utilizados na implementação de serviços de rede, assim como são identificados as entidades na área de QoS mais relevantes para a construção do modelo semântico.

No capítulo 3 é apresentado o conceito de ontologia como modelo de representação de conhecimento. Também é realizada uma breve análise a algumas propostas de ontologias sobre serviços de rede e QoS desenvolvidas pela comunidade científica ao longo dos últimos anos.

O capítulo 4 apresenta uma descrição pormenorizada do modelo semântico desenvolvido. São especificadas todas as tecnologias e a metodologia de planeamento aplicadas na construção da ontologia. É exposta a sua estrutura modular e apresentada uma descrição classe a classe contidas em cada um dos seus módulos.

No capítulo 5, é descrita a API desenvolvida com o objectivo de permitir a utilização do modelo semântico na construção de bases de conhecimento, para a integração em soluções aplicacionais sobre gestão de redes multiserviço.

No capitulo 6, são apresentadas as principais conclusões e tópicos de trabalho futuro.

Capítulo 2

Modelos e Conceitos QoS

2.1 Modelos QoS

2.1.1 Serviços Integrados

Com a experiência resultante do suporte a aplicações em rede com necessidades tempo-real, nomeadamente da rede MBONE, verificou-se a incapacidade da arquitectura da Internet em providenciar suporte a aplicações tempo-real (Wang, 2001). A primeira tentativa de desenvolvimento duma arquitectura QoS resultou na estratégia de Serviços Integrados (**IntServ**). Um dos principais desafios encontrados no desenvolvimento do modelo de Serviços Integrados foi a construção de uma arquitectura que operasse sobre o modelo orientado ao datagrama característico da Internet, mas mantivesse alguma independência desse modelo com a inclusão de novos conceitos. Com os Serviços Integrados, foram introduzidos conceitos e mecanismos como modelo de serviços, protocolos para reserva de recursos, controlo de admissão, novos mecanismos de condicionamento e escalonamento de tráfego (Wang, 2001).

Os Serviços Integrados adoptam uma estratégia de reserva de recursos por fluxo de dados. Cada fluxo é continuamente identificado em cada nó da rede, obedecendo a uma especificação de fluxo (normalmente um vector composto por endereço IP de origem, endereço IP destino, porta de origem, porta destina e identificador de protocolo).

A aplicação começa por enviar um pedido de reserva de recursos, que se for aceite é posteriormente construído um caminho de recursos reservados na rede, sendo estabelecidos limites máximos possíveis de performance na transmissão do

fluxo de forma a compactuar com o serviço requerido. A conformidade com os limites previamente estabelecidos é garantida com implementação na rede de mecanismos de condicionamento, escalonamento de tráfego e estratégias de roteamento. É igualmente necessária a existência de uma componente para a admissão de pedidos de reserva de recursos e gestão dessa mesma reserva (Wang, 2001) (Marchese, 2007). Em seguida, efectua-se uma breve descrição das componentes do modelo.

2.1.1.1 Componentes do modelo

Nos Serviços Integrados existem dois planos de operação: o plano de controlo e plano de dados. Na Figura 2.1 podemos observar os vários componentes que compõem os dois planos (Wang, 2001).

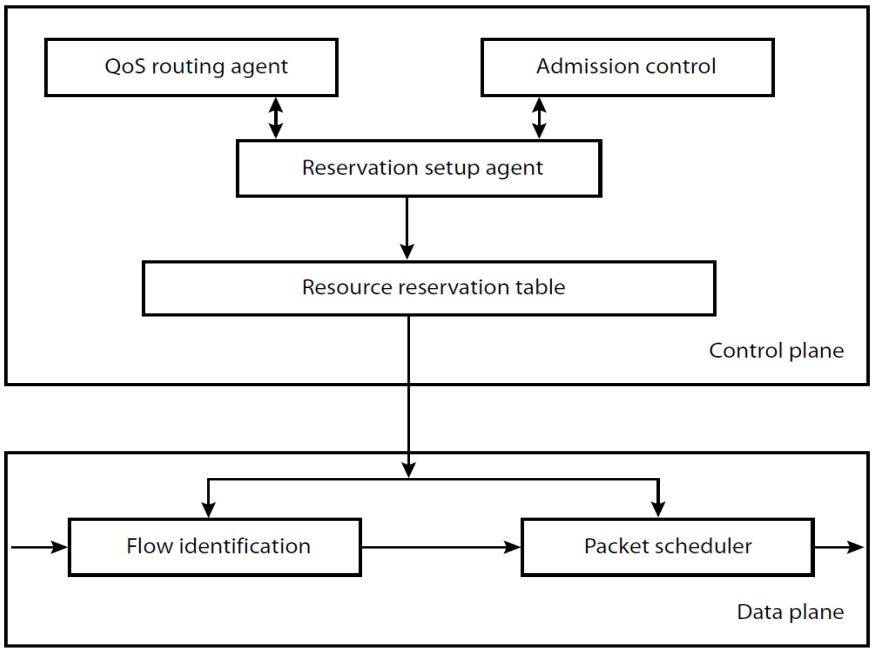

Figura 2.1: Modelo de Serviços Integrados (Wang, 2001).

Controlo de Admissão

O controlo de admissão tem duas funções principais interligadas entre si. A primeira função consiste na análise e indagação de pedidos de reserva de recursos. Para a reserva de recursos, o controlo de admissão é o primeiro componente encontrado e o que dá o veredicto sobre se essa reserva é aceite ou não. Para essa decisão, é necessário ter a noção de que recursos estão reservados, como e a quem

estão reservados. A segunda função centra-se sobre a monitorização dos recursos usados e disponíveis.

Reserva de Recursos

A reserva de recursos passa pela reserva individual nó-a-nó, estabelecendo assim caminho para o qual foram reservados recursos a serem utilizados para o encaminhamento de pacotes englobados no serviço requerido. Para esse efeito são utilizados protocolos de reserva de recursos, como o RSVP, que instalam em cada nó um estado de reserva. Estes protocolos carregam informação para suporte sobre a possibilidade de reserva de recursos nesse nó. Outra característica nos protocolos de reserva de recursos (semelhante aos protocolos de encaminhamento) é a de reacção a mudanças na topologia.

Agente de Encaminhamento

Para a reserva de recursos tem que se determinar qual a rota para a utilização óptima desses recursos. Os protocolos de encaminhamento IP não são ideais para determinação de rotas optimizadas a QoS, pois informação que em se baseiam não é suficiente. A determinação de tais rotas depende de mais de que um factor e muitas vezes verifica-se que se tornam em problemas NP completos. Contudo, existem alguns algoritmos propostos no âmbito da engenharia de tráfego.

Especificação de Fluxo

Para verificar se um pacote pertence de facto ao fluxo para o qual foram reservados recursos é necessário proceder à identificação. Essa identificação é feita usando cinco parâmetros: endereço IP de origem, endereço IP destino, porta de origem, porta destino e identificador de protocolo. Neste processo é usado uma tabela de ligação entre a especificação dos fluxos reservados e a sua identificação atribuída na reserva de recursos (por exemplo: identificação RSVP do fluxo). Este processo aplica-se a todos os pacotes contribuindo para os problemas de computação e escalabilidade inerentes aos Serviços Integrados.

Escalonamento de Pacotes

O processo de escalonamento determina a ordem pela qual os pacotes vão ser encaminhados, impactando directamente no atraso de transmissão e, em casos de congestionamento da rede, decide sobre o descarte de pacotes.

2.1.1.2 Desvantagens do modelo

O principal ponto de falha dos Serviços Integrados é a sua escalabilidade. Isto é principalmente devido à aplicação nó-a-nó do processo de especificação de fluxo em todos os pacotes, o que leva algum atraso indesejável principalmente se os pacotes são transmitido em rajada. Os processos de reserva de recursos e escalonamento de pacotes são aplicados por fluxo, não prevendo uma filosofia de agregação dos mesmos com características ou requisitos semelhantes. Todos os pontos anteriores levam a que a complexidade computacional aumenta muito de acordo com o número de fluxos existente. Ainda mais, o uso de protocolos para reserva de recursos e a sua manutenção de estado implicam maior sobrecarga na rede e consomem muitos recursos (Marchese, 2007).

2.1.2 Serviços Diferenciados

Os Serviços Diferenciados(**DiffServ**) surgiram com a necessidade de um modelo de QoS de maior simplicidade, com o fim de suprimir parte dos problemas de escalabilidade presentes nos Serviços Integrados. Neste modelo, o tráfego é dividido em um número pequeno de classes e a alocação de recursos é feita de forma agregada em vez de por fluxo como é aplicado nos Serviços Integrados. Não existe a necessidade de um mecanismo de reserva de recursos e a informação sobre a identificação da classe é transportada no próprio cabeçalho do pacote.

No entanto existe uma diferenciação de funcionalidades entre os nós do interior da rede e os nós de fronteira. Os nós de fronteira são responsáveis pela classificação e condicionamento do tráfego. Isto inclui identificação do fluxo, aprovar a conformidade deste com os requisitos predefinidos ou negociados, resultando na atribuição de uma classe ao pacote ou o seu descarte. No interior da rede apenas é necessária a distinção e tratamento do tráfego dentro de um limitado número de classes. É aplicado o encaminhamento requerido pela classe através de técnicas de escalonamento de pacotes e gestão activa de filas de encaminhamento (Wang, 2001) (Marchese, 2007).

Os serviços são implementados na rede com a conjunção destes componentes: classificação e condicionamento de tráfego à entrada do domínio, mapeamento do tráfego a um PHB específico e encaminhamento de tráfego no interior do domínio baseado no PHB (Wang, 2001).

2.1.2.1 Comportamento por nó (PHB)

Elemento essencial na definição de serviços de rede. Nos Serviços Diferenciados cada classe especifica qual o tratamento de encaminhamento aplicado aos pacotes.

O tratamento especificado pela classe é designado de *Per-hop Behaviour* (**PHB**) , sendo definido localmente, ou seja, um valor DSCP num domínio pode corresponder a um PHB diferente do que corresponde noutro domínio. Um conjunto de PHBs pode formar um grupo PHB. Para formar um grupo PHB, tem de existir a partilha de pelo menos uma restrição comum dentro do grupo. Dentro do grupo, cada PHB define diferentes níveis de recursos reservados tais como largura de banda reservada ou precedência de descarte. Na rede, o PHB é obtido através de mecanismos de escalonamento e *buffering* de tráfego (Wang, 2001) (Babiarz et al., 2006).

AF PHB

O PHB atribuído a classes *Assured Forwarding* (**AF**) é dividido em 4 níveis diferentes de recursos reservados: AF1, AF2, AF3 e AF4. Dentro de cada nível, podem-se especificar até três níveis de precedência de descarte. A precedência de descarte indica a prioridade de descarte atribuída ao pacote em caso de necessidade devido a congestão na rede. Esta prioridade no descarte de pacotes aplica-se apenas se for aplicado um mecanismo de gestão activa de filas. As precedências de descarte entre diferentes classes AF reservadas são independentes entre si.

As classes AF são atribuídas a serviços com mais de um nível de recursos reservados. Todo o tráfego dentro do perfil esperado é atribuído a uma classe AF com maior prioridade e menor precedência de descarte dentro do perfil do serviço. Para tráfego excedente pode-se optar por atribuir um nível menor de prioridade de encaminhamento ou uma precedência maior de descarte, penalizando-o em termos de recursos reservados (Wang, 2001) (Babiarz et al., 2006).

Este PHB é dedicado a serviços "elásticos", cujo receptor tenha a capacidade de detectar perdas e variações do atraso, e o emissor seja capaz de adaptar a sua taxa de transmissão conforme a capacidade disponível da rede (Babiarz et al., 2006).

EF PHB

O EF PHB equivale a um comportamento similar a um circuito virtual. Normalmente pacotes marcados como pertencentes ao PHB EF são encaminhados sempre que possível, com prioridade sobre qualquer outro tipo de tráfego. Normalmente, é usado uma fila prioritária com técnica de escalonamento de pacotes garantido assim preferência de encaminhamento sobre qualquer outra fila implementada. É de boa prática reservar apenas uma pequena parte da largura e não existe a atribuição de precedência de descarte (Wang, 2001) (Babiarz et al., 2006).

O EF PHB é dedicado a serviços "inelásticos", com características de tempo-real e, na maior parte casos, com taxa de transmissão constante. Aplicações de

voz, como o VoIP, enquadram-se neste tipo de serviços (Babiarz et al., 2006).

DF PHB

O tratamento aplicado ao tráfego dentro do serviço Best-Effort. Não são oferecidas garantias de QoS por parte da rede (Babiarz et al., 2006).

2.1.2.2 Comportamento num domínio (PDB)

Indica o tratamento expectável que um agregado de tráfego vai receber desde a entrada até à saída de um domínio de rede. O PDB resulta do conjunto de regras de classificação e de condicionamento, e do PHB (ou um grupo de PHBs) ao qual os pacotes estão sujeitos no percurso do domínio de rede. Cada PDB é descrito por uma série de parâmetros correspondentes a métricas de desempenho da rede. Esta quantificação do comportamento *edge-to-edge*, facilita a composição de serviços fornecidos no domínio de rede, expondo apenas a informação suficiente que permita a garantia de QoS. Os PDBs podem ser relacionados directamente com os SLSs definidos nos contratos de serviço (Nichols e Carpenter, 2001).

Virtual Wire PDB

Tratamento agregado de tráfego emulando um circuito dedicado. Com base no PHB correspondente ao EF, os pacotes são entregues com altos níveis de garantias de desempenho por parte da rede, com valores baixos de latência, variação de atraso e perda de pacotes.

Assured Rate PDB

Baseado no AF PHB, com garantia de largura de banda, mas sem limites de latência e variação de atraso. De acordo com as características do AF PHB, é possível especificar ainda um valor de largura de banda em excesso para além da reservada, e vários níveis de probabilidade de descarte de tráfego. Requer o condicionamento de tráfego em três níveis de conformidade.

Low Effort PDB

Tratamento dedicado ao tráfego não crítico, sobre o qual possam ser retirados recursos em prol do correcto funcionamento da rede. Atribui-se-lhe menor prioridade do que ao tráfego dentro do serviço *Best-Effort*. Portanto, não deve corresponder a nenhum serviço de rede, mas sim a uma estratégia de gestão interna com o fim

de prevenir a competição por recursos de rede entre o tipo de tráfego abrangido por este PDB e outros serviços, nomeadamente o serviço *Best-Effort* (Bless et al., 2003).

2.2 Conceitos QoS

Nesta secção, explicitam-se alguns dos conceitos presentes no modelo semântico desenvolvido ao longo desta dissertação.

2.2.1 SLS

As especificações do serviço pretendido pelo cliente são inicialmente acordadas através da realização de SLAs. Para além de informação referente a aspectos administrativos, nos SLAs também são especificados os níveis de serviços (SLS) requeridos pelo cliente, correspondendo às componentes técnicas do SLA. Geralmente, nos SLS são especificados o âmbito do serviço, regras de classificação e condicionamento de tráfego (**TCS**), garantias QoS, etc. Com a especificação de todos estes parâmetros, os SLSs são a principal referência para a implementação dos serviços contratados na rede. Os SLAs/SLS podem ser estáticos ou dinâmicos. Estes últimos permitem a sua renegociação conforme as necessidades do cliente. A renegociação dinâmica de SLSs requer um sistema de gestão de redes que automatize o processo de reconfiguração da rede. A estrutura para SLS apresentada a seguir é baseada no trabalho de desenvolvido em Goderis et al. (2002) e Salsano et al. (2000):

Scope Especifica os limites lógicos dentro dos quais o serviço é providenciado, gerido e onde as politicas especificadas no contrato são aplicadas. Normalmente, os SLSs estão associados a fluxos de dados unidireccionais entre o ponto de entrada e o de saída. Nos serviços bidireccionais são-lhe associados mais do que um SLS. Os pontos de entrada e de saída representam as interfaces de entrada (*ingress*) e de saída do domínio (*egress*) respectivamente. Nestes pontos são aplicadas políticas de identificação e condicionamento de tráfego. A cardinalidade dos pontos entrada e saída pode ser: (i) um-para-um, (ii) um-para-muitos, (iii) um-para-algum, (iv) muitos-para-um ou (v) algum-para-um. Cada ponto de entrada ou saida que compõem o *Scope* do SLS sao representados por um identificador único, como por exemplo um endereço IP..

Classificação de Tráfego Especifica como são identificados os fluxos de dados pertencentes ao serviço contratado para permitir um tratamento diferenciado

das mesmas. O fluxo é identificado por um ou mais parâmetros (como o endereço de origem ou o de destino). A classificação de tráfego pode ser aplicada a micro-fluxos ou a macro-fluxos, i.e., (micro-fluxos agregados de um determinado serviço). Os parâmetros usados na classificação de tráfego podem ser: (i) um ou um conjunto de endereços (IPv4, IPv6, MAC), (ii) um ou um conjunto de marcas atribuídas (campo DSCP do IPv4, campo Exp do MPLS) ou um conjunto de informações sobre o protocolo usado (identificador de protocolo, número de porta, campo Flow Label do IPv6, etc.).

Condicionador de Tráfego Tem como objectivo a implementação de políticas de policiamento e/ou marcação sobre o tráfego previamente classificado. Existem três diferentes componentes que podem ser implementadas: *policer*, *shaper*, marcador.

Garantias de QoS Nelas são especificadas garantias providenciadas pelo ISP sobre QoS do serviço contratado. Normalmente são definidas quatro métricas de QoS: largura de banda, atraso, *jitter* e perda de pacotes. Sempre que existam garantias de QoS especificadas também tem que ser especificado um condicionador de tráfego. Os parâmetros de atraso, *jitter* e perdas são aplicados ao tráfego dentro do perfil estipulado no SLS. Os parâmetros de atraso e *jitter* são usualmente representados pelo limite máximo permitido ou um par de números compostos pelo limite máximo e um quantil (percentil). A perda de pacotes representa o rácio de pacotes perdidos detectados na interface de saída dentro do *scope*. Os parâmetros de performance também podem ser especificados de uma forma qualitativa. É usual encontrar os níveis de serviço dividido em *gold*, *silver* ou *bronze*.

Duração do Serviço Define o período no qual é prestado o serviço associado com o SLS. É necessário a definição de uma data de inicio, e em caso de período reservado de prestação de serviço, uma data de fim de serviço.

Fiabilidade Normalmente é especificada pelo tempo médio de indisponibilidade do serviço por ano (**MDT** – *Mean Downtime per year*) e pelo tempo máximo permitido para recuperação do serviço (**TTR** – *Time To Repair*). Assim como nas garantias de QoS, o não cumprimento das garantias acordadas podem resultar em penalizações, previamente estabelecidas em contrato, para o fornecedor de serviço.

2.2.2 DSCP

O *Differentiated Services Code Point* (**DSCP**) é o campo do IPv4 usado para identificar os varios níveis de tratamento diferenciado providenciado pela rede na

garantia de QoS . O DSCP sucede ao antigo campo TOS do IPv4 usado também para identificação do tipo de serviço. O campo DSCP carrega identificação da classe de tráfego à qual todos os pacotes recebem mesmo tratamento por parte da rede, i.e., o DSCP é mapeado pelo nó DiffServ num PHB. Apesar de existirem várias recomendações, o mapeamento do DSCP num PHB é totalmente configurável. O campo DSCP é composto por 8 bits, mas apenas 6 deles são actualmente utilizados Os últimos 2 bits estão reservados para notificação fim a fim de congestão na rede (**ECN**) (Ramakrishnan et al., 2001). O campo é dividido em duas partes. A parte *Class Selector* corresponde aos primeiros 3 bits e usualmente identifica os diferentes níveis de prioridade dada ao tráfego. Os restantes 3 bits correspondem à parte de *Precedence* e são usualmente reservados par indexar os diferentes níveis de precedência de descarte de pacotes. Alguns dos DSCPs estão reservados a tráfego especial como o utilizado para a manutenção e gestão de rede ou para a sinalização (Nichols et al., 1998) (Marchese, 2007).

2.2.3 Flow Label

O campo *Flow Label* é uma das inovações trazidas pelo IPv6 no âmbito das capacidades QoS. Com o *Flow Label*, oferece-se a possibilidade de identificação de fluxos de dados com a utilização de apenas um campo como parâmetro. Esta nova característica pretende aliviar alguma da complexidade computacional e escalabilidade inerente à especificação de fluxos, um problema claramente identificado na implementação IPv4. O IPv6 manteve o campo DSCP do IPv4, designando-se de *Traffic Class*. Com estes dois campos o IPv6 contem capacidades coerentes com os paradigmas já existentes como o DiffServ e IntServ e abrindo espaço para o desenvolvimento de novos paradigmas (Marchese, 2007).

2.2.4 Métricas QoS

A seguir descreve-se as métricas indicadoras do desempenho da rede.

2.2.4.1 Largura de banda

Número de bits transferidos por unidade de tempo. Quando é especificada no SLS, geralmente indica o limite mínimo de largura de banda garantida. Esta é conseguida através da utilização de técnicas de escalonamento de filas.

2.2.4.2 Atraso

Mede o tempo que um pacote demora a chegar entre dois pontos na rede. Geralmente é medido em fracções de segundos. Existem factores que afectam o atraso final, podendo-se então dividir em vários tipos: atraso na transmissão do pacote, no processamento de pacotes por parte dos nós da rede, de propagação no canal de transmissão e atraso introduzido pelas filas de espera e pelos *buffers*. Todos estes factores resultam no atraso final percebido. Contudo, alguns destes tipos de atraso podem ser desprezáveis ou omitidos por opção (Marchese, 2007) (Almes et al., 1999a) (Almes et al., 1999c).

Segundo as normas apresentadas pelo IETF-IPPM e ITU-T, o atraso é medido desde que o primeiro bit deixa a origem até à chegada do último bit no destino. Apesar de nomes diferentes, OWD por parte do IETF-IPPM e IPTD por parte do ITU-T, as duas especificações apresentadas para métrica equivalem-se (Almes et al., 1999a) (Almes et al., 1999c).

2.2.4.3 Variação do atraso (Jitter)

A palavra *jitter* a significa variação duma métrica. Nas ciências das telecomunicações o *jitter* usado para referir a variação do atraso na transmissão de pacotes entre dois pontos na rede. Aplicações de transmissão em tempo-real e isócronas como aplicações de transmissão de voz e vídeo em directo são particularmente sensíveis a variação de atrasos na transmissão. No entanto essa sensibilidade pode ser colmatada com implementação de *buffers anti-jitter*. Mas usualmente aplicações inelásticas com necessidades de tempo real requerem valores de *jitter* muito baixos (Demichelis e Chimento, 2002).

A medição do *jitter* é também útil também para o redimensionamento de *buffers* e para percepção das dinâmicas no desempenho das filas de espera implementadas nos nós da rede. As especificações apresentadas pelas IETF-IPPM e ITU-T, diferem na forma como a variação do atraso é medida. Na especificação IETF, a variação é calculada entre valores de atraso consecutivos. Na especificação ITU-T, é considerado um valor referência para o cálculo dessa variação (Marchese, 2007) (Demichelis e Chimento, 2002).

2.2.4.4 Perda de pacotes

É especificada pelo rácio entre os pacotes perdidos e o total de pacotes transmitidos. A perda de pacotes é principalmente provocada pela congestão na rede, sendo usada como indicador por protocolos que implementam mecanismos de controlo de congestão. Afecta aplicações tempo-real com impossibilidade de retransmissão

de pacotes perdidos, significando perda de informação, embora também provoque a redução da taxa de transmissão em protocolos adaptativos com controlo de congestão (Marchese, 2007) (Almes et al., 1999b).

As normas IETF-IPPM e ITU-T apresentam especificações semelhantes. A norma IPPM contém ainda a especificação OWLP para a medição do padrão da perda de pacotes (Almes et al., 1999b).

2.2.5 Classificação de tráfego

Na classificação de tráfego são identificados os pacotes para a aplicação de tratamento diferenciado de acordo com as especificações predefinidas. A classificação rege-se segundo de regras, pode ser feita de duas formas: *Multifield* (**MF**) e *Behavior Aggregate* (**BA**).

A classificação MF consiste na definição de um ou mais parâmetros. Estes dependem das tecnologias utilizadas: o vector de cinco parâmetros (endereço origem e destino, porta origem e destino e identificador de protocolo) em IPv4, Flow Label em IPv6, VPI/VCI em ATM, Label em MPLS. Também existem implementações que utilizam a camada aplicacional da pilha protocolar TCP/IP para classificação de pacotes. A combinação de parâmetros é feita através de disjunção ou conjunção.

A classificação BA é baseada numa marca atribuída ao tráfego. Esta classificação é aplicada apenas em tráfego previamente marcado. São utilizados os campos reservados para marcação agregada de tráfego: DSCP no IPv4, Traffic Class no IPv6, Exp no MPLS (Marchese, 2007) (Goderis et al., 2002).

Normalmente no modelo DiffServ, a classificação MF é feita no nó de entrada do domínio, a não ser que a marcação de tráfego já tenha sido previamente efectuada num outro domínio ou no cliente. A classificação BA é feita nos nós posteriores à marcação de tráfego. A Especificação de Fluxo do IntServ pode ser vista como um tipo de classificação especial. Gere-se pela regra mais rígida e menos expansiva com a utilização do vector dos cinco parâmetros. É aplicada em todos os nós DiffServ da rede (Wang, 2001).

2.2.6 Condicionamento de Tráfego

A função do condicionador passa por medir tráfego, comparando-o a um perfil de tráfego predefinido e tomar uma decisão sobre os vários níveis de conformidade de tráfego. O condicionamento de tráfego é importante para garantir que os pacotes entrem no domínio em conformidade com os níveis de perfil de serviço pré-estabelecidos. Também é uma medida importante na distinção dos vários níveis de conformidade para que possa ser aplicado um tratamento diferenciado sobre o

tráfego (Wang, 2001). Os vários tipos de condicionamento podem ser distinguidos com base no tratamento dado ao tráfego em não conformidade por entidades como o: *Policer* e *Shaper*.

2.2.6.1 Policer

Com o *policer* são aplicadas acções imediatas sobre os pacotes, de acordo com a sua conformidade em relação a parâmetros de descrição de tráfego previamente especificados. A designação e número de parâmetros de descrição do tráfego dependem do algoritmo escolhido para o policiamento. Dependendo também do algoritmo de policiamento, podem existir vários níveis de conformidade do tráfego nos quais podem ser tomadas acções, que passam pela transmissão, remarcação ou exclusão de pacotes. Por exemplo, o *policer* pode ser binário, dividindo os níveis de conformidade em tráfego *"in-profile"* e tráfego *"out-of-profile"*.

Token Bucket

Algoritmo aplicado no policiamento de tráfego. Este algoritmo é definido por dois parâmetros. A taxa à qual os créditos (*tokens*) são acumulados, correspondendo à taxa de transmissão de tráfego dentro do perfil e a capacidade do "balde", onde são acumulado os créditos para providenciar alguma tolerância ao tráfego transmitido em rajadas. Conforme o número de créditos presentes no "balde", os pacotes são considerados dentro ou fora do perfil de tráfego. A estes dois níveis de conformidade é então aplicada uma acção imediata.

Single Rate Three Color Marker (srTCM)

O trTCM consiste num *policer* que funciona com base em três parâmetros: *Committed Information Rate (CIR)*, *Committed Burst Size (CBS)*, *Excess Burst Size (EBS)*. Marca o tráfego em três cores (verde, amarelo, vermelho), ou seja, em três níveis de conformidade. O srTCM é composto por dois *Token Buckets* C e E, com capacidades de acordo com os parâmetros CBS e EBS respectivamente. Os dois *Token Buckets* são preenchidos à mesma taxa, o parâmetro CIR. Se o tamanho do pacote é menor ou igual ao número de *tokens* presentes no *Token Bucket* C, o pacote é assinalado como "verde". Caso contrário, se o tamanho do pacote está em conformidade com o número de *tokens* presentes no *Token Bucket* E, o pacote é marcado como "amarelo". Se não se verificar nenhuma das regras anteriores, o pacote é assinalado como "vermelho" (Heinanen e Guerin, 1999a).

Two Rate Three color Marker (trTCM)

O trTCM funciona com base em quatro parâmetros: *Committed Information Rate (CIR)*, *Committed Burst Size (CBS)*, *Peak Information Rate (PIR)* e *Peak Burst Size (PBS)*. Marca o tráfego em três níveis de conformidade diferentes (verde, amarelo, vermelho). O trTCM compõem-se por dois *Token Buckets* P e C, com capacidades PBS e CBS, e diferentes taxas de preenchimento, PIR e CIR respectivamente. Se o tamanho do pacote é maior que numero de *tokens* presentes no *Token Bucket* P, o pacote é assinalado como "vermelho". Caso contrário, se o tamanho do pacote é maior que o número de *tokens* presentes no *Token Bucket* C, o pacote é assinalado como "amarelo". Se não se verificar nenhuma das regras anteriores, o pacote é assinalado como "verde" (Heinanen e Guerin, 1999b).

2.2.6.2 Shaper

No *shaper* são usados também o algoritmo e parâmetros de conformidade do tráfego, mas não é aplicada nenhuma acção imediata. Os pacotes fora dos parâmetros de conformidade mantido numa fila de espera até se voltar atingir os índices de conformidades pretendidos. Os pacotes em excesso são mantidos na fila de espera até a um certo limite.

Leaky Bucket

Algoritmo usualmente utilizado no *shaping* de tráfego. Neste algoritmo é apenas definido um parâmetro, correspondendo à taxa de transmissão de tráfego. Desta forma, a transmissão em rajada de tráfego é eliminada, sendo que os pacotes são transmitidos para o canal até à taxa predefinida. Para incluir alguma tolerância a rajadas poder-se-á implementa uma versão *shaper* do algoritmo Token Bucket.

2.3 Sumário do Capítulo

Para a construção de uma ontologia descritiva de serviços em redes multiserviço, é necessária uma identificação inicial de paradigmas e conceitos relacionados com o processo de implementação de serviços na rede. Neste capítulo, começou-se por descrever os dois principais modelos de QoS existentes: IntServ e DiffServ. Também foram efectuadas descrições acerca dos conceitos envolvidos na realização de contractos de serviços e no processo de implementação de serviços na rede. No próximo capítulo apresenta-se o conceito de ontologia e várias propostas de modelos semânticos na área de QoS e redes multiserviço.

Capítulo 3

Conceito de Ontologia e Trabalho Relacionado

3.1 Conceito de Ontologia

Na filosofia, o termo ontologia (em grego *ontos* e *logoi*, significa conhecimento do ser) refere-se ao estudo formal e material das essências, assim como da diferença entre o ser e o existir, e de como as entidades possam ser classificadas de acordo com propriedades comuns.

Nas ciências computacionais, existem variadas definições. Segundo Neches et al. (1991), "uma ontologia define os termos básicos e relações compreendendo o vocabulário de uma área como também as regras para a coligação de termos e relações para a definição de extensões ao vocabulário". De uma forma ainda mais divulgada, Gruber (1993) define ontologia como "um especificação explícita de uma conceptualização". Por outro lado, Goyal et al. (1996) destaca a possibilidade de reutilização da mesma ontologia, servindo como estrutura para a construção de variadas bases de conhecimento, "uma ontologia é um conjunto de termos estruturado hierarquicamente para a descrição de um domínio que pode servir de esqueleto a uma base de conhecimento".

Uma ontologia é uma representação de um domínio de conhecimento através da definição de conceitos (classes) constituidos por propriedades (relações e atributos) e restrições. Com um conjunto de instanciações das suas classes forma uma base de conhecimento, embora na realidade, exista uma ténue fronteira entre a base de conhecimento e a ontologia (Noy e Mcguinness, 2001).

Com o desenvolvimento de ontologias é possível a realização de um modelo comum, que possa ser interpretado tanto por humanos como por máquinas e sobre o qual se possam tirar conclusões. A possibilidade de se poder retirar inferências sobre um modelo descritivo de conhecimento, facilita a automatização de processos, sendo que as ontologias são um dos métodos utilizados para representação de conhecimento no ramo da inteligência artificial. De acordo com Noy e Mcguinness (2001), as razões para o recurso à utilização de ontologias especificam-se na partilha de conhecimento comum sobre uma estrutura de informação entre humanos e agentes de software, para permitir a reutilização dessa informação, explicitar suposições sobre o domínio de conhecimento, para separar a informação sobre o domínio de conhecimento da informação operacional e para a analise da informação.

Usualmente as ontologias descrevem conceitos como um conjunto de propriedades orientadas por restrições. Os conceitos são instanciados como indivíduos, que compõem a base de conhecimento. Dependendo do paradigma da ontologia, os componentes básicos variam. As ontologias baseadas em *frames* e lógica de primeira ordem usam classes como representação de conceitos (podem ser abstractas ou específicas), verificando-se a existência de relações entre classes, atributos, funções (tipo especial de relações), axiomas formais e instâncias de classes. No paradigma da lógica de descrição mantem-se a designação de conceito (pode ser primitivos ou definido) e indivíduo (como instância de conceito) e as relações são designadas de *roles* (Gómez-Pérez et al., 2007).

Existem vários tipos de ontologias (Gómez-Pérez et al., 2007):

- Ontologias de Representação de Conhecimento (Ontologias KR) – definem as primitivas de representação (ex: classes, relações, atributos, axiomas, etc.) para a formalização de conhecimento num paradigma KR.
- Ontologias Gerais ou Comuns – representam senso comum reutilizável entre vários domínios de conhecimento.
- Ontologias de Alto Nível – descrevem e providenciam conceitos gerais que constituem as raízes taxonómicas, sobre os quais se devem interligar as ontologias.
- Ontologias de Domínio de Conhecimento – reutilizáveis dentro de um domínio de conhecimento.
- Ontologias de Métodos – definem conceitos e relações aplicadas na especificação de um processo de inferência. de forma a realizar uma determinada tarefa.
- Ontologias de Tarefas – descrevem o vocabulário relacionado com a realização de uma tarefa ou actividade específica.

- Ontologias Aplicacionais – descrevem um domínio de conhecimento, especializando o vocabulário para um determinado tipo de aplicação.

As ontologias podem ser exprimidas em várias linguagens. Algumas destas baseadas no paradigma da lógica de primeira ordem (ex: KIF), em *frames* combinada com lógica de primeira ordem (ex: CycL, Ontolingua, OCML, FLogic) ou no paradigma da lógica de descrição (ex: LOOM) (Gómez-Pérez et al., 2007). Um passo importante na evolução de linguagens para o desenvolvimento de ontologias foi a conceptualização da Web Semântica. Especificada pelo *World Wide Web Consortium* (**W3C**), a Web Semântica tenta combinar as funcionalidades da Internet com as virtudes da representação semântica. Com a Web Semântica, novas linguagens para o desenvolvimento de ontologias surgiram conhecidas como linguagens *markup* para a definição ontologias. Essas linguagens têm como base o *Extensible Markup Language* (**XML**) e o *Resource Description Framework* (**RDF**). Uma das mais importantes linguagens da Web Semântica, a linguagem DAML+OIL, que conjuga a *DARPA Agent Markup Language* (**DAML**) e a *Ontology Inference Layer* (**OIL**), foi desenvolvida conjuntamente pela *US Defence Advanced Research Project Agency* (**DARPA**) e pela *EU Information Society Technologies* (**IST**). Mais tarde surgiu a linguagem *Web Ontology Language* (**OWL**), uma extensão à DAML+OIL criada pelo W3C. O OWL tornou-se a principal tecnologia base para a Web Semântica. Hoje em dia as ontologias são aplicadas em varias áreas como inteligência artificial, engenharia de sistemas, engenharia de software, bioinformática, processamento de linguagens, comércio electrónico, educação e a já mencionada Web Semântica (Gómez-Pérez et al., 2007).

3.2 Trabalho Relacionado - Ontologias QoS

Algumas propostas de ontologias relacionadas com QoS foram desenvolvidas pela comunidade científica. Parte das ontologias encontradas focalizam-se no fornecimento de QoS a *Web Services* (**WS**). Outras focalizam-se mais na conceptualização de SLAs/SLSs e doutras entidades relacionadas com redes multiserviço.

A `QoSOnt` (Dobson et al., 2005) é uma ontologia desenvolvida em OWL, que se centra na comparação de métricas e definição de requisitos. Para o propósito de extensibilidade, a ontologia foi dividida em diferentes camadas estruturais: a camada base, do domínio de utilização, de atributos e a camada de unidades. Esta estrutura permite a substituição de uma das camadas de acordo com as necessidades de utilização. Apesar de esta ontologia providenciar uma semântica correcta em relação à comparação de requisitos, isto nunca é mostrado devido à limitação de definição de tipos no OWL 1.0. Para ultrapassar este problema foi

criada uma solução, recorrendo ao XML, perdendo assim algumas das vantagens oferecidas pelo OWL (Dobson e Sanchez-Macian, 2006).

A `DAML-QoS` (Zhou et al., 2004) é uma ontologia, que descreve métricas QoS para WS, desenvolvida usando DAML+OIL, com objectivo de integração na *framework* DAML-S (antecedente da OWL-S). A ontologia está dividida em três camadas: `QoSProfile`, `QoSPropertyDefinition` e `QoSMetrics`. Em `QoSProfile` são definidas classes que descrevem o cenário aplicacional de utilização da ontologia. Poderá ser num cenário de inquirição por parte do cliente, de anúncio de garantias de QoS ou de definição de *templates*. A camada de `QoSPropertyDefinition` corresponde ao domínio das regras das propriedades QoS, ou seja as condições aplicadas na medição e comparação das mesmas. As definições das propriedades QoS utilizadas encontram-se na camada de `QoSMetrics`. O conceito de métrica está dividido em métricas complexas e métricas atómicas. As métricas atómicas correspondem a métricas directamente medidas, as métricas complexas são compostas por métricas atómicas ou complexas e um operador lógico que as relaciona. Em (Zhou et al., 2005) é apresentado o conceito de *Service Level Object* (**SLO**), semantização de aspectos de monitorização e cálculo estatístico. Através da comparação de SLOs é possível inferir sobre os níveis de desempenho dos WSs requeridos que estão a ser ou não ser cumpridos. Foram apontados dois problemas a este modelo em (Dobson e Sanchez-Macian, 2006). Primeiro, neste modelo não existe uma relação explícita entre a definição de métricas e sobre o que elas medem. Segundo, a utilização da cardinalidade para impor restrições quantitativas às métricas QoS definidas corresponde a uma prática incorrecta dentro da convenção OWL. Actualmente, é pouco viável o desenvolvimento de ontologias com a finalidade de integração na *framework* OWL-S, que foi definitivamente abandonada quando o W3C adoptou oficialmente o SA-WDSL.

MOQ (Kim et al., 2007) é uma outra proposta de um modelo semântico sobre aplicação de QoS a WS, mas não é exactamente uma ontologia. Apenas especifica axiomas e não apresenta uma taxonomia concreta ou um dicionário de conceitos. Contudo, esta proposta aprofunda o conceito de métricas compostas e rastreabilidade de serviço. MOQ está dividida em modelo de requisitos, de monitorização e modelo de rastreabilidade. Através de axiomas definidos nesta proposta é possível a comparação de requisitos, identificação do nível de complexidade de requisitos, satisfação dos mesmos através do estabelecimento de pontos de conformidade e rastreio à utilização dos serviços.

A ontologia MonONTO (Moraes et al., 2008) aponta para a construção de uma base de conhecimento por forma a suportar um sistema de recomendação de especificações de serviço ao cliente. A ontologia é composta pelos seguintes conceitos chave: `ApplicationCharacteristics`, `TrafficCharacteristics`, `User`, `Application`, `NetworkEntity`, `NetworkCharacteristics`, `MeaurementTool`. A

ontologia serve de apoio a uma ferramenta de suporte à decisão, providenciando informação de alto nível ao utilizador sobre conformidade da rede em relação aos níveis de serviço exigidos. Este processo é essencialmente realizado através da comparação individual das características da rede contra as características de tráfego exigidas pelo serviço ou aplicação. Esta comparação dá-se entre indivíduos dos conceitos descritivos de métricas QoS. Alguns dos indivíduos do conceito `NetworkCharacteristics` relacionam-se com indivíduos da classe `MonitoringTools` com o objectivo de monitorização de serviços. Desta forma conceptualiza-se o aspecto da monitorização presente na gestão de serviços. Esta ontologia foi desenvolvida em OWL e SWRL.

Em (Alípio et al., 2006) é proposta uma ontologia que visa a automatização dos processos de gestão de serviços e mapeamento dos requisitos de serviços na rede. A ontologia é dividida em três perspectivas: classificação de serviços de rede, definição de contractos, desenvolvimento e implementação de serviços na rede. Esta ontologia foi desenvolvida em Flora-2 (baseada em F-Logic e na *framework Transaction Logic*). Apesar de uma linguagem mais poderosa, a F-Logic carece das características de interoperabilidade e reutilização presentes nas tecnologias de Web Semântica.

Em (Green, 2006) é apresentado um conjunto de ontologias que constitui parte de uma *framework* de suporte à construção de SLAs. Por exemplo, a ontologia `Unit` funciona como base para a criação de o todo tipo de elementos comparáveis presentes num SLA. Para este efeito, é possível definir qualquer tipo de unidades mensuráveis, comparadores suportados por essas unidades e a criação de operações de comparação sobre as mesmas. Desta ontologia provêem outras ontologias como: ontologia `Temporal` para elementos temporais como eventos ou intervalos de tempo, a ontologia `Network Units` para elementos relativos a redes de telecomunicações e a ontologia SLA que apresenta uma definição básica de SLAs. Em vez de uma ontologia no seu todo, são propostas uma série de ontologias com vista a proporcionar apoio a outras soluções de modelação semântica de aspectos referentes ao QoS.

Em (Royer et al., 2008) é proposta uma ontologia de SLAs, concentrando-se mais em aspectos de *Authentication*, *Authoring* e *Accounting* (AAA). É aplicada uma solução que tem como base a conceptualização do perfil de cliente para efeitos de autenticação e permissão de usabilidade do serviço. Os perfis de utilizador são negociados num contracto de serviço entre o fornecedor e o cliente. O contrato de serviço também se relaciona com um conjunto de SLAs para negociação dos níveis de serviço requeridos. Os indivíduos da classe SLS podem ser do tipo Melhor-Esforço ou Serviços Diferenciados. Os SLS Diferenciados são aplicados a serviços que requerem um tratamento diferenciado de forma a obedecer aos requisitos de QoS contratados. Os parâmetros QoS podem ser exprimidos de uma forma quanti-

tativa ou qualitativa. Existe a opção de restringir um determinado SLS a um perfil de utilizador. Por razões de escalabilidade são agrupados utilizadores consoante o seu perfil. Desenvolvida em OWL, esta ontologia destaca-se pela sua focalização em aspectos relativos ao Controlo de Admissão.

A ontologia, desenvolvida em OWL, `NetQoSOnt` (Prudencio et al., 2009) pretende fornecer apoio a ferramentas de suporte à decisão através da comparação e combinação de requisitos de qualidade de serviço. Promove a definição de SLS, contendo vários tipos de parâmetros de qualidade pertencentes a diferentes camadas de nível de serviço: Qualidade de Experiência, Qualidade ao Nível Aplicacional, Qualidade ao Nível de Rede e Qualidade ao Nível de Ligação. Em `NetQoSOnt` é apresentado o conceito Camada e por cada tipo deste conceito resulta um módulo diferente. Desta forma, a estrutura da ontologia `NetQoSOnt` tenta emular a pilha protocolar TCP/IP. Um módulo base providencia o esqueleto para a construção das várias camadas especificadoras de nível de serviço. A classe `QoSSpec` serve de superclasse para todas as especificações QoS. Para a definição de métricas é usada ontologia *Measurement Units Ontology* (**MUO**). Os requerimentos são comparados camada a camada.

Grande parte das propostas analisadas centram-se na descrição de propriedades QoS e até mesmo numa conceptualização mais abstracta do próprio serviço de rede como em (Prudencio et al., 2009). Por outro lado, o trabalho desenvolvido nesta dissertação incide sobretudo em aspectos de definição e gestão de SLSs, de classes de serviço e mapeamento serviço-rede. Também houve a preocupação de possibilitar a expansão do modelo e a integração com outras propostas dentro do domínio das redes multiserviço.

3.3 Sumário do Capítulo

Neste capítulo foram apresentados alguns conceitos gerais sobre ontologias. A sua definição, composição, os diferentes paradigmas e linguagens utilizadas na sua representação. Verificou-se também a diversidade de perspectivas nas várias propostas de ontologia no domínio de QoS encontradas na comunidade científica.Face às propostas apresentadas, verificaram-se algumas lacunas no que se refere à modelação dos vários elementos que compõem uma rede multiserviços e da sua relação com a definição de contrato de serviços no âmbito de QoS. Tendo identificado o conceito de ontologia e vários exemplares no mesmo domínio de conhecimento passa-se à descrição do modelo semântico proposto nesta dissertação.

Capítulo 4

Ontologia para Redes Multiserviço

4.1 Tecnologias Utilizadas e Planeamento

Nesta secção foram apresentadas as tecnologias utilizadas no desenvolvimento do modelo semântico de redes multiserviço: o OWL como linguagem de expressão do modelo, o SWRL como linguagem para a definição de regras de inferência sobre a base de conhecimento e o Protégé como ferramenta que facilita a construção da ontologia. Tal como na engenharia de software, também existem várias metodologias para o planeamento e desenvolvimento de ontologias. Uma breve explicação de da metodologia utilizada neste trabalho e a informação resultante do planeamento do mesmo, também são apresentadas nesta secção.

4.1.1 OWL

O *Ontology Web Language* (**OWL**) é a linguagem aprovada pelo W3C, que tendo evoluido do DAML+OIL, serve como uma das linguagens base para a Web Semântica. O OWL tem como base o RDF e o RDF Schema. O RDF é uma linguagem baseada em XML orientada a modelos de dados, relacionando recursos Web em triplos (sujeito, predicado, objecto), obtendo-se na totalidade um grafo de triplos. O RDF Schema providencia as definições de classe e propriedade, valorizando semanticamente RDF. Por cima deste, o OWL enriquece o vocabulário semântico com a definição de relações entre indivíduos, restrições de cardinalidade, equivalência, tipagem mais rica das propriedades, características das propriedades e classes enumeradas (van Harmelen e McGuinness, 2004) (Gómez-Pérez et al., 2007).

Existem três variantes do OWL: OWL Lite, OWL DL, OWL Full (van Har-

melen e McGuinness, 2004). O OWL Lite apenas suporta hierarquias simples e impõe mais restrições ao nível das relações (ex: apenas suporta a cardinalidade 0 ou 1). OWL DL implementa expressividade máxima mantendo computabilidade e decidibilidade, ou seja as inferências são computáveis e em tempo finito. OWL Full permite a liberdade sintáctica do RDF, mas com perda computacional e de decidibilidade. O OWL Full é mais apropriado para integração de documentos RDF sem a preocupação das restrições impostas pelas variações DL e Lite.

Em OWL, os conceitos são conhecidos como classes, que são criadas através da definição de condições necessárias e suficientes (classes definidas) e/ou condições necessárias (classes primitivas). Estas condições podem ser compostas por classes, restrições de propriedades (relações e atributos) e enumeração de indivíduos. Os elementos (átomos) que compõem a condição são relacionados entre si através de conjunção, disjunção e complementaridade. Existem dois tipos de propriedades: relações cujo contradomínio é uma ou mais classes ou atributos cujo contradomínio é um tipo de dados. As propriedades são restringidas através de condições existenciais, de universalidade, de cardinalidade ou a atribuição de um valor concreto. As relações podem conter regras lógicas de funcionalidade, transitividade, reflexividade e simetria. Os indivíduos (instancias de classes) são criados através de notações RDF (van Harmelen e McGuinness, 2004).

Uma das vantagens do OWL é característica de *Open World Assumption* (**OWA**). Contrariamente a linguagens como o SQL. Isto significa que não podem ser feitos pressupostos do que não existe explicitamente e do que não pode ser inferido por factos confirmados, i.e., o que não pode ser provado verdadeiro não significa que seja falso (Dobson et al., 2005).

Como resultado da experiência de vários anos utilização do OWL, o OWL 2 foi desenvolvido para compensar algumas carências de primeira versão, trazendo novas características à notação OWL. A nova característica mais importante é introdução de perfis OWL, que funcionam como sub-linguagens OWL, trazendo novos níveis de eficiência e decidibilidade à custa do nível expressividade da linguagem, conforme as necessidades da aplicação. Existem três novos perfis: OWL 2 EL, OWL 2 QL e OWL RL. O OWL 2 EL é indicado para ontologias com um grande numero de classes e propriedades. O tempo computacional nos problemas de inferência é de proporcionalidade polinomial ao tamanho da ontologia. O perfil OWL 2 QL é indicado a ontologias com um grande número de indivíduos. Este perfil OWL 2 está optimizado a nível de consulta de informação, perdendo algum poder de expressividade da linguagem. O OWL 2 QL também permite a reescrita de *queries* em outras linguagem de consulta como o SQL. O OWL 2 RL permite inferência escalável sobre ontologia sem tirar muito do poder de expressividade da linguagem. Este perfil suporta implementação de motores de inferência baseados em regras (Motik et al., 2009a).

Outra das principais novas características é a restrição de propriedade conhecida como *Key*. Com esta restrição, que simula a propriedade inversamente funcional do RDFS, verifica-se que se as instâncias de uma propriedade contêm o mesmo valor como contradomínio, então os domínios dessas mesmas instâncias são equivalentes. Esta restrição permite a definição de uma chave identificadora de um indivíduo (Motik et al., 2009b).

4.1.2 SWRL

A SWRL consiste numa linguagem de concepção de regras envolvendo elementos da ontologia com o fim de inferir sobre a mesma. Resulta da combinação entre as linguagens OWL DL e Lite com *Unary/Binary Datalog RuleML* (sublinguagem da *Rule Markup Language*). As regras são compostas por condições antecedentes (corpo) e condições consequentes (cabeça). Entre corpo e cabeça verifica-se uma relação lógica de implicação, ou seja, se as condições antecedentes se verificam, logo as condições consequentes também se verificam. Ambos, corpo e cabeça são compostos por zero ou mais átomos. Um átomo pode ter a forma de `Classe(x)`, `Propriedade(x,y)`, `sameAs(x,y)`, `differentFrom(x,y)` ou uma função integrada (chamada *BuiltIn*) com um número ilimitado de argumentos (Horrocks et al., 2004).

A especificação base do SWRL apresenta algumas limitações. O SWRL não suporta definição de regras de suporte a propriedades que dependam do contexto (por exemplo, da hora ou localização actual) (Zwaal, 2006), não é possível a aplicação de quantificadores e a criação de indivíduos. Os dois últimos problemas são resolvidos com extensões ao SWRL, mas com a desvantagem de cada umas das extensões ser suportada por motores de inferências distintos.

4.1.3 Protégé

Protégé é uma ferramenta *open-source*, desenvolvida na Standord Medical Informatics, que serve de apoio visual na criação e edição de ontologias. A ferramenta suporta dois tipos de paradigmas: Protégé-Frame para desenvolvimento de ontologias baseado no paradigma *Frame-based* e Protégé-OWL para o desenvolvimento de ontologias, de acordo com os princípios de Web Semântica.

A variante Protégé-OWL (de maior interesse para este trabalho) permite trabalhar sobre ontologias OWL, mais concretamente na sub-linguagem OWL DL. O Protégé-OWL facilita a criação e edição de componentes OWL (classes, indivíduos, relações e atributos). O utilizador não necessita possuir o conhecimento profundo da linguagem, mas sim das regras que a regem. O Protégé também

permite a inclusão e produção de *plugins* para a adição de novas funcionalidades à ferramenta e à integração de motores de inferência com o FACT++, o Pellet ou o Hermit. Neste momento estão disponíveis as versões Protégé 3 para apoio a ontologias OWL e Protégé 4 para apoio a ontologias OWL 2.

Foram desenvolvidas versões da ontologia para Protégé 3 e para Protégé 4. A versão da ontologia para Protégé 3 foi produzida com intenção de efectuar testes iniciais, dado que grande parte dos componentes presentes foram abandonados no desenvolvimento da versão 4 do Protégé, como por exemplo, o apoio à criação de indivíduos, através de formulários personalizados e a inquirição da base de conhecimento por meio de *queries* SPARQL.

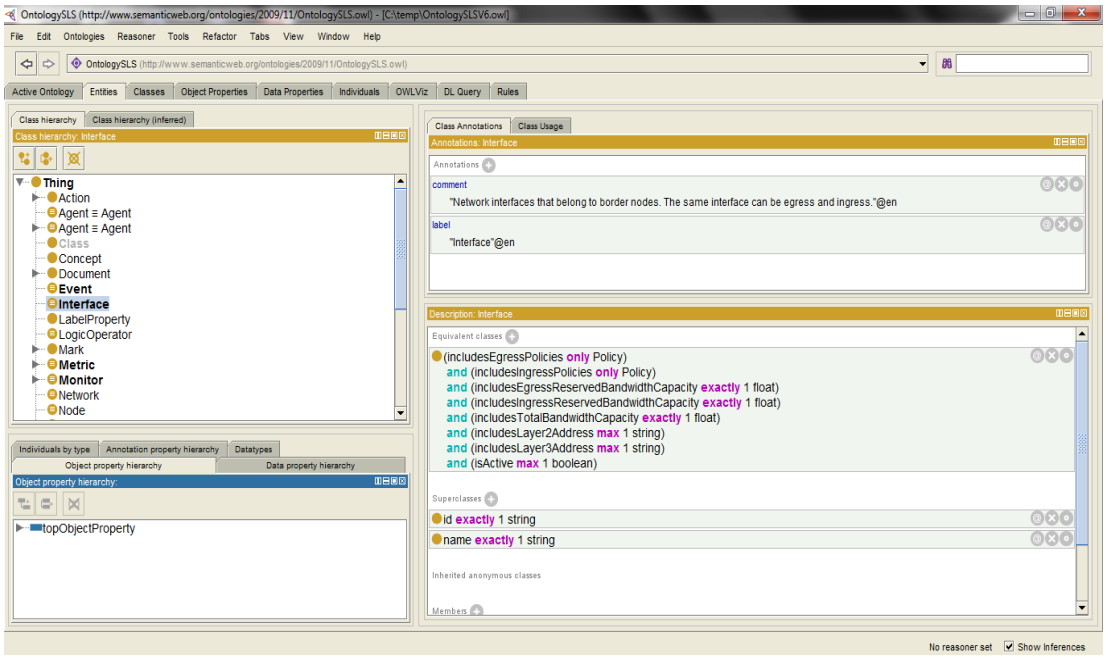

Figura 4.1: Interface gráfico da aplicação Protégé.

4.1.4 METHONTOLOGY

Para desenho do modelo semântico foi usada a metodologia METHONTOLOGY (Gómez-Pérez et al., 2007). Esta metodologia é um método para desenvolvimento de ontologias criado pelo grupo de Ontologias da Universidade Politécnica de Madrid. A METHONTOLOGY baseia-se fortemente em metodologias ágeis de desenvolvimento de software em metodologias de engenharia do conhecimento. O desenvolvimento da ontologia é um processo iterativo, permitindo a adição, modificação e remoção de componentes da ontologia em cada iteração. O processo

está dividido em três grupos de actividades: de gestão (controlo e garantia de qualidade), de desenvolvimento (especificação, conceptualização, formalização, implementação e manutenção) e de suporte (aquisição de conhecimento, integração, avaliação, documentação, e gestão de configuração). As actividades de gestão e de suporte decorrem em paralelo à sequência das de desenvolvimento ao longo do ciclo de vida do processo. Na Figura 4.2 pode-se observar a estrutura da metodologia e o relacionamento entre as várias actividades.

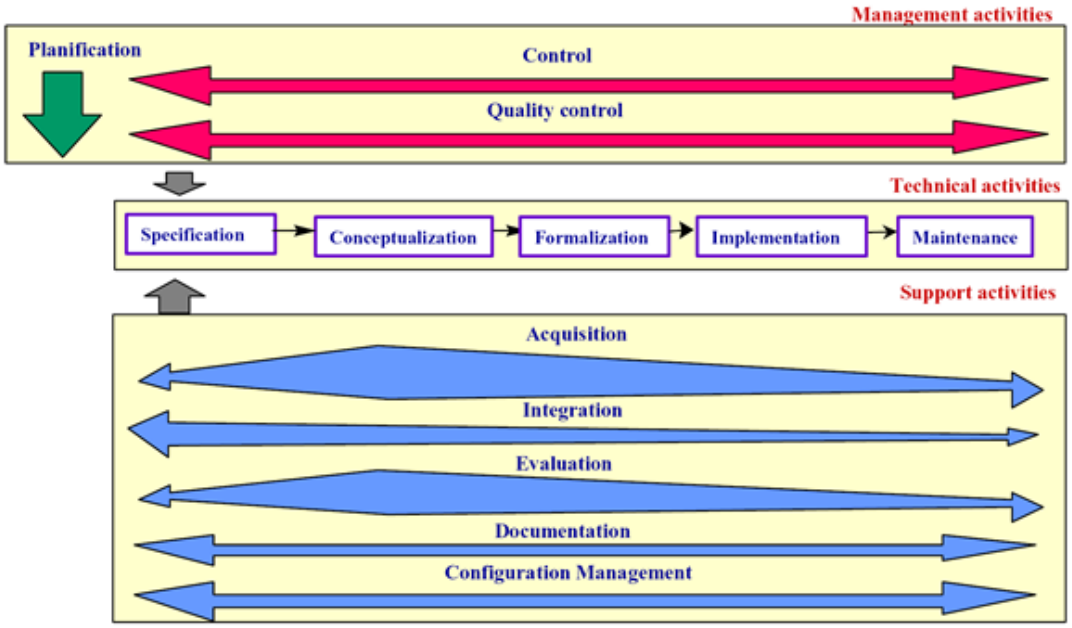

Figura 4.2: Modelo do processo METHONTOLOGY (Gómez-Pérez et al., 2007).

As diversas fases dentro da actividade de conceptualização sao adicionadas ou omitidas conforme as necessidades da ontologia a construir e da tecnologia utilizada. De seguida são descritas as várias fases envolvidas no planeamento do modelo desenvolvido nesta dissertação.

4.1.4.1 Especificação

Neste trabalho planeia-se o desenvolvimento de modelo semântico de serviços em redes multiserviço de modo a facilitar a definição de contratos de serviço e a sua implementação na rede, assim como permitir a sua consulta, por parte do cliente e fornecedor, acerca da informação relativa à gestão serviços e ao estado dos vários elementos que compõem a rede multiserviços.

Perguntas de Competência

As perguntas de competência ajudam a estabelecer objectivos e a delimitar o domínio de acção da ontologia. Equivalem a perguntas de consulta usando a própria ontologia. Neste estudo, em concreto, as perguntas visam ilustrar que tipo de informação é requerido ao modelo por parte do cliente e fornecedor de serviços. No entanto, a informação requerida pelos dois actores envolvidos no processo não tem o mesmo nível de exigência, havendo uma separação de perguntas aplicadas pelo cliente e pelo fornecedor de serviço.

Perguntas Cliente:

- Que pacotes de serviços são oferecidos?
- Que serviços estão contidos no pacote de serviços **x**?
- Qual o nível de tolerância do serviço **x** a atrasos?
- Qual o nível de tolerância do serviço **x** a *jitter*?
- Qual o nível de tolerância do serviço **x** a perdas?
- Qual a largura de banda mínima reservada para serviço **x**?
- Qual o nível de fiabilidade do serviço **x**?

Perguntas Fornecedor de Serviço:

- Que pacotes de serviços são oferecidos?
- Que serviços estão contidos no pacote de serviços **x**?
- Qual o nível de tolerância do serviço **x** a atrasos?
- Qual o nível de tolerância do serviço **x** a *jitter*?
- Qual o nível de tolerância do serviço **x** a perdas?
- Qual a largura de banda mínima reservada para serviço **x**?
- Qual o nível de fiabilidade do serviço **x**?
- Qual o *scope* do serviço **x**?
- Quais os parâmetros de classificação para o serviço **x**?
- Quais os parâmetros de condicionamento de tráfego para o serviço **x**?

- Que tempo e métricas são monitorizadas ao serviço **x** do cliente **y**?
- Quais as interfaces do nó de rede **n**?
- Que parâmetros de classificação estão presentes na interface **n.i**?
- Que parâmetros de condicionamento estão presentes na interface **n.i**?
- Quanta largura de banda está disponível na interface **n.i**?
- Que serviços entram pela interface **n.i**?
- Que serviços saem pela interface **n.i**?

4.1.4.2 Conceptualização

Fase 1 - Glossário de termos

Identificação de todos os termos do domínio de conhecimento relevantes para a construção da ontologia. São listados todo o tipo de termos: conceitos, atributos e relações. As tabelas criadas nesta fase encontram-se na secção A.1.

Fase 2 - Taxonomia de conceitos

Identificação da hierarquia de conceitos presente na ontologia com a construção de diagramas demonstrativos da taxonomia de conceitos. Os diagramas criados nesta fase encontram-se na secção A.2.

Fase 3 - Relações binárias ad hoc

Esta fase consiste no estabelecimento de relações binárias entre conceitos. Este processo é realizado com construção de diagramas de relações e utilizando os termos do tipo relação especificados na Fase 1. Os diagramas criados nesta fase encontram-se na secção A.3.

Fase 4 - Dicionário de conceitos

Com as relações binárias entre conceitos estabelecidas, procede-se ao estabelecimento de ligações entre conceitos e as suas propriedades (atributos e relações). Os atributos e relações correspondem aos termos que descrevem o conceito na ontologia. As tabelas criadas nesta fase encontram-se na secção A.4.

Fase 5 - Descrição de relações ad-hoc

Nesta fase procede-se à definição em pormenor das relações ad hoc presentes no dicionário de conceitos. Para cada relação é importante definir o seu domínio, contradomínio e cardinalidade. Algumas das podem também conter uma relação inversa. As tabelas criadas nesta fase encontram-se na secção A.5.

Fase 6 - Descrição de atributos

Esta fase corresponde à descrição de atributos incluídos no dicionário de conceitos. Para cada atributo é necessário especificar o nome, o conceito correspondente ao domínio da propriedade, o tipo de valor, a cardinalidade da propriedade. Em alguns casos (principalmente em tipos numéricos), também é descrita a unidade da medida. Algumas delas podem também conter uma relação inversa. As tabelas criadas nesta fase encontram-se na secção A.6.

Fase 7 - Descrição de regras

Nesta fase procede-se à descrição de regras necessárias na ontologia. Para cada regra são ainda identificados as variáveis, conceitos, atributos, relações e funções integradas (*Buit-Ins*) envolvidas na sua definição. As tabelas criadas nesta fase encontram-se na secção A.7.

4.2 Descrição da ontologia

Por questões de organização e extensibilidade a ontologia está dividida em dois módulos: rede multiserviço e gestão de serviços. A organização dos módulos é equivalente a uma estrutura em camadas, na qual a camada superior (gestão de serviços) depende parcialmente da camada inferior (rede multiserviço). A estrutura modular da ontologia permite alguma independência entre elementos que compõem o modelo, permitindo redefinição de cada módulo conforme futuras necessidades.

O módulo de rede funciona como camada base. Contém todos os elementos referentes à implementação física da rede do domínio, como os elementos pertencentes à topologia da rede, configurações e mecanismos para a implementação de serviços na rede. O módulo de gestão é composto por elementos referentes à gestão de serviços, passando pela conceptualização de contratos de serviço, monitorização e outros elementos de apoio à gestão de serviços de rede. Neste módulo também permite a definição de classes de serviço por parte do fornecedor de rede.

A divisão modular também está de acordo com tipo de entidades que interagem com os elementos de cada módulo. O módulo de redes multiserviço compreende a base de conhecimento que retira a informação da própria rede física do fornecedor. O módulo de gestão serve de apoio a uma aplicação para a definição e gestão de serviços de rede.

4.2.1 Módulo de Rede

Como foi descrito anteriormente, este módulo contém todos os conceitos relacionados com redes multiserviço dentro da ontologia. Nesta fase do trabalho, a rede é representada pelos nós de fronteira e consequentemente pelas interfaces de entrada/saída do domínio.Assume-se também que existe pelo menos um caminho entre cada nó de fronteira. Nesta fase são modelados também os mecanismos QoS correspondentes à fase de classificação e condicionamento de tráfego.

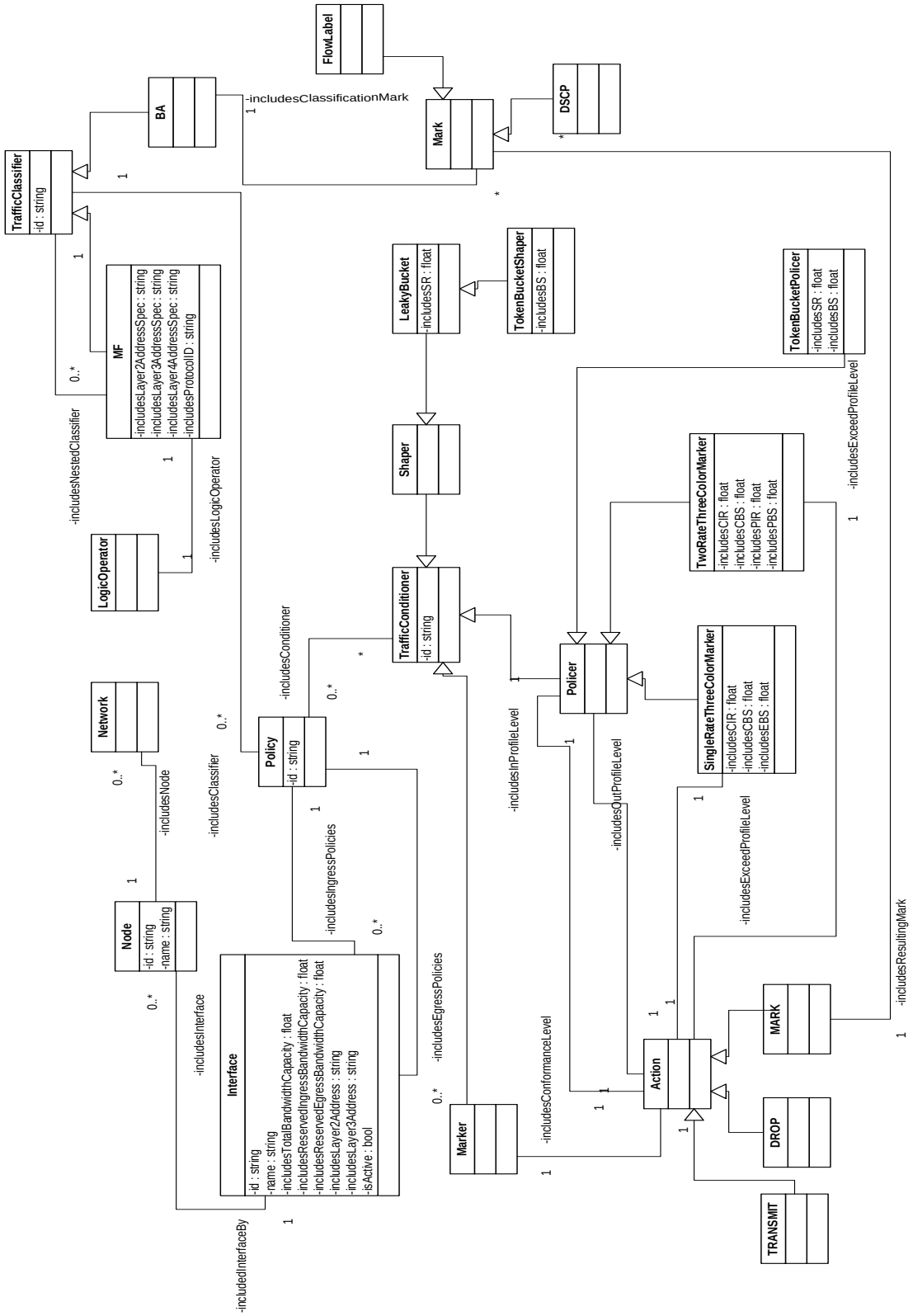

Figura 4.3: Diagrama do módulo de rede.

4.2.1.1 Network

Conceito representativo de um domínio de rede do fornecedor de serviço. Contém um número ilimitado de nós de rede.

Tabela 4.1: Conceito Network

Nome da classe	Network
Condição necessária e suficiente (intersecção)	
includesNodes	
Condição necessária (intersecção)	
Não Existente	
Atributos	
id *Tipo:* string *Cardinalidade:* 1	
name *Tipo:* string *Cardinalidade:* 1	
Relações	
includesNodes *Contradomínio:* Node *Cardinalidade:* 1 ... *	
Superclasses	*Não Existente*
Subclasses	*Não Existente*
Instâncias	*Não Existente*

4.2.1.2 Node

Representa um nó de fronteira (ou seja, nó de entrada ou saída de tráfego no domínio) dentro da rede do fornecedor de serviço. Contém um número ilimitado de interfaces. As interfaces de um nó representadas na ontologia referem-se apenas às interfaces de entrada e saída do domínio.

Tabela 4.2: Conceito Node

Nome da classe	Node
Condição necessária e suficiente (intersecção)	
includesInterfaces	
Condição necessária (intersecção)	
id name	
Atributos	
id *Tipo:* string *Cardinalidade:* 1	
name *Tipo:* string *Cardinalidade:* 1	
Relações	
includesInterfaces *Contradomínio:* Interface *Cardinalidade:* 1 ... *	
Superclasses	*Não Existente*
Subclasses	*Não Existente*
Instâncias	*Não Existente*

4.2.1.3 Interface

Representa uma interface de rede de entrada e saída do domínio do fornecedor de serviço. Não existe separação conceptual dos dois sentidos de fluxo de tráfego, numa interface existe sempre o sentido de entrada e de saída de tráfego. Existe no entanto a separação de políticas impostas tanto num sentido como no outro. Para cada interface é especificada a capacidade total de largura de banda e a capacidade de largura de banda reservada nos dois sentidos. A capacidade total de largura de de banda é sempre a mesma nos dois sentidos. A capacidade de largura de banda reservada refere-se à acumulação de larguras de banda mínimas garantidas aos serviços no âmbito da interface, podendo ser diferente nos dois sentidos. Contudo, em termos futuros poder-se-á implementar uma solução que passa pelo cálculo dinâmico da largura de banda reservada momentaneamente na interface de forma a fornecer apoio a um sistema de controlo de admissão. A classe interface também contém representações dos seus endereços a nível da camada de ligação e de rede. O atributo booleano `isActive` indica se a interface está no âmbito de algum serviço de rede.

Tabela 4.3: Conceito Interface

Nome da classe	Interface
Condição necessária e suficiente (intersecção)	
includesLayer2Address includesLayer3Address includesIngressPolicies includesEgressPolicies includesTotalBandwidthCapacity includesIngressReservedBandwidthCapacity includesEgressReservedBandwidthCapacity isActive	
Condição necessária (intersecção)	
id name	
Atributos	
Continua na página seguinte	

Tabela 4.3 – continuação da página anterior

id *Tipo:* string *Cardinalidade:* 1
name *Tipo:* string *Cardinalidade:* 1
includesTotalBandwidthCapacity *Tipo:* float *Cardinalidade:* 1
includesIngressReservedBandwidthCapacity *Tipo:* float *Cardinalidade:* 1
includesEgressReservedBandwidthCapacity *Tipo:* float *Cardinalidade:* 1
includesLayer2Address *Tipo:* string *Cardinalidade:* 0 ... 1
includesLayer3Address *Tipo:* string *Cardinalidade:* 0 ... 1
isActive *Tipo:* boolean *Cardinalidade:* 1
Relações
includesIngressPolicies *Contradomínio:* Policy
Continua na página seguinte

Tabela 4.3 – continuação da página anterior

Cardinalidade: 0 ... *	
includesEgressPolicies *Contradomínio:* Policy *Cardinalidade:* 0 ... *	
Superclasses	*Não Existente*
Subclasses	*Não Existente*
Instâncias	*Não Existente*

4.2.1.4 Policy

Principal função desta classe é a combinação de um classificador com um conjunto de condicionadores de tráfego. O classificador identifica o fluxo em questão e é aplicado o condicionador sobre esse fluxo. Opta-se pela combinação com uma série de condicionadores, porque podem ser aplicados vários sobre um fluxo identificado. Por exemplo, pode ser aplicado um *shaper* e depois os pacotes serem remarcados.

Tabela 4.4: Conceito Policy

Nome da classe	Policy
Condição necessária e suficiente (intersecção)	
includesClassifier includesConditioner	
Condição necessária (intersecção)	
id	
Atributos	
id *Tipo:* string *Cardinalidade:* 1	
	Continua na página seguinte

Tabela 4.4 – continuação da página anterior

Relações	
includesClassifier *Contradomínio:* Classifier *Cardinalidade:* 1	
includesConditioner *Contradomínio:* Conditioner *Cardinalidade:* 1 ... *	
Superclasses	*Não Existente*
Subclasses	*Não Existente*
Instâncias	*Não Existente*

4.2.1.5 TrafficClassification

Classe na qual se definem regras para a identificação do tráfego com o objectivo de tratamento diferenciado. Representa o conceito de classificador de tráfego. Os dois tipos de classificação de tráfego são representados pelas subclasses: BA e MF.

4.2.1.6 BA

Classificador do tipo *Behaviour Aggregate* (BA), no qual o tráfego é identificado através de uma marca. Contém um atributo correspondente à marca utilizada para identificação do tráfego.

Tabela 4.5: Conceito BA

Nome da classe	BA
Condição necessária e suficiente (intersecção)	
TrafficClassifier includesClassificationMark	
Continua na página seguinte	

Tabela 4.5 – continuação da página anterior

Condição necessária (intersecção)	
id	
Atributos	
id *Tipo:* string *Cardinalidade:* 1	
Relações	
includesClassificationMark *Contradomínio:* Mark *Cardinalidade:* 1	
Superclasses	TrafficClassifier
Subclasses	*Não Existente*
Instâncias	*Não Existente*

4.2.1.7 MF

Classificador do tipo *Multi-Field* (MF), no qual o tráfego é identificado através de vários parâmetros definidos. Não existe limite no número de parâmetros usados para a classificação de tráfego. A regra de classificação pode ser composta por um ou mais parâmetros. No caso de mais do que dois parâmetros, é necessária a definição de um operador lógico que os combine. Os tipos de parâmetros consistem, por exemplo, em endereços protocolares ou identificadores de protocolo. No caso de endereços, é feita a distinção entre endereços de origem ou de destino. Este tipo de classificador pode conter outras sub-regras de qualquer tipo (instâncias da classe `TrafficClassifier`). Essas sub-regras são relacionadas através do operador lógico.

Tabela 4.6: Conceito MF

Nome da classe	MF
Condição necessária e suficiente (intersecção)	
TrafficClassifier (includesClassificationFields) ∨ ((includesClassificationFields) ∧ (includesLogicOperator)) includesNestedClassifier	
Condição necessária (intersecção)	
id	
Atributos	
id *Tipo:* string *Cardinalidade:* 1	
includesClassificationFields *Tipo:* string *Cardinalidade:* 0 ... *	
Relações	
includesNestedClassifier *Contradomínio:* TrafficClassifier *Cardinalidade:* 0 ... *	
includesLogicOperator *Contradomínio:* LogicOperator *Cardinalidade:* 0 ... 1	
Superclasses	TrafficClassifier
Subclasses	*Não Existente*
Instâncias	*Não Existente*

4.2.1.8 LogicOperator

Classe enumerada com a representação dos operadores lógicos de conjunção e de disjunção. Este operadores lógicos são utilizados no relacionamento dos vários parâmetros que compõe um classificador de tráfego do tipo MF.

Tabela 4.7: Conceito LogicOperator

Nome da classe	LogicOperator
Condição necessária e suficiente (intersecção)	
{AND,OR}	
Condição necessária (intersecção)	
Não Existente	
Atributos	
Não Existente	
Relações	
Não Existente	
Superclasses	*Não Existente*
Subclasses	*Não Existente*
Instâncias	AND, OR

4.2.1.9 Mark

Abrange alguns dos campos protocolares utilizados na marcação agregada de tráfego ou de fluxos de tráfego.

4.2.1.10 DSCP

Representação da marcação agregada de tráfego através do campo DSCP do IPv4 ou Traffic Class do IPv6.

Tabela 4.8: Conceito DSCP

Nome da classe	DSCP
Condição necessária e suficiente (intersecção)	
{ EF, AF11, AF12, AF13, AF21, AF22, AF23, AF31, AF32, AF33, AF41, AF42, AF43, DF }	
Condição necessária (intersecção)	
Não Existente	
Atributos	
Não Existente	
Relações	
Não Existente	
Superclasses	Mark
Subclasses	*Não Existente*
Instâncias	EF, AF11, AF12, AF13, AF21, AF22, AF23, AF31, AF32, AF33, AF41, AF42, AF43, DF

4.2.1.11 FlowLabel

Representação de marcação de fluxos de tráfego através do campo Flow Label do IPv6.

Tabela 4.9: Conceito FlowLabel

Nome da classe	FlowLabel
Condição necessária e suficiente (intersecção)	
Não Existente	
	Continua na página seguinte

Tabela 4.9 – continuação da página anterior

Condição necessária (intersecção)	
Não Existente	
Atributos	
Não Existente	
Relações	
Não Existente	
Superclasses	*Não Existente*
Subclasses	*Não Existente*
Instâncias	*Não Existente*

4.2.1.12 TrafficConditioner

Elemento de condicionamento de tráfego, geralmente aplicado em conjunção com um classificador de tráfego. Contém três subclasses representando três tipos de condicionamento: `Marker`, `Policer` e `Shaper`.

4.2.1.13 Marker

Não é bem um condicionador dado que não mede ou condiciona parâmetros tais como a taxa de transmissão de acordo com um determinado perfil de tráfego. Foi incluído apenas para providenciar a opção de aplicar uma acção imediata sobre o tráfego sem qualquer tipo de policiamento.

Tabela 4.10: Conceito Marker

Nome da classe	Marker
Condição necessária e suficiente (intersecção)	
TrafficConditioner includesConformanceLevel	
Continua na página seguinte	

Tabela 4.10 – continuação da página anterior

Condição necessária (intersecção)	
id	
Atributos	
id *Tipo:* string *Cardinalidade:* 1	
includesClassificationFields *Tipo:* string *Cardinalidade:* 0 ... *	
Relações	
includesConformanceLevel *Contradomínio:* Action *Cardinalidade:* 1	
Superclasses	TrafficConditioner
Subclasses	*Não Existente*
Instâncias	*Não Existente*

4.2.1.14 Policer

Compara o fluxo de tráfego com perfil de tráfego predeterminado, estabelecendo níveis de conformidade do tráfego aplicando posteriormente acções imediatas sobre o tráfego específicas ao nível de conformidade. Como base, é obrigatório ter pelo menos dois níveis de conformidade: *in-profile* e *out-profile*.

4.2.1.15 TokenBucketPolicer

Divide o tráfego em dois níveis de conformidade, cada um deles com uma acção específica sobre o fluxo de tráfego. Para o policiamento de tráfego são definidos os parâmetros *Single Rate* (SR) que regula a taxa média de transmissão do fluxo e *Burst Size* (BS) para controlar a transmissão de tráfego em rajadas.

Tabela 4.11: Conceito TokenBucketPolicer

Nome da classe	TokenBucketPolicer
Condição necessária e suficiente (intersecção)	
Policer includesSR includesBS	
Condição necessária (intersecção)	
Não Existente	
Atributos	
id *Tipo:* string *Cardinalidade:* 1	
includesSR *Tipo:* float *Cardinalidade:* 1	
includesBS *Tipo:* float *Cardinalidade:* 1	
Relações	
includesInProfileLevel *Contradomínio:* Action *Cardinalidade:* 1	
includesOutProfileLevel *Contradomínio:* Action *Cardinalidade:* 1	
Superclasses	Policer
Subclasses	*Não Existente*

Continua na página seguinte

Tabela 4.11 – continuação da página anterior

Instâncias	*Não Existente*

4.2.1.16 SingleRateThreeColorMarker

Implementação do *policer Single Rate Three-Color Marker* (srTCM). O tráfego é "colorido" de acordo com três níveis de conformidade: *in-profile*, *exceed-profile* e *out-profile*. Existe apenas um parâmetro para o controlo da taxa de transmissão, o *Committed Information Rate* (CIR), e dois parâmetros "baldes" com capacidades diferentes, o *Committed Burst Size* (CBS) e o *Excess Burst Size* (EBS) para regular a dois níveis o tráfego em rajadas.

Tabela 4.12: Conceito SingleRateThreeColorMarker

Nome da classe	SingleRateThreeColorMarker
Condição necessária e suficiente (intersecção)	
Policer includesCIR includesCBS includesEBS includesExcessProfileLevel	
Condição necessária (intersecção)	
Não Existente	
Atributos	
id *Tipo:* string *Cardinalidade:* 1	
includesCIR *Tipo:* float	
Continua na página seguinte	

Tabela 4.12 – continuação da página anterior

Cardinalidade: 1	
includesCBS *Tipo:* float *Cardinalidade:* 1	
includesEBS *Tipo:* float *Cardinalidade:* 1	
Relações	
includesInProfileLevel *Contradomínio:* Action *Cardinalidade:* 1	
includesExcessProfileLevel *Contradomínio:* Action *Cardinalidade:* 1	
includesOutProfileLevel *Contradomínio:* Action *Cardinalidade:* 1	
Superclasses	Policer
Subclasses	*Não Existente*
Instâncias	*Não Existente*

4.2.1.17 TwoRateThreeColorMarker

Implementação do *policer Two Rate Three-Color Marker* (trTCM). Tem como principal característica a existência de dois parâmetros para o controlo da taxa de transmissão, o *Committed Information Rate* (CIR) e o *Peak Information Rate* (PIR), e os dois "baldes" de capacidade diferente, o *Committed Burst Size* (CBS) e o *Peak Burst Size* (PBS), para regular o tratamento médio e de pico das rajadas

de tráfego. São especificados três níveis de conformidade.

Tabela 4.13: Conceito TwoRateThreeColorMarker

Nome da classe	TwoRateThreeColorMarker
Condição necessária e suficiente (intersecção)	
Policer includesCIR includesCBS includesPIR includesPBS includesExcessProfileLevel	
Condição necessária (intersecção)	
Não Existente	
Atributos	
id *Tipo:* string *Cardinalidade:* 1	
includesCIR *Tipo:* float *Cardinalidade:* 1	
includesCBS *Tipo:* float *Cardinalidade:* 1	
includesPIR *Tipo:* float *Cardinalidade:* 1	
includesPBS *Tipo:* float	

Continua na página seguinte

Tabela 4.13 – continuação da página anterior

Cardinalidade: 1	
Relações	
includesInProfileLevel *Contradomínio:* Action *Cardinalidade:* 1	
includesExcessProfileLevel *Contradomínio:* Action *Cardinalidade:* 1	
includesOutProfileLevel *Contradomínio:* Action *Cardinalidade:* 1	
Superclasses	Policer
Subclasses	*Não Existente*
Instâncias	*Não Existente*

4.2.1.18 Shaper

Também policia o tráfego de acordo com perfis pré-estabelecidos, não é aplicada nenhuma acção imediata sobre tráfego em não conformidade. O tráfego em não conformidade é retido em fila de espera até a um limite, e transmitido quando se verificar conformidade com o perfil tráfego negociado. Para esse efeito, não existe especificação de níveis de conformidade. Para a aplicação de acções sobre o tráfego transmitido pode se combinar a instância da classe `Shaper` com instâncias da classe `Marker`.

4.2.1.19 LeakyBucket

Contem um parâmetro (SR) para controlo da taxa de transmissão. É insensível à transmissão em rajada de tráfego, eliminando-a.

Tabela 4.14: Conceito LeakyBucket

Nome da classe	LeakyBucket
Condição necessária e suficiente (intersecção)	
Shaper includesSR	
Condição necessária (intersecção)	
Não Existente	
Atributos	
id *Tipo:* string *Cardinalidade:* 1	
includesSR *Tipo:* float *Cardinalidade:* 1	
Relações	
Não Existente	
Superclasses	Shaper
Subclasses	*Não Existente*
Instâncias	*Não Existente*

4.2.1.20 TokenBucketShaper

Inclui dois parâmetros: *Single Rate* (SR) e *Burst Size* (BS). Acrescenta a capacidade de permitir transmissão de tráfego em rajada até a um determinado limite. Algoritmo equivalente ao `TokenBucketPolicer`, mas aplicado ao *shaping* de tráfego.

Tabela 4.15: Conceito TokenBucketShaper

Nome da classe	TokenBucketShaper
Condição necessária e suficiente (intersecção)	
LeakyBucket includesBS	
Condição necessária (intersecção)	
Não Existente	
Atributos	
id *Tipo:* string *Cardinalidade:* 1	
includesSR *Tipo:* float *Cardinalidade:* 1	
includesBS *Tipo:* float *Cardinalidade:* 1	
Relações	
Não Existente	
Superclasses	LeakyBucket
Subclasses	*Não Existente*
Instâncias	*Não Existente*

4.2.1.21 Action

Definição de acções aplicadas sobre o tráfego. Existem três tipos de acção: `TRANSMIT`, `MARK`, `DROP`.

4.2.1.22 TRANSMIT

Acção que corresponde simplesmente à transmissão do pacote sem qualquer tipo de atribuição de marca.

Tabela 4.16: Conceito TRANSMIT

Nome da classe	TRANSMIT
Condição necessária e suficiente (intersecção)	
{TRANSMIT_PACKETS}	
Condição necessária (intersecção)	
Não Existente	
Atributos	
Não Existente	
Relações	
Não Existente	
Superclasses	Action
Subclasses	*Não Existente*
Instâncias	TRANSMIT_PACKETS

4.2.1.23 MARK

Acção que corresponde à remarcação de tráfego, quer tenha sido previamente marcado ou não.

Tabela 4.17: Conceito MARK

Nome da classe	MARK
Condição necessária e suficiente (intersecção)	
Continua na página seguinte	

Tabela 4.17 – continuação da página anterior

Action includesResultingMark	
Condição necessária (intersecção)	
Não Existente	
Atributos	
Não Existente	
Relações	
includesResultingMark *Contradomínio:* Mark *Cardinalidade:* 1	
Superclasses	Action
Subclasses	*Não Existente*
Instâncias	*Não Existente*

4.2.1.24 DROP

Acção que corresponde ao descarte de pacotes.

Tabela 4.18: Conceito DROP

Nome da classe	DROP
Condição necessária e suficiente (intersecção)	
{DROP_PACKETS}	
Condição necessária (intersecção)	
Não Existente	
Atributos	
Continua na página seguinte	

Tabela 4.18 – continuação da página anterior

Não Existente	
Relações	
Não Existente	
Superclasses	Action
Subclasses	*Não Existente*
Instâncias	DROP_PACKETS

4.2.2 Módulo de Gestão de Serviços

Este módulo contém todos os conceitos essenciais à gestão de contractos de serviço. Mais concretamente focaliza-se na conceptualização de SLSs contidos em contractos de serviço e dos elementos que os compõem. Alguns desses elementos são conceitos pertencentes ao módulo de rede. Neste módulo estão definidos os conceitos relativos às métricas essenciais para a especificação de nível de serviço e os conceitos que contém informação de monitorização de contratos de serviço e da desempenho da rede entre dois nós. Também se verifica a presença de classes definidoras de tipos de serviço oferecidos.

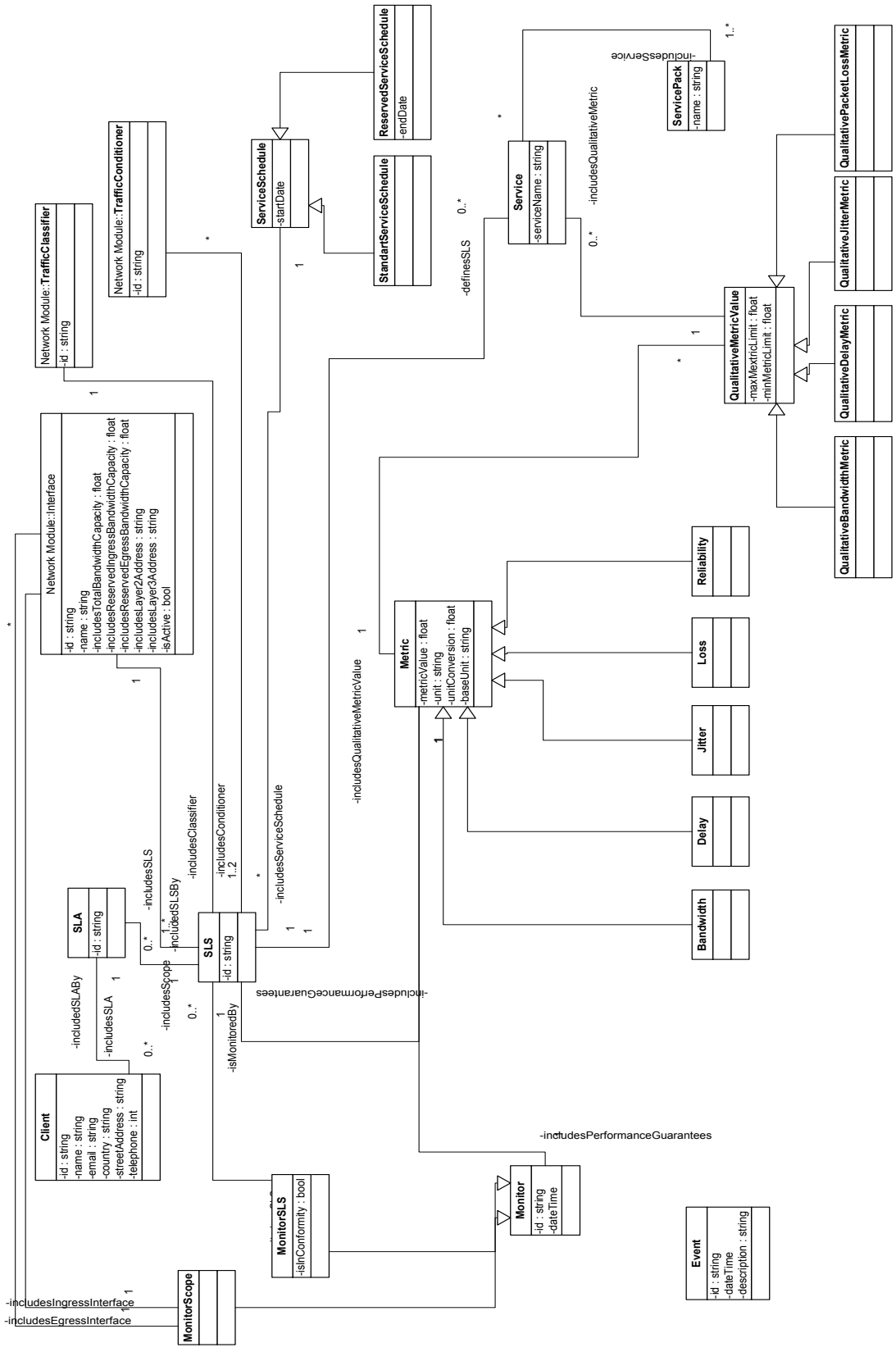

Figura 4.4: Diagrama do módulo de gestão e contractos de serviços.

4.2.2.1 Client

Representa a entidade que contrata o serviço. A classe `Client` é definida como uma subclasse de `Person`, classe da ontologia importada *Friend Of A Friend* (**FOAF**) (Brickley e Miller, 2007). Pode representar um indivíduo ou um colectivo, como por exemplo uma empresa. Um cliente está relacionado com um ou mais contratos de serviço (SLAs).

Tabela 4.19: Conceito Client

Nome da classe	Client
Condição necessária e suficiente (intersecção)	
includesSLA	
Condição necessária (intersecção)	
Não Existente	
Atributos	
Não Existente	
Relações	
includesSLA *Contradomínio:* SLA *Cardinalidade:* 1 ... *	
Superclasses	Person
Subclasses	*Não Existente*
Instâncias	*Não Existente*

4.2.2.2 SLA

Representa um contrato do nível de serviço prestado, celebrado entre um cliente e um fornecedor de serviço. Nesta ontologia não foram especificados outros parâmetros administrativos do SLA, sendo apenas considerada a sub-componente técnica (SLS). Um SLA contém um ou mais SLS. Existe também uma ligação com um cliente (o cliente com o qual se faz o contracto).

Tabela 4.20: Conceito SLA

Nome da classe	SLS
Condição necessária e suficiente (intersecção)	
includesSLS	
Condição necessária (intersecção)	
Não Existente	
Atributos	
id *Tipo:* string *Cardinalidade:* 1	
Relações	
includesSLS *Contradomínio:* SLS *Cardinalidade:* 1 ... *	
Superclasses	*Não Existente*
Subclasses	*Não Existente*
Instâncias	*Não Existente*

4.2.2.3 SLS

Especificação do nível de serviço prestado, representando a componente técnica de um SLA. Especifica os níveis de serviço propostos e as medidas a implementar a fim de honrar essas mesmas propostas. O SLS define: a esfera de acção do serviço, a classificação e condicionamento aplicados, os níveis de desempenho e fiabilidade do serviço, assim como período de prestação de serviço. Na esfera de acção são definidas as interfaces de rede de entrada e de saída, desenhando as fronteiras do domínio onde se presta o serviço. A definição das interfaces de entrada e saida podem obedecer às seguintes cardinalidades: um-um, um-muitos, muitos-um, muitos-muitos. O nível de serviço contratado é unidireccional, i.e., é aplicado ao tráfego que flui no sentido especificado. Para o tráfego que flui no sen-

tido contrária é necessária a definição de um novo SLS. As políticas de classificação e condicionamento de tráfego especificadas no SLS, normalmente são aplicadas à entrada de tráfego no domínio. Contudo, é acrescentada a possibilidade de aplicação das mesmas na saída de tráfego do domínio. Os níveis de desempenho são divididos pelas seguintes métricas: largura de banda mínima, atraso máximo permitido, variação máxima do atraso, e rácio máximo tolerado de perda de pacotes. Todas as interfaces de rede dentro do âmbito do SLS, são detectadas e identificadas através da relação `includesNonConformantInterface`, para posteriores medidas correctivas por parte da aplicação.

Tabela 4.21: Conceito SLS

Nome da classe	SLS
Condição necessária e suficiente (intersecção)	
includesScope includesTrafficIngressClassifier includesTrafficEgressClassifier includesTrafficConditioner includesPerformanceGuarantees includesReliability includesSchedule	
Condição necessária (intersecção)	
id name **Atributos**	
id *Tipo:* string *Cardinalidade:* 1	
name *Tipo:* string *Cardinalidade:* 1	
Continua na página seguinte	

Tabela 4.21 – continuação da página anterior

Relações	
includesScope *Contradomínio:* Interface *Cardinalidade:* 1 . . . *	
includesIngressTrafficClassifier *Contradomínio:* TrafficClassifier *Cardinalidade:* 1	
includesEgressClassifier *Contradomínio:* TrafficClassifier *Cardinalidade:* 0 . . . 1	
includesTrafficConditioner *Contradomínio:* TrafficConditioner *Cardinalidade:* 0 . . . *	
includesPerformanceGuaratees *Contradomínio:* Metric *Cardinalidade:* 0 . . . *	
includesReliability *Contradomínio:* Reliability *Cardinalidade:* 0 . . . *	
includesSchedule *Contradomínio:* ServiceSchedule *Cardinalidade:* 1	
isMonitorizedBy *Contradomínio:* MonitorSLS *Cardinalidade:* 0 . . . *	
Superclasses	*Não Existente*

Continua na página seguinte

Tabela 4.21 – continuação da página anterior

Subclasses	*Não Existente*
Instâncias	*Não Existente*

4.2.2.4 ServiceSchedule

Definição do período no qual o serviço é prestado. Contém duas subclasses: `StandardServiceSchedule` e `ReservedServiceSchedule`.

4.2.2.5 StandardServiceSchedule

Apenas é designada um data de iniciação da prestação de serviço. O serviço é terminado explicitamente pelo cliente.

Tabela 4.22: Conceito StandardServiceSchedule

Nome da classe	StandardServiceSchedule
Condição necessária e suficiente (intersecção)	
Não Existente	
Condição necessária (intersecção)	
Não Existente	
Atributos	
id *Tipo:* string *Cardinalidade:* 1	
startDate *Tipo:* date *Cardinalidade:* 1	
Continua na página seguinte	

Tabela 4.22 – continuação da página anterior

Relações	
Não Existente	
Superclasses	ServiceSchedule
Subclasses	*Não Existente*
Instâncias	*Não Existente*

4.2.2.6 ReservedServiceSchedule

O período de prestação de serviço é designado por um data de início e uma data de fim de serviço.

Tabela 4.23: Conceito ReservedServiceSchedule

Nome da classe	ReservedServiceSchedule
Condição necessária e suficiente (intersecção)	
Não Existente	
Condição necessária (intersecção)	
Não Existente	
Atributos	
id *Tipo:* string *Cardinalidade:* 1	
startDate *Tipo:* date *Cardinalidade:* 1	
endDate *Tipo:* date	
Continua na página seguinte	

Tabela 4.23 – continuação da página anterior

Cardinalidade: 1	
Relações	
Não Existente	
Superclasses	ServiceSchedule
Subclasses	*Não Existente*
Instâncias	*Não Existente*

4.2.2.7 Metric

Superclasse para todos as métricas representadas na ontologia e funciona como base para todos os tipos de métricas a definir. Na classe `Metric` é possível definir o valor da métrica, a unidade, unidade base, o valor de conversão da unidade definida para a unidade base e opcionalmente o valor qualitativo de métrica. O valor qualitativo pode ser calcula através da Regra 4.2.3.4. Como subclasses existem as métricas: largura de banda, atraso, variação de atraso, rácio de perda de pacotes e fiabilidade do serviço.

Tabela 4.24: Conceito Metric

Nome da classe	Metric
Condição necessária e suficiente (intersecção)	
((baseUnit) $\wedge$ (metricValue) $\wedge$ (unit) $\wedge$ (unitConversion)) $\vee$ (includesQualitativeValue)	
Condição necessária (intersecção)	
Não Existente	
Atributos	
value *Tipo:* float	
Continua na página seguinte	

Tabela 4.24 – continuação da página anterior

Cardinalidade: 1	
unit *Tipo:* string *Cardinalidade:* 1	
baseUnit *Tipo:* string *Cardinalidade:* 1	
unitConversion *Tipo:* float *Cardinalidade:* 1	
Relações	
includesQualitativeMetricValue *Contradomínio:* QualitativeMetricValue *Cardinalidade:* 1	
Superclasses	Metric
Subclasses	*Não Existente*
Instâncias	Bandwidth, Delay, Jitter, PacketLoss, Reliability

4.2.2.8 QualitativeMetricValue

Representa os símbolos usados na designação de valores qualitativos de métricas. A atribuição do valor qualitativo a uma métrica depende do limite superior e inferior definido. Os valores qualitativos são utilizados para identificar a que tipo de serviço pertence um SLS. Contém quatro subclasses: `QualitativeBandwidthMetric`, `QualitativeDelayMetric`, `QualitativeJitterMetric`, `QualitativePacketLossMetric`.

Tabela 4.25: Conceito QualitativeMetricValue

Nome da classe	QualitativeMetricValue
Condição necessária e suficiente (intersecção)	
{maxMetricValue {minMetricValue	
Condição necessária (intersecção)	
Não Existente	
Atributos	
maxMetricValue *Tipo:* float *Cardinalidade:* 1	
minMetricValue *Tipo:* float *Cardinalidade:* 1	
Relações	
Não Existente	
Superclasses	*Não Existente*
Subclasses	*Não Existente*
Instâncias	QualitativeBandwidthMetric, QualitativeDelay-Metric, QualitativeJitterMetric, QualitativePac-ketLossMetric

4.2.2.9 Monitor

Classe com a função de armazenamento de informação relativa à monitorização da rede. Existem dois tipos de monitorização: monitorização entre duas interfaces de rede e monitorização sobre a utilização de um serviço resultante da aplicação de um SLS.

4.2.2.10 MonitorSLS

Monitorização da actividade ligada a um SLS. Útil para sistemas de contabilização e para a verificação de conformidade do serviço prestado com o SLS definido. As métricas monitorizadas em não conformidade com a especificação de serviço são identificadas através da relação `includesNonConformantMetric`.

Tabela 4.26: Conceito MonitorSLS

Nome da classe	MonitorSLS
Condição necessária e suficiente (intersecção)	
Monitor monitorizesSLS	
includesNonConformantMetric	
Condição necessária (intersecção)	
Não Existente	
Atributos	
dateTime *Tipo:* dateTime *Cardinalidade:* 1	
Relações	
includesPerformanceGuarantees *Contradomínio:* Metric *Cardinalidade:* 1 ... *	
monitoresSLS *Contradomínio:* SLS *Cardinalidade:* 1	
includesNonConformantMetric *Contradomínio:* Metric *Cardinalidade:* 0 ... *	

Continua na página seguinte

Tabela 4.26 – continuação da página anterior

Superclasses	Monitor
Subclasses	*Não Existente*
Instâncias	*Não Existente*

4.2.2.11 MonitorScope

Monitorização de desempenho entre duas interfaces de rede. Este tipo monitorização serve para a verificação do nível de desempenho momentâneo entre dois pontos da rede e para inferir sobre a capacidade da rede de prestar determinado nível de serviço dentro de uma esfera de acção. É necessária a especificação de uma interface de entrada e uma interface de saída.

Tabela 4.27: Conceito MonitorScope

Nome da classe	MonitorScope
Condição necessária e suficiente (intersecção)	
monitorizesIngressInterface monitorizesEgressInterface **Condição necessária (intersecção)**	
Não Existente	
Atributos	
dateTime *Tipo:* dateTime *Cardinalidade:* 1	
Relações	
includesPerformanceGuarantees *Contradomínio:* Metric *Cardinalidade:* 1 ... *	
	Continua na página seguinte

Tabela 4.27 – continuação da página anterior

monitorizesIngressInterface *Contradomínio:* Interface *Cardinalidade:* 1	
monitorizesEgressInterface *Contradomínio:* Interface *Cardinalidade:* 1	
Superclasses	Monitor
Subclasses	*Não Existente*
Instâncias	*Não Existente*

4.2.2.12 Event

Armazenamento de informação sobre qualquer tipo de eventos registados na rede. Contém um identificador de evento, uma descrição textual, e data/hora do evento.

Tabela 4.28: Conceito Event

Nome da classe	Event
Condição necessária e suficiente (intersecção)	
dateTime description **Condição necessária (intersecção)**	
Não Existente	
Atributos	
id *Tipo:* string *Cardinalidade:* 1	
	Continua na página seguinte

Tabela 4.28 – continuação da página anterior

dateTime *Tipo:* dateTime *Cardinalidade:* 1	
description *Tipo:* string *Cardinalidade:* 1	
Relações	
Não Existente	
Superclasses	*Não Existente*
Subclasses	*Não Existente*
Instâncias	*Não Existente*

4.2.2.13 ServicePack

Conceito que cobre a definição de um pacote de serviços. Um pacote contém um ou mais serviços.

Tabela 4.29: Conceito ServicePack

Nome da classe	ServicePack
Condição necessária e suficiente (intersecção)	
includesServices	
Condição necessária (intersecção)	
name	
Atributos	
name *Tipo:* string	
Continua na página seguinte	

Tabela 4.29 – continuação da página anterior

Cardinalidade: 1	
Relações	
includesServices *Contradomínio:* Services *Cardinalidade:* 1 . . . *	
Superclasses	*Não Existente*
Subclasses	*Não Existente*
Instâncias	*Não Existente*

4.2.2.14 Service

Definição de um tipo de serviço de rede. Serve para especificar que tipos de serviço vão ser implementados pelo fornecedor. O tipo de serviço é caracterizado por valores qualitativos atribuídos às métricas de desempenho. Através das suas características, um tipo de serviço é então relacionado com vários SLSs.

Tabela 4.30: Conceito Service

Nome da classe	Service
Condição necessária e suficiente (intersecção)	
serviceName includesQualValue definesSLS	
Condição necessária (intersecção)	
Não Existente	
Atributos	
serviceName	
Continua na página seguinte	

Tabela 4.30 – continuação da página anterior

Tipo: string *Cardinalidade:* 1	
Relações	
includesQualValue *Contradomínio:* QualitativeMetricValue *Cardinalidade:* 0 ... *	
definesSLS *Contradomínio:* SLS *Cardinalidade:* 0 ... *	
Superclasses	*Não Existente*
Subclasses	*Não Existente*
Instâncias	*Não Existente*

4.2.3 Descrição de regras

4.2.3.1 Regra 1 - Conformidade da Interface

Regra de verificação de conformidade das interfaces de rede no âmbito do contracto em relação do limite mínimo de largura de banda garantido. Se a capacidade de largura de banda total de qualquer interface de rede dentro do âmbito de um contracto de serviço for menor que a largura de banda mínima contratada, então

essa interface é marcada como em não conformidade com o contracto.

$$Interface(?interface) \wedge Bandwidth(?bandwidth) \wedge SLS(?sls)$$
$$\wedge includesBandwidth(?sls, ?bandwidth) \wedge includesScope(?sls, ?interface)$$
$$\wedge includesTotalBandwidthCapacity(?interface, ?totalBandwidth)$$
$$\wedge metricValue(?bandwidth, ?bandwidthMetricValue)$$
$$\wedge greaterThanOrEqual(?bandwidthMetricValue, ?totalBandwidth)$$
$$\rightarrow includesNonConformantInterface(?sls, ?interface)$$

4.2.3.2 Regra 2 - Cumprimento do SLS

Regra de verificação cumprimento de um contrato na sua monitorização. Cada uma das métricas medidas é avaliada numa regra diferente, devido aos diversos tipos de avaliação requeridos dependendo do tipo de métrica. Se a métrica monitorizada estiver fora do limite (maior ou menor)contratado, então essa métrica é marcada como em não conformidade.

Regra 2 aplicada à métrica de largura de banda:

$$MonitorSLS(?monitorSLS) \wedge SLS(?sls)$$
$$\wedge includesBandwidth(?monitorSLS, ?monitorBandwidth)$$
$$\wedge includesBandwidth(?sls, ?slsBandwidth)$$
$$\wedge metricValue(?monitorBandwidth, ?monitorBandwidthValue)$$
$$\wedge metricValue(?slsBandwidth, ?slsBandwidthValue)$$
$$\wedge unitConversion(?monitorBandwidth, ?monitorConversion)$$
$$\wedge unitConversion(?slsBandwidth, ?slsConversion)$$
$$\wedge lessThan(?realSLSValue, ?realMonitorValue)$$
$$\wedge multiply(?realMonitorValue, ?monitorBandwidthValue, ?monitorConversion)$$
$$\wedge multiply(?realSLSValue, ?slsBandwidthValue, ?slsConversion)$$
$$\rightarrow includesNonConformantMetric(?monitorSLS, ?monitorBandwidth)$$

Regra 2 aplicada à métrica de atraso:

$$MonitorSLS(?monitorSLS) \wedge SLS(?sls)$$
$$\wedge\, includesDelay(?monitorSLS, ?monitorDelay)$$
$$\wedge\, includesDelay(?sls, ?slsDelay)$$
$$\wedge\, metricValue(?monitorDelay, ?monitorDelayValue)$$
$$\wedge\, metricValue(?slsDelay, ?slsDelayValue)$$
$$\wedge\, unitConversion(?monitorDelay, ?monitorConversion)$$
$$\wedge\, unitConversion(?slsDelay, ?slsConversion)$$
$$\wedge\, greaterThan(?realSLSValue, ?realMonitorValue)$$
$$\wedge\, multiply(?realMonitorValue, ?monitorDelayValue, ?monitorConversion)$$
$$\wedge\, multiply(?realSLSValue, ?slsDelayValue, ?slsConversion)$$
$$\rightarrow includesNonConformantMetric(?monitorSLS, ?monitorDelay)$$

Regra 2 aplicada à métrica de variação de atraso:

$$MonitorSLS(?monitorSLS) \wedge SLS(?sls)$$
$$\wedge\, includesJitter(?monitorSLS, ?monitorJitter)$$
$$\wedge\, includesJitter(?sls, ?slsJitter)$$
$$\wedge\, metricValue(?monitorJitter, ?monitorJitterValue)$$
$$\wedge\, metricValue(?slsJitter, ?slsJitterValue)$$
$$\wedge\, unitConversion(?monitorJitter, ?monitorConversion)$$
$$\wedge\, unitConversion(?slsJitter, ?slsConversion)$$
$$\wedge\, greaterThan(?realSLSValue, ?realMonitorValue)$$
$$\wedge\, multiply(?realMonitorValue, ?monitorJitterValue, ?monitorConversion)$$
$$\wedge\, multiply(?realSLSValue, ?slsJitterValue, ?slsConversion)$$
$$\rightarrow includesNonConformantMetric(?monitorSLS, ?monitorJitter)$$

Regra 2 aplicada à métrica de perda de pacotes:

$$MonitorSLS(?monitorSLS) \wedge SLS(?sls)$$
$$\wedge includesPacketLoss(?monitorSLS, ?monitorLoss)$$
$$\wedge includesPacketLoss(?sls, ?slsLoss)$$
$$\wedge metricValue(?monitorLoss, ?monitorLossValue)$$
$$\wedge metricValue(?slsLoss, ?slsLossValue)$$
$$\wedge unitConversion(?monitorLoss, ?monitorConversion)$$
$$\wedge unitConversion(?slsLoss, ?slsConversion)$$
$$\wedge greaterThan(?realSLSValue, ?realMonitorValue)$$
$$\wedge multiply(?realMonitorValue, ?monitorLossValue, ?monitorConversion)$$
$$\wedge multiply(?realSLSValue, ?slsLossValue, ?slsConversion)$$
$$\rightarrow includesNonConformantMetric(?monitorSLS, ?monitorLoss)$$

4.2.3.3 Regra 3 - Actividade da Interface

Regra para obrigar à operacionalidade dos interfaces no âmbito de um contracto. Se qualquer interface de rede estiver dentro do âmbito de um contrato de serviço, então esse interface está activo.

$$Interface(?interface) \wedge SLS(?sls) \wedge includesScope(?sls, ?interface)$$
$$\rightarrow isActive(?interface, true)$$

4.2.3.4 Regra 4 - Atribuição de valor qualitativo

Regras para atribuição de valores qualitativos para as métricas de desempenho de um serviço de rede. Existe uma regra para cada tipo de métrica, devido às diferentes formas avaliação. Se o valor calculado a partir da métrica estiver entre o limite mínimo (e inclusivé, se métrica significar um limite mínimo) e o limite máximo (e inclusivé, se métrica significar um limite máximo) impostos pelo valor qualitativo, então a métrica contém esse valor qualitativo.

Regra 4 aplicada à métrica de largura de banda:

$$Bandwidth(?bandwidth) \wedge QualitativeBandwidthMetric(?qualitativeMetric)$$
$$\wedge maxMetricLimit(?qualitativeMetric, ?maxLimit) \wedge metricValue(?bandwidth, ?metricValue)$$
$$\wedge minMetricLimit(?qualitativeMetric, ?minLimit) \wedge unitConversion(?bandwidth, ?conversion)$$
$$\wedge greaterThanOrEqual(?realValue, ?minLimit) \wedge lessThan(?realValue, ?maxLimit)$$
$$\wedge multiply(?realValue, ?metricValue, ?conversion)$$
$$\rightarrow includesQualitativeValue(?bandwidth, ?qualitativeMetric)$$

Regra 4 aplicada à métrica de atraso:

$$Delay(?delay) \wedge QualitativeDelayMetric(?qualitativeMetric)$$
$$\wedge maxMetricLimit(?qualitativeMetric, ?maxLimit) \wedge metricValue(?delay, ?metricValue)$$
$$\wedge minMetricLimit(?qualitativeMetric, ?minLimit) \wedge unitConversion(?delay, ?conversion)$$
$$\wedge greaterThan(?realValue, ?minLimit) \wedge lessThanOrEqual(?realValue, ?maxLimit)$$
$$\wedge multiply(?realValue, ?metricValue, ?conversion)$$
$$\rightarrow includesQualitativeValue(?delay, ?qualitativeMetric)$$

Regra 4 aplicada à métrica de variação de atraso:

$$Jitter(?jitter) \wedge QualitativeJitterMetric(?qualitativeMetric)$$
$$\wedge maxMetricLimit(?qualitativeMetric, ?maxLimit) \wedge metricValue(?jitter, ?metricValue)$$
$$\wedge minMetricLimit(?qualitativeMetric, ?minLimit)$$
$$\wedge unitConversion(?jitter, ?conversion) \wedge greaterThan(?realValue, ?minLimit)$$
$$\wedge lessThanOrEqual(?realValue, ?maxLimit)$$
$$\wedge multiply(?realValue, ?metricValue, ?conversion)$$
$$\rightarrow includesQualitativeValue(?jitter, ?qualitativeMetric)$$

Regra 4 aplicada à métrica de perda de pacotes:

$$PacketLoss(?loss) \wedge QualitativePacketLossMetric(?qualitativeMetric)$$
$$\wedge maxMetricLimit(?qualitativeMetric, ?maxLimit) \wedge metricValue(?loss, ?metricValue)$$
$$\wedge minMetricLimit(?qualitativeMetric, ?minLimit)$$
$$\wedge unitConversion(?loss, ?conversion) \wedge greaterThan(?metricValue, ?minLimit)$$
$$\wedge lessThanOrEqual(?metricValue, ?maxLimit)$$
$$\wedge multiply(?realValue, ?metricValue, ?conversion)$$
$$\rightarrow includesQualitativeValue(?loss, ?qualitativeMetric)$$

4.2.3.5 Regra 5 - Identificação do Tipo de Serviço

Regra para a identificação do tipo de serviço fornecido através das métricas definidas no SLS. Se as métricas definidas no SLS contêm valores qualitativos equivalentes aos definidos no tipo de serviço, então o SLS é do tipo desse serviço.

$$SLS(?sls) \wedge Service(?service) \wedge includesBandwidth(?sls, ?bandwidth)$$
$$\wedge includesBandwidthQualValue(?service, ?qualiBandwidth)$$
$$\wedge includesDelay(?sls, ?delay) \wedge includesDelayQualValue(?service, ?qualiDelay)$$
$$\wedge includesJitter(?sls, ?jitter) \wedge includesJitterQualValue(?service, ?qualiJitter)$$
$$\wedge includesLossQualValue(?service, ?qualiLoss) \wedge includesPacketLoss(?sls, ?loss)$$
$$\wedge includesQualitativeValue(?bandwidth, ?qualiBandwidth)$$
$$\wedge includesQualitativeValue(?delay, ?qualiDelay)$$
$$\wedge includesQualitativeValue(?jitter, ?qualiJitter)$$
$$\wedge includesQualitativeValue(?loss, ?qualiLoss) \rightarrow definesSLS(?service, ?sls)$$

4.3 Sumário do Capítulo

Após uma breve descrição de tecnologias e linguagens usadas, este capítulo mostra-nos uma descrição detalhada de cada classe contida nos dois módulos que compõem o modelo semântico desenvolvido. Realizou-se também uma breve introdução a cada uma das tecnologias e métodos considerados no desenvolvimento da ontologia. O próximo capítulo trata da integração em ambiente aplicacional do modelo aqui descrito.

Capítulo 5

QoSModel API

5.1 Tecnologias utilizadas no desenvolvimento da API

A API desenvolvida resulta da conjugação de diferentes tecnologias, cada uma com uma funcionalidade própria. Para facilitar a compreensão da estrutura da API, primeiro são apresentadas todas as tecnologias envolvidas no desenvolvimento da QoSModel API. A Figura 5.1 mostra como as diferentes tecnologias são conjugadas.

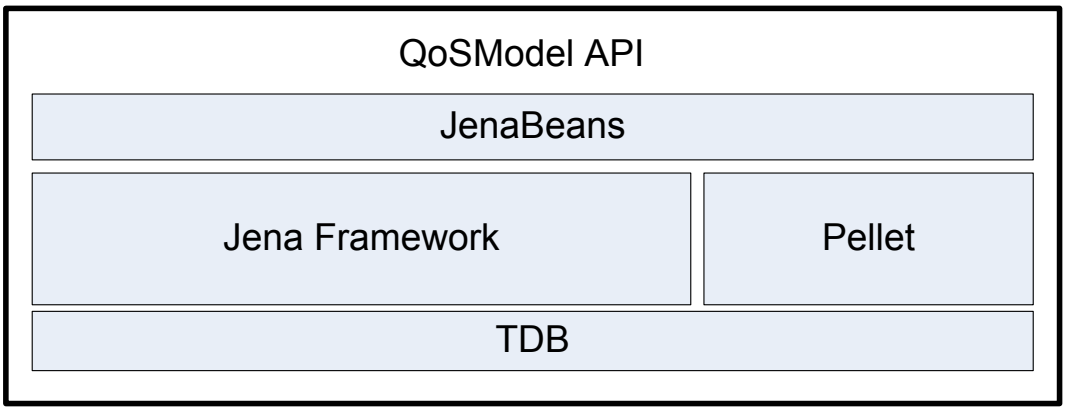

Figura 5.1: Diagrama da QoSModelAPI.

5.1.1 Jena

A Jena consiste numa *framework* Java que permite trabalhar em ambiente de programação com documentos baseados na linguagem RDF. Incluido na *framework* Jena, a Jena Ontology API concentra-se na manipulação de ontologias. Permite a criação de classes, indivíduos, definição de relações e de propriedades. As linguagens ontológicas suportadas vão desde OWL a RDFS, passando pelo DAML+OIL.

A Jena inclui uma série de motores de inferência embutidos, possibilitando também suporte para a integração de outros inferidores disponíveis como por exemplo o Pellet. A funcionalidade de um motor de inferência consiste na introdução de novas relações no grafo de triplos, através de deduções retiradas da ontologia e base conhecimento.

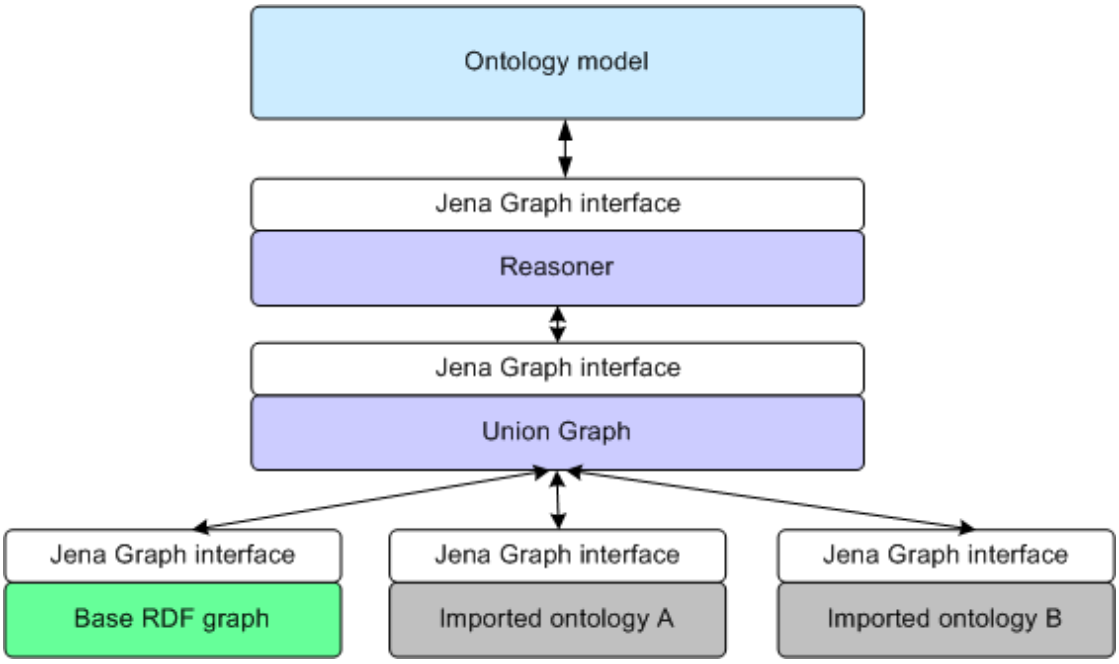

Figura 5.2: Diagrama estrutural da Jena API.

Com a Jena Ontology API trabalha-se o conceito de modelo ontológico. Esse modelo resulta da combinação de um grafo de triplos RDF, que é programado ou descarregado de um documento, com um conjunto de triplos RDF deduzidos pelo motor de inferência. A *framework* também facilita o desenvolvimento modular de ontologias com a gestão automática das importações necessárias ao modelo. Cada grafo resultante de uma importação é unido com o grafo base formando um grafo unificado (ver Figura 5.2).

5.1.2 Persistência de informação

Para a persistência de modelos ontológicos foram consideradas duas soluções: TDB e SDB. A TDB é um sistema nativo ao Jena para garantir a persistência de triplos RDF com metodologias de indexação e armazenamento próprios. O SDB é uma *framework* que permite a persistência de triplos RDF, utilizando bases de dados transaccionais SQL.

A escolha recaiu na solução TDB pela facilidade de implementação e por apresentar melhores índices de desempenho comparativamente à solução SDB. A recorrência à utilização de base de dados transaccionais ainda levanta alguns problemas ao nível de desempenho, em relação à garantia de persistência de modelos ontológicos.

5.1.3 JenaBeans

A API JenaBeans visa preencher o vazio que existe na criação de instâncias de classes Java e classes RDFS. A API segue a convenção do JavaBeans, mapeando as designações definidas nas classes e propriedades Java com classes e propriedades RDFS. O mapeamento entre classes também pode ser realizado através de anotações. A integração de uma API com a JenaBeans confere à Jena uma perspectiva de manipulação de documentos RDF, em conformidade com o paradigma da programação por objectos.

5.1.4 SPARQL

SPARQL é uma linguagem de inquirição e manipulação de dados armazenados em formato RDF. Uma inquirição é composta por duas partes: a cláusula `SELECT` onde se definem as variáveis a inquirir e a cláusula `WHERE` onde é definida uma série de padrões triplos que compõem o chamado grafo padrão que poderá (ou não) corresponder a um sub-grafo de dados RDF. Os padrões triplos obedecem ao mesmo formato que os triplos RDF (compostos por sujeito, predicado e objecto), com a diferença que cada um dos componentes pode ser uma variável. Estão também disponíveis uma série de funções para a disjunção de padrões triplos (`UNION`), definição de padrões triplos opcionais (`OPTIONAL`), ordenação (`ORDER BY`) e filtragem (`FILTER`) de informação. Na API encontram-se algumas *queries* SPARQL predefinidas, existindo também a possibilidade da construção de *queries* personalizadas. A seguir é mostrado um exemplo de uma *query* SPARQL presente na API com o objectivo de devolver a identificação e o nome de todos nós da rede e as suas interfaces.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
```

```
PREFIX net: <http://www.semanticweb.org/ontologies/2009/11/OntologyQoSNetwork.owl#>
PREFIX sls: <http://www.semanticweb.org/ontologies/2009/11/OntologySLS.owl#>
PREFIX serv: <http://www.semanticweb.org/ontologies/2009/11/OntologyNetworkServices.owl#>
SELECT ?nodeID ?name ?interfaceID
WHERE { ?node a net:Node .
        ?node net:id ?nodeID .
        ?node net:name ?name .
        OPTIONAL{?node net:includesInterfaces ?interface.
                 ?interface net:id ?interfaceID}}
```

A *query* devolve a seguinte informação.

```
-----------------------------------
| nodeID |   name    | interfaceID |
===================================
|  "N1"  |  "Node1"  |   "N1I2"    |
|  "N1"  |  "Node1"  |   "N1I1"    |
|  "N2"  |  "Node2"  |   "N2I1"    |
|  "N3"  |  "Node3"  |   "N3I1"    |
-----------------------------------
```

5.1.5 RDFa

O RDFa é uma proposta do W3C de um conjunto de atributos que estendem o XHTML, de forma a possibilitar a inclusão de metadados RDF em paginas Web. A informação incluída na pagina Web poderá ser então extraída para reconstrução do grafo RDF. Desta forma, é possível embeber informação legível por humanos com informação que possa ser interpretada e manipulada por software. A API permite a construção de alguns documentos XHTML+RDFa que servem apenas como exemplos. Alguns dos atributos RDFa utilizados são:

- @about - especifica o recurso sujeito da informação.
- @property - especifica uma propriedade.
- @rel - especifica uma relação.
- @resource - especifica o objecto de uma relação.

No apêndice B é mostrado um exemplo de uma descrição de um SLS, utilizando RDFa.

5.2 QoSModel API

A QoSModel API foi desenvolvida para permitir a utilização aplicacional do modelo QoS desenvolvido no âmbito desta dissertação. A API, desenvolvida em JAVA, permite:

- Criação de instâncias num modelo base independente do modelo QoS.
- Validação da informação segundo as regras impostas pelo modelo QoS.
- Inquirição sobre informação presente no modelo.
- Persistência da informação presente no modelo.

O modelo e API desenvolvidos têm com objectivo de servir de suporte a uma aplicação: (i) que promova interoperabilidade entre cliente e ISP, (ii) para a gestão de contratos de serviço, (iii) acesso a informação sobre SLA/SLSs de uma forma individual ou agregada, (iv) monitorização e auditoria de serviços de rede.

A API segue o paradigma de separação do componente composto por todos os axiomas que definem as classes e propriedades (**T-Box**) e do componente composto por todos os axiomas que compõem as instancias (**A-Box**). Um modelo base é utilizado com o objectivo de albergar todas as instanciações que vão sendo criadas. Todas as operações disponibilizadas pela API são realizadas sobre o modelo base. O modelo QoS, que é importado pelo modelo base, fornece todas as regras (a componente T-Box) que serão utilizadas pelo motor de inferência para a classificação e validação de toda informação injectada no modelo. Este é criado e manipulado recorrendo ao *framework* Jena. Apesar de várias opções de motores de inferência integradas no Jena escolheu-se o Pellet, que oferece outras garantias como suporte à leitura de regras SWRL. Os modelos importados pela ontologia são sempre localizados através de URIs e podem-se situar local (funcionando como cache) ou remotamente (uma das grandes vantagens da Web Semântica). A persistência do modelo é garantida através da *framework* TDB.

5.2.1 Estrutura da API

A API está dividida em três componentes essenciais:

5.2.1.1 QoSModel

A componente `QoSModel` é a componente central da API. É responsável pela manutenção da ontologia e base de conhecimento. Essa manutenção envolve: gestão da estrutura modelar que compõe o modelo; o carregamento das componentes do modelo na memória volátil; inserção, remoção e validação de informação contida na base de conhecimento; sincronização com a informação contida em memória persistente.

O carregamento em memória da ontologia e base de conhecimento é gerido através da API Jena. A ontologia é carregada de um arquivo OWL. A persistência

da base de conhecimento é mantida pela componente TDB. A localização dos ficheiros OWL com o conteúdo de cada módulo da ontologia e a versão persistente da base de conhecimento é especificada pelo ficheiro de configuração da API.

A inserção e remoção de dados na base de conhecimento (ou seja, criação e remoção de instâncias das classes pertencentes à ontologia) é realizada com o suporte da *framework* JenaBeans. A inclusão do JenaBeans permite o tratamento de indivíduos OWL como se correspondessem a instanciações de classe JAVA, facilitando o manuseamento (introdução/consulta de informação) da base de conhecimento em ambiente de programação. A inserção e remoção de dados providenciada por métodos desta componente, segue todas regras especificadas na ontologia, complementando ainda com outras que não puderam fazer parte da mesma ontologia. Por exemplo, na introdução de um SLS são respeitadas todas as relações e atributos para que possa ser classificado como tal. Adicionalmente, em cada interface de rede no âmbito do contrato são adicionadas as políticas de QoS correspondentes e a sua largura de banda reservada é actualizada.

A unicidade de cada indivíduo é gerida pela *framework* JenaBeans. Quando requerido, o campo identificador de cada individuo tem um papel importante nessa garantia de unicidade. A responsabilidade da criação coerente de identificadores para cada indivíduo é delegada para a aplicação.

Também é delegada à aplicação, a invocação da validação da informação contida na base de conhecimento e sincronização desta com o suporte de memória persistente. A validação da informação, por parte do motor de inferência, é um processo pesado computacionalmente, portanto a sua delegação para a aplicação permite que esta a utilize de forma responsável e optimizada. Contudo, a validação é activada automaticamente na consulta de informação, se esta não tiver sido validada desde da última alteração.

5.2.1.2 QoSModelQuery

Componente que permite a inquirição de informação mantida na base de conhecimento. A consulta é realizada através de *queries* SPARQL, aproveitando a componente ARQL incluída na *framework* Jena. A componente `QoSModelQuery` inclui um série de inquirições pré-definidas, mas também permite a consulta personalizada de informação, providenciando as ferramentas para a realizar.

5.2.1.3 QosModelRDFa

Fornece suporte à inclusão de atributos RDFa em documentos XHTML, para representação de informação em formato Web. Esta componente permite incluir em tags HTML, atributos RDFa referentes a:

- Indivíduo representado
- Classe do indivíduo representado.
- Atributos e o seu valor.
- Relações e o seu contradomínio.

5.3 Sumário de Capítulo

Neste capítulo apresentou-se uma API que permite interligar o modelo semântico desenvolvido com uma eventual solução aplicacional sobre gestão de redes multi-serviço. Também são referenciadas todas as tecnologias conjugadas nesta API que possibilitam o manuseamento da base de conhecimento a vários níveis.

Capítulo 6

Conclusão

6.1 Principais Contribuições

O modelo de desenvolvido nesta dissertação consiste em numa abordagem inicial à construção de uma ontologia que engloba o domínio de serviços em redes multiserviço. Esta fase inicial centra-se principalmente na modelação da componente inserida nos contractos de serviço referente à especificação do nível de serviço prestado e dos mecanismos para a garantia de QoS correspondentes à classificação e condicionamento de tráfego.

Apesar de todas as virtudes, anteriormente mencionadas, de linguagens associadas à Web Semântica para expressão de ontologias, foram encontradas limitações a nível de desempenho na sua utilização no armazenamento de informação. Essas limitações devem-se ao processo de validação da base de conhecimento, que ainda não é realizado em tempo de computação satisfatório, principalmente em ontologias de grande dimensão.

No modelo actual existem ainda algumas limitações na definição de regras de inferência para a automatização de processos de gestão e implementação de serviços na rede. O ideal seria delegar estes processos para o modelo semântico, mas com as limitações encontradas no SWRL (referidas na Secção 4.1.2), algumas tiveram que ser programadas na API (por ex: instanciação de politicas QoS, definição de valores qualitativos para as métricas, etc.). Algumas das extensões do SWRL resolvem este problema, mas para isso é necessário a implementação de uma solução mais complexa com vários motores de inferência a funcionar concorrentemente.

O trabalho desenvolvido nesta dissertação serviu de assunto a um artigo publicado na conferência CRC2010 (Carlos Rodrigues, 2010).

6.2 Trabalho Futuro

A ontologia foi desenvolvida de forma a facilitar a sua extensibilidade e reutilização. Em termos futuros poder-se-ia expandir o modelo a novas áreas no domínio de serviços de rede. Os pontos de extensão do modelo seriam: i)na área do controlo de admissão; ii) inclusão de nos e definição de caminhos interiores ao domínio de rede; iii) modelação de mecanismos de escalonamento de tráfego; iv) modelação do processo de monitorização da rede e de serviços; v) melhoramento do suporte ao paradigma de serviços integrados; vi) implementação de uma solução híbrida (com vários motores de inferência) e optimização da QoSModel API.

Bibliografia

Alípio, P., Neves, J., e Carvalho, P. (2006). An ontology for network services. Em *International Conference on Computational Science (3)*, pgs. 240–243.

Almes, G., Kalidindi, S., e Zekauskas, M. (1999a). A one-way delay metric for ippm.

Almes, G., Kalidindi, S., e Zekauskas, M. (1999b). A one-way packet loss metric for ippm.

Almes, G., Kalidindi, S., e Zekauskas, M. (1999c). A round-trip delay metric for ippm.

Babiarz, J., Chan, K., e Baker, F. (2006). Configuration Guidelines for DiffServ Service Classes. RFC 4594 (Informational).

Bless, R., Nichols, K., e Wehrle, K. (2003). A lower effort per-domain behavior (pdb) for differentiated services.

Brickley, D. e Miller, L. (2007). The Friend Of A Friend (FOAF) vocabulary specification. http://xmlns.com/foaf/spec/.

Carlos Rodrigues, Solange Rito Lima, L. M. l.-S. P. C. (2010). Characterization and semantic modeling of services in multiservice networks. Em *CRC 2010*.

Demichelis, C. e Chimento, P. (2002). Ip packet delay variation metric for ip performance metrics (ippm).

Dobson, G., Lock, R., e Sommerville, I. (2005). Qosont: a qos ontology for service-centric systems. Em *EUROMICRO '05: Proceedings of the 31st EUROMICRO Conference on Software Engineering and Advanced Applications*, pgs. 80–87, Washington, DC, USA. IEEE Computer Society.

Dobson, G. e Sanchez-Macian, A. (2006). Towards unified qos/sla ontologies. Em *SCW '06: Proceedings of the IEEE Services Computing Workshops*, pgs. 169–174, Washington, DC, USA. IEEE Computer Society.

Goderis, D. et al. (2002). Service Level Specification Semantics and Parameters. Internet-Draft (work in progress), draft-tequila-sls-02.txt.

Gómez-Pérez, A., Fernández-López, M., e Corcho, O. (2007). *Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web. (Advanced Information and Knowledge Processing)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Goyal, M., Kumar, A., e Sharma, V. (1996). T. toward distributed use of large-scale ontologies. Em *In Gains BR, Musen MA, eds. Proc 10th Banff KA Workshop*, pgs. 605–621.

Green, L. (2006). Service level agreements: an ontological approach. Em *ICEC '06: Proceedings of the 8th international conference on Electronic commerce*, pgs. 185–194, New York, NY, USA. ACM.

Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowl. Acquis.*, 5(2):199–220.

Heinanen, J. e Guerin, R. (1999a). A single rate three color marker.

Heinanen, J. e Guerin, R. (1999b). A two rate three color marker.

Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosof, B., e Dean, M. (2004). Swrl: A semantic web rule language combining owl and ruleml. W3c member submission, World Wide Web Consortium.

Kim, H. M., Sengupta, A., e Evermann, J. (2007). Moq: Web services ontologies for qos and general quality evaluations. *Int. J. Metadata Semant. Ontologies*, 2(3):195–200.

Marchese, M. (2007). *QoS Over Heterogeneous Networks*. Wiley Publishing.

Moraes, P., Sampaio, L., Monteiro, J., e Portnoi, M. (2008). Mononto: A domain ontology for network monitoring and recommendation for advanced internet applications users. Em *Network Operations and Management Symposium Workshops, 2008. NOMS Workshops 2008. IEEE*, pgs. 116–123.

Motik, B., Fokoue, A., Horrocks, I., Wu, Z., Lutz, C., e Grau, B. C. (2009a). OWL 2 web ontology language profiles. W3C recommendation, W3C. http://www.w3.org/TR/2009/REC-owl2-profiles-20091027/.

Motik, B., Parsia, B., e Patel-Schneider, P. F. (2009b). OWL 2 web ontology language structural specification and functional-style syntax. W3C recommendation, W3C. http://www.w3.org/TR/2009/REC-owl2-syntax-20091027/.

Neches, R., Fikes, R., Finin, T., Gruber, T., Patil, R., Senator, T., e Swartout, W. (1991). Enabling Technology for Knowledge Sharing. *AI Magazine*, 12(3):36–56.

Nichols, K., Blake, S., Baker, F., e Black, D. (1998). Definition of the differentiated services field (ds field) in the ipv4 and ipv6 headers.

Nichols, K. e Carpenter, B. (2001). Definition of differentiated services per domain behaviors and rules for their specification.

Noy, N. F. e Mcguinness, D. L. (2001). Ontology development 101: A guide to creating your first ontology. Technical report, Stanford University.

Prudencio, A. C., Willrich, R., Diaz, M., e Tazi, S. (2009). Quality of service specifications: A semantic approach. *Network Computing and Applications, IEEE International Symposium on*, 0:219–226.

Ramakrishnan, K., Floyd, S., e Black, D. (2001). The addition of explicit congestion notification (ecn) to ip.

Royer, J. C., Willrich, R., e Diaz, M. (2008). User profile-based authorization policies for network qos services. *Network Computing and Applications, IEEE International Symposium on*, 0:68–75.

Salsano, S. et al. (2000). Definition and usage of SLSs in the AQUILA consortium. Internet-Draft (work in progress), draft-salsano-aquila-sls-00.txt.

van Harmelen, F. e McGuinness, D. L. (2004). OWL web ontology language overview. W3C recommendation, W3C. http://www.w3.org/TR/2004/REC-owl-features-20040210/.

Wang, Z. (2001). *Internet QoS: Architectures and Mechanisms for Quality of Service*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Zhou, C., Chia, L.-T., e Lee, B.-S. (2004). Daml-qos ontology for web services. *Web Services, IEEE International Conference on*, 0:472.

Zhou, C., Chia, L.-T., e Lee, B.-S. (2005). Qos measurement issues with daml-qos ontology. *E-Business Engineering, IEEE International Conference on*, 0:395–403.

Zwaal, H., H.-M. V. M. (2006). Manipulating context information with swrl. A-MUSE Deliverable D3.12. https://doc.telin.nl/dscgi/ds.py/Get/File-62333/Manipulating%20context%20information%20with%20SWRL.

Apêndice A

Informação de apoio ao planeamento

A.1 Fase 1 - Glossário de termos

A.1.0.4 Contractos de Serviço e Gestão de Rede

Tabela A.1: Glossário de termos no domínio de gestão e contracto de serviços.

Nome	Acrónimos	Descrição	Tipo
Client	-	Contratante do serviço.	Conceito
Service Level Agreement	SLA	Contracto de nível de serviço.	Conceito
Service Level Specification	SLS	Especificações de nível de serviço contratado.	Conceito
Metric	-	Conceito geral de métrica.	Conceito
Delay	-	Métrica de atraso.	Conceito
Jitter	-	Métrica de jitter.	Conceito
Loss	-	Métrica de perda.	Conceito
Bandwidth	-	Métrica de largura de banda.	Conceito
Reliability	-	Métrica de fiabilidade.	Conceito
QualitativeMetricValue	-	Valor qualitativo da métrica.	Conceito
Monitor	-	Conceito geral de monitorização.	Conceito
Monitor SLS	-	Monitorização de actividade relativa ao SLS.	Conceito
MonitorScope	-	Monitorização de actividade dentro de um determinado scope.	Conceito
ServiceSchedule	-	Período do contracto de serviço	Conceito
ReservedServiceSchedule	-	Período do contracto de serviço no qual a data de fim do contrato é especificada.	Conceito
StandardServiceSchedule	-	Período do contracto de serviço com terminação explícita por parte do cliente.	Conceito
Service	-	Serviço contratado pelo cliente	Conceito
ServicePack	-	Pacote de serviços	Conceito
Event	-	Descrição de eventos que ocorram na rede.	Conceito

Continua na página seguinte

Tabela A.1 – continuação da página anterior

Nome do Conceito	Acrônimos	Descrição	Tipo
includesSLA	-	SLA pertencentes a um cliente.	Relação
includesSLS	-	SLSs pertencentes a um SLA.	Relação
includesPerformanceGuarantees	-	Métricas de performance definidas no contrato.	Relação
includesDelay	-	Valor limite da métrica atraso definido no contracto.	Relação
includesJitter	-	Valor limite da métrica jitter no contracto.	Relação
includesLoss	-	Valor limite da métrica perda de pacotes no contracto.	Relação
includesBandwidth	-	Valor da largura de banda reservada ao serviço.	Relação
includesReliability	-	Valor fiabilidade do serviço.	Relação
includesScope	-	Nós de entrada e de saída de um contracto.	Relação
includesIngressInterface	-	Nós de entrada de um contracto.	Relação
includesEgressInterface	-	Nós de saída de um contracto	Relação
includesQualitativeValue	-	Valor qualitativo da métrica	Relação
monitorizesSLS	-	Relaciona informação de monitorização com um SLS.	Relação
monitorizesScope	-	Define entre que interfaces de rede são retirada a informação de monitorização.	Relação
monitorizesIngressInterface	-	Define entre que interface de entrada é retirada a informação de monitorização.	Relação
monitorizesEgressInterface	-	Define entre que interface de saída é retirada a informação de monitorização.	Relação
includesSchedule	-	Definição do período de prestação de serviço.	Relação
includesTrafficClassifier	-	Classificação de tráfego definida no SLS	Relação
includesIngressTrafficClassifier	-	Classificação de tráfego definida nos nós de entrada do SLS.	Relação
includesEgressTrafficClassifier	-	Classificação de tráfego definida nos nós de saída do SLS.	Relação
includesTrafficConditioner	-	Condicionadores de tráfego definidos no SLS	Relação
includesIngressTrafficConditioner	-	Condicionadores de tráfego definidos nos nós de entrada do SLS	Relação
includesEgressTrafficConditioner	-	Condicionadores de tráfego definidos nos nós de saída do SLS	Relação
includesQualMetric	-	Especificação de métricas qualitativas no tipo de serviço.	Relação
includesBandwidthQualMetric	-	Especificação da métrica qualitativa de largura de banda no tipo de serviço.	Relação
includesDelayQualMetric	-	Especificação da métrica qualitativa de largura de atraso no tipo de serviço.	Relação
includesJitterQualMetric	-	Especificação da métrica qualitativa da variação do atraso no tipo de serviço.	Relação
includesPacketLossQualMetric	-	Especificação da métrica qualitativa do rácio de perda de pacotes no tipo de serviço.	Relação
includesNonConformantMetric	-	Identifica das métricas em não conformidade com as especificações contratadas.	Relação
includesNonConformantInterface	-	Identifica dos interfaces de rede em não conformidade com as especificações contratadas.	Relação
includesServices	-	Tipo de serviços pertencentes um pacote de serviços.	Relação

Continua na página seguinte

Tabela A.1 – continuação da página anterior

Nome do Conceito	Acrónimos	Descrição	Tipo
definesSLS	-	Relaciona SLSs com um tipo de serviço.	Relação
dateTime	-	Definição da data e hora.	Atributo
description	-	Descrição textual do evento.	Atributo
startDate	-	Definição da data de inicio.	Atributo
endDate	-	Definição da data final.	Atributo
value	-	Valor quantitativo da métrica.	Atributo
unit	-	Unidade da métrica.	Atributo
baseUnit	-	Unidade base da métrica.	Atributo
unitConversion	-	Valor utilizado para conversão da métrica para a unidade base.	Atributo
id	-	Identificador de conceito.	Atributo
name	-	Nome.	Atributo
serviceName	-	Nome do tipo de serviço.	Atributo
maxMetricLimit	-	Limite máximo considerado na métrica qualitativa.	Atributo
minMetricLimit	-	Limite mínimo considerado na métrica qualitativa.	Atributo

A.1.0.5 Rede Multiserviço

Tabela A.2: Glossário de termos no domínio de implementação física de redes.

Nome	Acrónimos	Descrição	Tipo
Action	-	Acção a realizar sobre o pacote.	Conceito
TRANSMIT	-	Transmissão do pacote.	Conceito
MARK	-	Atribuição de uma marca ao pacote.	Conceito
DROP	-	Descarte do pacote.	Conceito
Interface	-	Interface de rede.	Conceito
LogicOperator	-	Operador lógico.	Conceito
Mark	-	Marcas singular atribuída ao pacote.	Conceito
DSCP	-	Campo para marcação DiffServ usado no cabeçalho IPv4 e IPv6.	Conceito
FlowLabel	-	Campo para marcação de fluxo usado no cabeçalho IPv6.	Conceito
Network	-	Rede multiserviço.	Conceito
Node	-	Nó de rede.	Conceito
Policy	-	Politica QoS.	Conceito
TrafficClassifier	-	Classificador de tráfego.	Conceito
BehaviorAggregate	BA	Classificador do tipo BA.	Conceito
Multifield	MF	Classificador do tipo MF.	Conceito
TrafficConditioner	-	Condicionador de tráfego.	Conceito
Marker	-	Marcador de tráfego.	Conceito
Policer	-	Condicionador do tipo policer.	Conceito
TokenBucketPolicer	-	Policer com dois níveis de conformidade.	Conceito
SingleRateThreeColorMarker	srTCM	Policer com três níveis de conformidade.	Conceito
TwoRateThreeColorMarker	trTCM	Policer com três níveis de conformidade e dois perfis de tráfego.	Conceito

Continua na página seguinte

Tabela A.2 – continuação da página anterior

Nome do Conceito	Acrónimos	Descrição	Tipo
Shaper	-	Condicionador do tipo shaper.	Conceito
LeakyBucket	-	Shaper com o algoritmo leaky bucket.	Conceito
TokenBucketShaper	-	Shaper com o algoritmo token bucket.	Conceito
includesAction	-	Acção do nível de conformidade	Relação
includesClassificationMark	-	Parâmetro usado na classificação agregada de tráfego	Relação
includesClassifier	-	Classificador de tráfego usado	Relação
includesConditioner	-	Condicionador de tráfego usado	Relação
includesConformanceLevel	-	Nível de conformidade do marcador	Relação
includesConformanceLevels	-	Vários níveis de conformidade do policer	Relação
includesInProfileLevel	-	Níveis de conformidade dedicado a tráfego dentro do perfil	Relação
includesExcessProfileLevel	-	Níveis de conformidade dedicado a tráfego em excesso ao perfil	Relação
includesOutProfileLevel	-	Níveis de conformidade dedicado a tráfego fora do perfil	Relação
includesLogicOperator	-	Operador lógico usado na classificação de tráfego	Relação
includesNestedClassifier	-	Classificador definido dentro doutro classificador	Relação
includesNodes	-	Nós pertencentes à rede	Relação
includesPolicies	-	Politicas de classificação e condicionamento de tráfego	Relação
includesIngressPolicies	-	Politicas de classificação e condicionamento de aplicadas na entrada de pacotes	Relação
includesEgressPolicies	-	Politicas de classificação e condicionamento de aplicadas na entrada de pacotes	Relação
includesResultingMark	-	Marca atribuída ao pacote	Relação
includesClassificationFields	-	Campos definidos na classificação de tráfego	Atributo
includesLayer2AddressSpec	-	Pacote classificado segundo o endereço da segunda camada de ligação	Atributo
includesLayer2AddressSourceSpec	-	Pacote classificado segundo o endereço de origem da camada de ligação	Atributo
includesLayer2AddressDestinationSpec	-	Pacote classificado segundo o endereço de destino da camada de ligação	Atributo
includesLayer3AddressSpec	-	Pacote classificado segundo o endereço da segunda camada de rede	Atributo
includesLayer3AddressSourceSpec	-	Pacote classificado segundo o endereço de origem da camada de rede	Atributo
includesLayer3AddressDestinationSpec	-	Pacote classificado segundo o endereço de destino da camada de rede	Atributo
includesLayer4AddressSpec	-	Pacote classificado segundo o endereço da segunda camada de transporte	Atributo
includesLayer4AddressSourceSpec	-	Pacote classificado segundo o endereço de origem da camada de transporte	Atributo
includesLayer4AddressDestinationSpec	-	Pacote classificado segundo o endereço de destino da camada de transporte	Atributo
includesProtocolID	-	Pacote classificado segundo o campo identificador de protocolo	Atributo
includesConditionerParameters	-	Parâmetros definidos no condicionamento de tráfego	Atributo

Continua na página seguinte

Tabela A.2 – continuação da página anterior

Nome do Conceito	Acrónimos	Descrição	Tipo
includesBS	-	Definição do Bucket Size	Atributo
includesCBS	-	Definição do Committed Bucket Size	Atributo
includesCIR	-	Definição do Committed Information Rate	Atributo
includesEBS	-	Definição do Excess Bucket Size	Atributo
includesPBS	-	Definição do Peak Bucket Size	Atributo
includesPIR	-	Definição do Peak Information Rate	Atributo
includesSR	-	Definição do Single Rate	Atributo
includesIngressBandwidthReservedCapacity	-	Definição da largura de entrada banda reservada na interface	Atributo
includesEgressBandwidthReservedCapacity	-	Definição da largura de saída de banda reservada na interface	Atributo
includesTotalBandwidthCapacity	-	Definição da largura de banda total do interface	Atributo
id	-	Identificador de conceito	Atributo
name	-	Nome	Atributo
isActive	-	Indica se a interface de rede está activa	Atributo

A.2 Fase2 - Taxonomia de conceitos

A.2.0.6 Gestão de Serviços

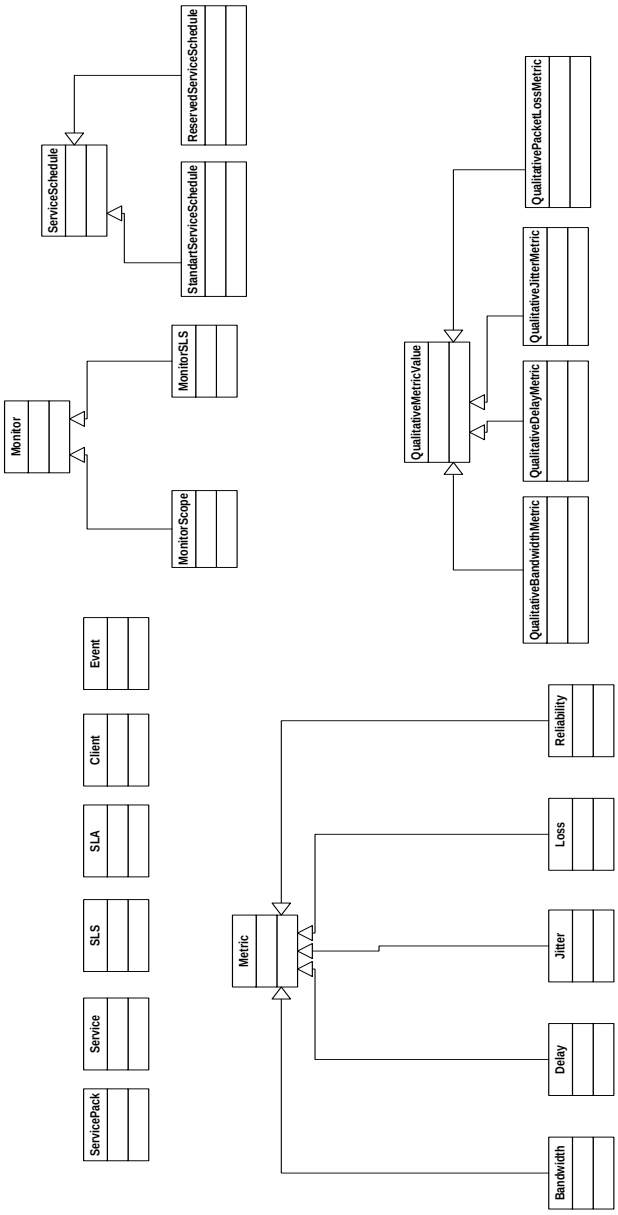

Figura A.1: Taxonomia de conceitos no domínio de gestão e contractos de serviços.

A.2.0.7 Rede Multiserviço

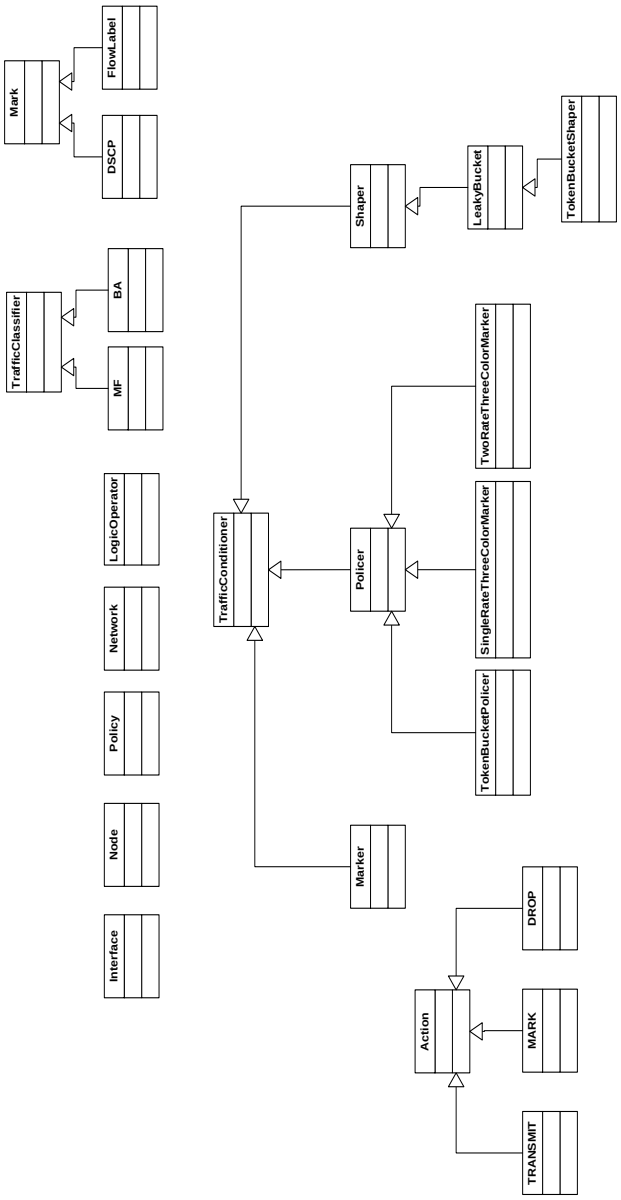

Figura A.2: Taxonomia de conceitos no domínio de implementação física de redes.

A.3 Fase 3 - Relações binárias ad hoc

A.3.0.8 Gestão de Serviços

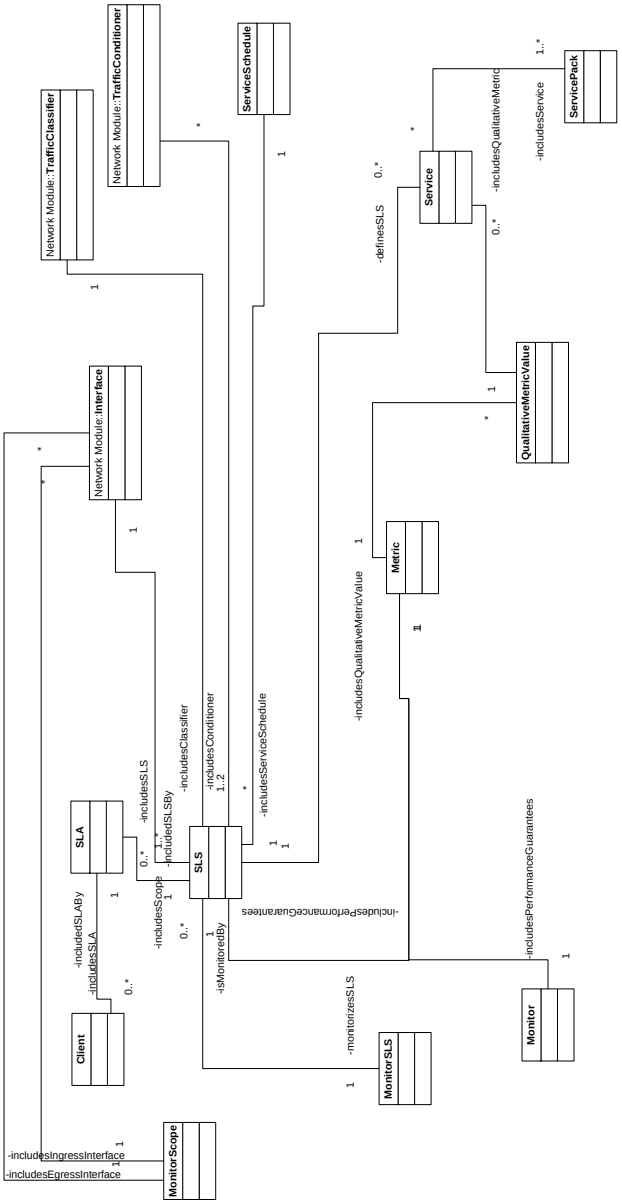

Figura A.3: Relações binarias ad hoc entre conceitos no domínio de gestão e contractos de serviços.

A.3.0.9 Rede Multiserviço

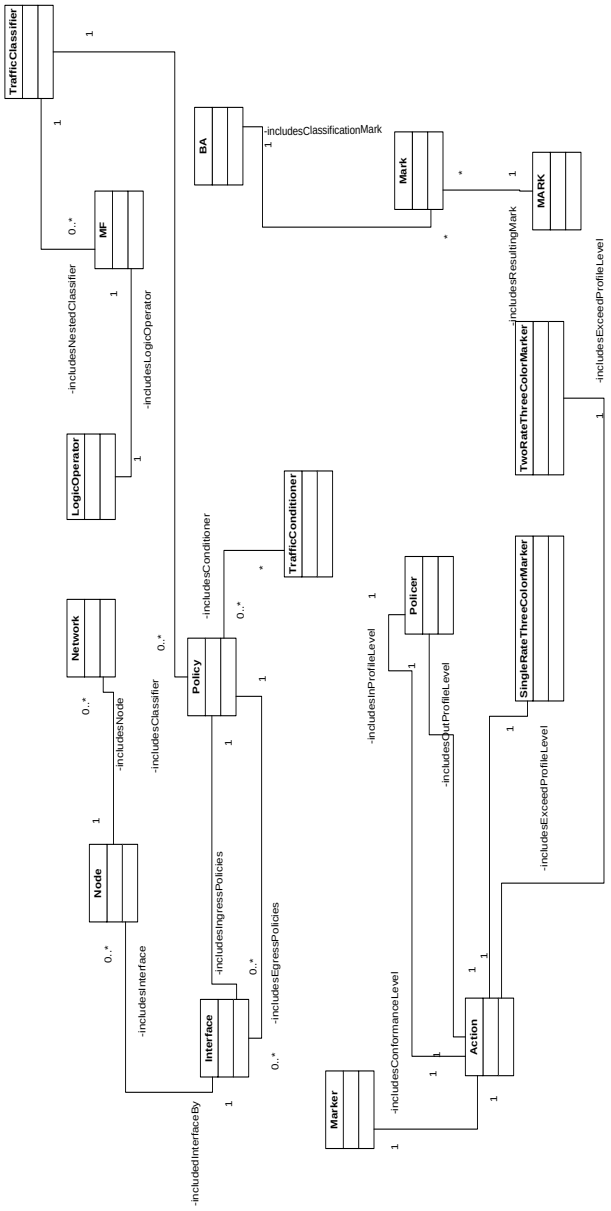

Figura A.4: Relações binárias ad hoc entre conceitos no domínio de implementação física de redes.

A.4 Fase 4 - Dicionário de conceitos

A.4.0.10 Gestão e Contractos de Serviço

Tabela A.3: Dicionário de conceitos no domínio de gestão e contracto de serviço.

Nome do Conceito	Atributos	Relações
Client		includesSLA
SLA	id name	includesSLS
SLS	id	includesScope includesPerformanceGuarantees includesTrafficClassifier includesTrafficConditioner includesReliability includesSchedule isMonitorizedBy includesNonConformantInterface
Monitor	id dateTime	includesPerformanceGuarantees
MonitorSLS	id dateTime	includesPerformanceGuarantees includesNonConformantMetric
MonitorScope	id dateTime	includesPerformanceGuarantees includesIngressInterface includesEgressInterface
ServiceSchedule	startDate	
ReservedServiceSchedule	startDate endDate	
ServicePack		includesServices
Service	serviceName	includesQualMetric
StandardServiceSchedule	startDate	
vent E	id dateTime description	
Metric	unit value baseUnit unitConversion	includesQualitativeValue
Bandwidth	unit value baseUnit unitConversion	includesQualitativeValue
Delay	unit value baseUnit unitConversion	includesQualitativeValue
Jitter	unit value baseUnit unitConversion	includesQualitativeValue
Loss	unit value	includesQualitativeValue

Continua na página seguinte

Tabela A.3 – continuação da página anterior

Nome do Conceito	Atributos	Relações
	baseUnit unitConversion	
Reliability	unit value baseUnit unitConversion	includesQualitativeValue
QualitativeMetricValue	id maxMetricLimit minMetricLimit	
QualitativeBandwidthValue	id maxMetricLimit minMetricLimit	
QualitativeDelayValue	id maxMetricLimit minMetricLimit	
QualitativeJitterValue	id maxMetricLimit minMetricLimit	
QualitativePacketLossValue	id maxMetricLimit minMetricLimit	

A.4.0.11 Rede Multiserviço

Tabela A.4: Dicionário de conceitos no domínio de implementação física de redes.

Nome do Conceito	Atributos	Relações
Action		
MARK		includesResultingMark
DROP		
Interface	id name includesTotalBandwidthCapacity includesIngressBandwidthReservedCapacity includesIngressBandwidthReservedCapacity includesLayer2Address includesLayer3Address isActive	includesIngressPolicies includesEgressPolicies
LogicOperator		
Mark		
DSCP		
FlowLabel		
Network		includesNodes
Node	id name	includesInterfaces
Policy	id	includesClassifier includesConditioner

Continua na página seguinte

Tabela A.4 – continuação da página anterior

Nome do Conceito	Atributos	Relações
TrafficClassifier	id	
BehaviorAggregate	id	includesClassificationMark
Multifield	id includesClassificationFields includesLogicOperator	includesNestedClassifier
TrafficConditioner	id	
Marker	id	includesConformanceLevel
Policer	id	includesInProfileLevel includesOutProfileLevel
TokenBucketPolicer	id includesSR includesBS	includesInProfileLevel includesOutProfileLevel
SingleRateThreeColorMarker	id includesCIR includesCBS includesEBS	includesInProfileLevel includesExcessProfileLevel includesOutProfileLevel
TwoRateThreeColorMarker	id includesCIR includesCBS includesPIR includesPBS	includesInProfileLevel includesExcessProfileLevel includesOutProfileLevel
Shaper	id	
	id includesSR	
LeakyBucket TokenBucketShaper	id includesSR includesBS	

A.5 Fase 5 - Descrição de relações ad-hoc

A.5.0.12 Contractos de Serviço e Gestão de Rede

Tabela A.5: Relações ad-hoc no domínio de gestão e contracto de serviços.

Nome	Domínio	Card.	Contradomínio	Relação inversa
includesPerformanceGuarantees	SLS Scope	4	Metric	includedPerformanceGuaranteeBy
includesBandwidth	SLS Scope	1	Metric	includedPerformanceGuaranteeBy
includesDelay	SLS Scope	1	Metric	includedPerformanceGuaranteeBy
includesJitter	SLS Scope	1	Metric	includedPerformanceGuaranteeBy
includesPacketLoss	SLS Scope	1	Metric	includedPerformanceGuaranteeBy

Continua na página seguinte

Tabela A.5 – continuação da página anterior

Nome	Domínio	Card.	Contradomínio	Relação inversa
includeQualitativeValue	Metric	1	QualitativeMetricValue	
includesReliability	SLS	1	Reliability	
includesSchedule	SLS	1	ServiceSchedule	
includesScope	SLS	N	Interface	includedScopeBy
includesIngressScope	SLS	N	Interface	
includesEgressScope	SLS	N	Interface	
includesSLA	Client	N	SLA	includedSLABy
includesSLS	SLA	N	SLS	includedSLSBy
includesTrafficClassifier	SLS	N	TrafficClassifier	includedTrafficClassifierBy
includesIngressTrafficClassifier	SLS	1	TrafficClassifier	includedIngressTrafficClassifierBy
includesEgressTrafficClassifier	SLS	1	TrafficClassifier	includedEgressTrafficClassifierBy
includesTrafficConditioner	SLS	N	TrafficConditioner	includedTrafficConditionerBy
includesIngressTrafficConditioner	SLS	N	TrafficConditioner	includedIngressTrafficConditionerBy
includesEgressTrafficConditioner	SLS	N	TrafficConditioner	includedEgressTrafficConditionerBy
includesServices	ServicePack	N	Service	
includesQualValue	Service	N	QualitativeMetricValue	
includesBandwidthQualValue	Service	N	QualitativeBandwidthValue	
includesDelayQualValue	Service	N	QualitativeDelayMetric	
includesJitterQualValue	Service	N	QualitativeJitterMetric	
includesPacketLossQualValue	Service	N	QualitativePacketLossMetric	
definesSLS	Service	N	SLS	
includesNonConformantInterface	SLS	N	Interface	
includesNonConformantMetric	SLS	N	Metric	
monitorizesScope	MonitorScope	2	Interface	
monitorizesIngressScope	MonitorScope	1	Interface	
monitorizesEgressScope	MonitorScope	1	Interface	
isMonitorizedBy	SLS	N	MonitorSLS	monitorizesSLS

A.5.0.13 Rede Multiserviço

Tabela A.6: Relações ad-hoc no domínio de redes multiserviço.

Nome	Domínio	Card.	Contradomínio	Relação inversa
includesClassificationMark	BA	1	Mark	
includesClassifier	Policy	1	TrafficClassifier	includedClassifierBy
includesConditioner	Policy	N	TrafficConditioner	includedConditionerBy
includesConformanceLevel	Marker	1	Action	
includesConformanceLevels	Policer	1	Action	
includesInProfileLevel	Policer	1	Action	
includesExcessProfileLevel	SingleRateThreeColorMarker TwoRateThreeColorMarker	1	Action	
includesOutProfileLevel	Policer	1	Action	
includesInterfaces	Node	N	Interface	includedInterfaceBy
includesLogicOperator	MF	1	LogicOperator	
includesNestedClassifier	MF	N	TrafficCalssifier	
includesPolicies	Interface	N	Policiy	

Continua na página seguinte

Tabela A.6 – continuação da página anterior

Nome	Domínio	Card.	Contradomínio	Relação inversa
includesIngressPolicies	Interface	N	Policiy	
includesEgressPolicies	Interface	N	Policiy	
includesResultingMark	MARK	1	Mark	

A.6 Fase 6 - Descrição de atributos

A.6.0.14 Contractos de Serviço e Gestão de Rede

Tabela A.7: Atributos no domínio de redes multiserviço.

Nome	Conceito	Tipo	Unidade de medida	Cardinalidade
id	Thing	String		1
name	Thing	String		1
baseUnit	Metric	String		1
unit	Metric	String		1
unitConversion	Metric	Float		1
value	Metric	Float		1
dateTime	Event Monitor	DateTime		1
description	Event	String		1
startDate	ServiceSchedule	DateTime		1
endDate	ReservedServiceSchedule	DateTime		1
serviceName	Service	String		1
maxMetricLimit	QualitativeMetric	Float		1
minMetricLimit	QualitativeMetric	Float		1

A.6.0.15 Rede Multiserviço

Tabela A.8: Atributos no domínio de gestão e contracto de serviços.

Nome	Conceito	Tipo	Unidade de medida	Cardinalidade
id	Thing	String		1
name	Thing	String		1
includesCalssificationFiels	MF	String		N
includesLayer2AddressSpec	MF	String		N
includesLayer2AddressSourceSpec	MF	String		N
includesLayer2AddressDestinationSpec	MF	String		N
includesLayer3AddressSpec	MF	String		N
includesLayer3AddressSourceSpec	MF	String		N

Continua na página seguinte

Tabela A.8 – continuação da página anterior

Nome	Conceito	Tipo	Unidade de medida	Cardinalidade
includesLayer3AddressDestinationSpec	MF	String		N
includesLayer4AddressSpec	MF	String		N
includesLayer4AddressSourceSpec	MF	String		N
includesLayer4AddressDestinationSpec	MF	String		N
includesProtocolId	MF	String		N
includesConditionerParameters	TrafficConditioner	Float		1
includesSR	TrafficConditioner	Float	bps	1
includesBS	TrafficConditioner	Float	byte	1
includesCIR	TrafficConditioner	Float	bps	1
includesCBS	TrafficConditioner	Float	byte	1
includesEBS	TrafficConditioner	Float	byte	1
includesPIR	TrafficConditioner	Float	bps	1
includesPBS	TrafficConditioner	Float	byte	1
includesIngressReservedCapacity	Interface	Float	bps	1
includesEgressReservedCapacity	Interface	Float	bps	1
includesTotalBandwidthCapacity	Interface	Float	bps	1
includeLayer2Address	Interface	String		1
includeLayer3Address	Interface	String		1
isActive	Interface	Boolean		1

A.7 Fase 7 - Descrição de regras

Tabela A.9: Regra 1 - Conformidade da Interface.

Nome da regra	Conformidade da Interface
Descrição	Regra de verificação de conformidade das interfaces de rede no âmbito do contracto em relação ao limite mínimo de largura de banda garantido.
Expressão	Interface(?interface) , Bandwidth(?bandwidth) , SLS(?sls) , includesBandwidth(?sls, ?bandwidth) , includesScope(?sls, ?interface) , includesTotalBandwidthCapacity(?interface, ?totalBandwidth) , metricValue(?bandwidth, ?bandwidthMetricValue) , greaterThanOrEqual(?bandwidthMetricValue, ?totalBandwidth) -> includesNonConformantInterface(?sls, ?interface)
Conceitos	Interface Bandwidth SLS
Atributos	includesTotalBandwidthCapacity metricValue
Relações binárias	includesBandwidth includesScope includesNonConformantInterface
Buit-In	greaterThanOrEqual
Variáveis	?interface ?bandwidth ?sls ?totalBandwidth ?bandwidthMetricValue

Tabela A.10: Regra 2.a - Validação da Largura de Banda.

Nome da regra	Validação da Largura de Banda
Descrição	Regra de verificação cumplicidade do limite mínimo de largura de banda especificado num contrato com a sua monitorização.
Expressão	MonitorSLS(?monitorSLS), SLS(?sls), includesBandwidth(?monitorSLS, ?monitorBandwidth), includesBandwidth(?sls, ?slsBandwidth), metricValue(?monitorBandwidth, ?monitorBandwidthValue), metricValue(?slsBandwidth, ?slsBandwidthValue), unitConversion(?monitorBandwidth, ?monitorConversion), unitConversion(?slsBandwidth, ?slsConversion), lessThan(?realSLSValue, ?realMonitorValue), multiply(?realMonitorValue, ?monitorBandwidthValue, ?monitorConversion), multiply(?realSLSValue, ?slsBandwidthValue, ?slsConversion) -> includesNonConformantMetric(?monitorSLS, ?monitorBandwidth)
Conceitos	MonitorSLS SLS
Atributos	metricValue unitConversion
Relações binárias	includesBandwidth
Buit-In	lessThan multiply
Variáveis	?monitorSLS ?sls ?monitorBandwidth ?monitorBandwidthValue ?slsBandwidth ?slsBandwidthValue ?monitorConversion ?slsConversion ?realMonitorValue ?realSLSValue

Tabela A.11: Regra 2.b - Validação do Atraso.

Nome da regra	Validação do Atraso
Descrição	Regra de verificação cumplicidade do limite máximo de atraso especificado num contrato com a sua monitorização.
Expressão	MonitorSLS(?monitorSLS), SLS(?sls), includesDelay(?monitorSLS, ?monitorDelay), includesDelay(?sls, ?slsDelay), metricValue(?monitorDelay, ?monitorDelayValue), metricValue(?slsDelay, ?slsDelayValue), unitConversion(?monitorDelay, ?monitorConversion), unitConversion(?slsDelay, ?slsConversion), greaterThan(?realSLSValue, ?realMonitorValue), multiply(?realMonitorValue, ?monitorDelayValue, ?monitorConversion), multiply(?realSLSValue, ?slsDelayValue, ?slsConversion) -> includesNonConformantMetric(?monitorSLS, ?monitorDelay)
Conceitos	MonitorSLS SLS
Atributos	metricValue unitConversion

Continua na página seguinte

Tabela A.11 – continuação da página anterior

Relações binárias	includesDelay
Buit-In	greaterThan multiply

Continua na página seguinte

Tabela A.11 – continuação da página anterior

Variáveis	?monitorSLS ?sls ?monitorDelay ?monitorDelayValue ?slsDelay ?slsDelayValue ?monitorConversion ?slsConversion ?realMonitorValue ?realSLSValue

Tabela A.12: Regra 2.c - Validação do Jitter.

Nome da regra	Validação do Jitter
Descrição	Regra de verificação cumplicidade do limite máximo de variação de atraso especificado num contrato na sua monitorização.
Expressão	MonitorSLS(?monitorSLS), SLS(?sls), includesJitter(?monitorSLS, ?monitorJitter), includesJitter(?sls, ?slsJitter), metricValue(?monitorJitter, ?monitorJitterValue), metricValue(?slsJitter, ?slsJitterValue), unitConversion(?monitorJitter, ?monitorConversion), unitConversion(?slsJitter, ?slsConversion), greaterThan(?realSLSValue, ?realMonitorValue), multiply(?realMonitorValue, ?monitorJitterValue, ?monitorConversion), multiply(?realSLSValue, ?slsJitterValue, ?slsConversion) -> includesNonConformantMetric(?monitorSLS, ?monitorJitter)
Conceitos	MonitorSLS SLS
Atributos	metricValue unitConversion
Relações binárias	includesJitter
Buit-In	greaterThan multiply
Variáveis	?monitorSLS ?sls ?monitorJitter ?monitorJitterValue ?slsJitter ?slsJitterValue ?monitorConversion ?slsConversion ?realMonitorValue ?realSLSValue

Tabela A.13: Regra 2.d - Validação de Perdas.

Nome da regra	Validação de Perdas
Descrição	Regra de verificação cumplicidade do limite máximo de perda de pacotes especificado num contrato na sua monitorização.

Continua na página seguinte

Tabela A.13 – continuação da página anterior

Expressão	MonitorSLS(?monitorSLS), SLS(?sls), includesPacketLoss(?monitorSLS, ?monitorLoss), includesPacketLoss(?sls, ?slsLoss), metricValue(?monitorLoss, ?monitorLossValue), metricValue(?slsLoss, ?slsLossValue), unitConversion(?monitorLoss, ?monitorConversion), unitConversion(?slsLoss, ?slsConversion), greaterThan(?realSLSValue, ?realMonitorValue), multiply(?realMonitorValue, ?monitorLossValue, ?monitorConversion), multiply(?realSLSValue, ?slsLossValue, ?slsConversion) -> includesNonConformantMetric(?monitorSLS, ?monitorLoss)
Conceitos	MonitorSLS SLS
Atributos	metricValue unitConversion
Relações binárias	includesPacketLoss
Buit-In	greaterThan multiply
Variáveis	?monitorSLS ?sls ?monitorLoss ?monitorLossValue ?slsLoss ?slsLossValue ?monitorConversion ?slsConversion ?realMonitorValue ?realSLSValue

Tabela A.14: Regra 3 - Actividade da Interface.

Nome da regra	Actividade da Interface
Descrição	Regra para obrigar â operacionalidade dos interfaces no âmbito de um contracto.
Expressão	Interface(?interface) , SLS(?sls) , includesScope(?sls, ?interface) -> isActive(?interface, true)
Conceitos	Interface SLS
Atributos	isActive
Relações binárias	includesScope
Buit-In	
Variáveis	?interface ?sls

Tabela A.15: Regra 4.a - Largura de Banda Qualitativa.

Nome da regra	Largura de Banda Qualitativa
Descrição	Regras para atribuição de valores qualitativos para a métrica de largura de banda.

Continua na página seguinte

Tabela A.15 – continuação da página anterior

Expressão	Bandwidth(?bandwidth), QualitativeBandwidthMetric(?qualitativeMetric), maxMetricLimit(?qualitativeMetric, ?maxLimit), metricValue(?bandwidth, ?metricValue), minMetricLimit(?qualitativeMetric, ?minLimit), unitConversion(?bandwidth, ?conversion), greaterThanOrEqual(?realValue, ?minLimit), lessThan(?realValue, ?maxLimit), multiply(?realValue, ?metricValue, ?conversion) -> includesQualitativeValue(?bandwidth, ?qualitativeMetric)
Conceitos	Bandwidth QualitativeBandwidthMetric
Atributos	maxMetricLimit minMetricLimit metricValue unitConversion
Relações binárias	includesQualitativeValue
Buit-In	greaterThanOrEqual lessThan multiply
Variáveis	?bandwidth ?qualitativeMetric ?metricValue ?maxLimit ?minLimit ?conversion ?realValue

Tabela A.16: Regra 4.b - Atraso Qualitativo.

Nome da regra	Atraso Qualitativo
Descrição	Regras para atribuição de valores qualitativos para a métrica de atraso.
Expressão	Delay(?delay), QualitativeDelayMetric(?qualitativeMetric), maxMetricLimit(?qualitativeMetric, ?maxLimit), metricValue(?delay, ?metricValue), minMetricLimit(?qualitativeMetric, ?minLimit), unitConversion(?delay, ?conversion), greaterThan(?realValue, ?minLimit), lessThanOrEqual(?realValue, ?maxLimit), multiply(?realValue, ?metricValue, ?conversion) -> includesQualitativeValue(?delay, ?qualitativeMetric)
Conceitos	Delay QualitativeDelayMetric
Atributos	maxMetricLimit minMetricLimit metricValue unitConversion
Relações binárias	includesQualitativeValue
Buit-In	greaterThan lessThanOrEqual multiply
Variáveis	?delay ?qualitativeMetric ?metricValue ?maxLimit ?minLimit ?conversion ?realValue

Tabela A.17: Regra 4.c - Jitter Qualitativo.

Nome da regra	Jitter Qualitativo
Descrição	Regras para atribuição de valores qualitativos para a métrica de variação de atraso.
Expressão	Jitter(?jitter), QualitativeJitterMetric(?qualitativeMetric), maxMetricLimit(?qualitativeMetric, ?maxLimit), metricValue(?jitter, ?metricValue), minMetricLimit(?qualitativeMetric, ?minLimit), unitConversion(?jitter, ?conversion), greaterThan(?realValue, ?minLimit), lessThanOrEqual(?realValue, ?maxLimit), multiply(?realValue, ?metricValue, ?conversion) -> includesQualitativeValue(?jitter, ?qualitativeMetric)
Conceitos	Jitter QualitativeJitterMetric
Atributos	maxMetricLimit minMetricLimit metricValue unitConversion
Relações binárias	includesQualitativeValue
Buit-In	greaterThan lessThanOrEqual multiply
Variáveis	?jitter ?qualitativeMetric ?metricValue ?maxLimit ?minLimit ?conversion ?realValue

Tabela A.18: Regra 4.d - Perda Qualitativa.

Nome da regra	Perda Qualitativa
Descrição	Regras para atribuição de valores qualitativos para a métrica do rácio de perda de pacotes.
Expressão	PacketLoss(?loss), QualitativePacketLossMetric(?qualitativeMetric), maxMetricLimit(?qualitativeMetric, ?maxLimit), metricValue(?loss, ?metricValue), minMetricLimit(?qualitativeMetric, ?minLimit), unitConversion(?loss, ?conversion), greaterThan(?metricValue, ?minLimit), lessThanOrEqual(?metricValue, ?maxLimit), multiply(?realValue, ?metricValue, ?conversion) -> includesQualitativeValue(?loss, ?qualitativeMetric)
Conceitos	PacketLoss QualitativePacketLossMetric
Atributos	maxMetricLimit minMetricLimit metricValue unitConversion
Relações binárias	includesQualitativeValue
Buit-In	greaterThan lessThanOrEqual multiply

Continua na página seguinte

Tabela A.18 – continuação da página anterior

Variáveis	?loss ?qualitativeMetric ?metricValue ?maxLimit ?minLimit ?conversion ?realValue

Tabela A.19: Regra 5 - Identificação do Tipo de Serviço.

Nome da regra	Identificação do Tipo de Serviço
Descrição	Regra para identificação do tipo de serviço fornecido através das métricas definidas no SLS.
Expressão	SLS(?sls), Service(?service), includesBandwidth(?sls, ?bandwidth), includesBandwidthQualValue(?service, ?qualiBandwidth), includesDelay(?sls, ?delay), includesDelayQualValue(?service, ?qualiDelay), includesJitter(?sls, ?jitter), includesJitterQualValue(?service, ?qualiJitter), includesLossQualValue(?service, ?qualiLoss), includesPacketLoss(?sls, ?loss), includesQualitativeValue(?bandwidth, ?qualiBandwidth), includesQualitativeValue(?delay, ?qualiDelay), includesQualitativeValue(?jitter, ?qualiJitter), includesQualitativeValue(?loss, ?qualiLoss) -> definesSLS(?service, ?sls)
Conceitos	Service SLS
Atributos	
Relações binárias	includesBandwidth includesBandwidthQualValue includesDelay includesDelayQualValue includesJitter includesJitterQualValue includesPacketLoss includesLossQualValue includesQualitativeValue definesSLS
Buit-In	
Variáveis	?sls ?service ?bandwidth ?delay ?jitter ?loss ?qualiBandwidth ?qualiDelay ?qualiJitter ?qualiLoss

Apêndice B

Exemplo RDFa

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML+RDFa 1.0//EN"
"http://www.w3.org/MarkUp/DTD/xhtml-rdfa-1.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
xmlns:net="http://www.semanticweb.org/ontologies/2009/11/OntologyQoSNetwork.owl/"
xmlns:sls="http://www.semanticweb.org/ontologies/2009/11/OntologySLS.owl/"
xmlns:serv="http://www.semanticweb.org/ontologies/2009/11/OntologyQoSNetwork.owl/"
version="XHTML+RDFa 1.0" xml:lang="en">
<header>
<title>SLS1</title>
</header>
<body>
<h1>SLS1</h1>
<p about="http://www.semanticweb.org/ontologies/2009/11/OntologySLS.owl#SLS/SLS1"
typeof="http://www.semanticweb.org/ontologies/2009/11/OntologySLS.owl#SLS">
<p>ID: <span property="sls:id">SLS1</span>
</p>
<p>
Ingress Scope: <br />
<span rel="sls:includesIngressInterface"
resource="http://www.semanticweb.org/ontologies/2009/11/OntologyQoSNetwork.owl#Interface/N1I1">
<span about="http://www.semanticweb.org/ontologies/2009/11/OntologyQoSNetwork.owl#Interface/N1I1"
typeof="http://www.semanticweb.org/ontologies/2009/11/OntologyQoSNetwork.owl#Interface">
<span property="net:id">N1I1</span>
</span>
</span>
<br />
</p>
<p>
Egress Scope: <br />
<span rel="sls:includesEgressInterface"
resource="http://www.semanticweb.org/ontologies/2009/11/OntologyQoSNetwork.owl#Interface/N3I1">
<span about="http://www.semanticweb.org/ontologies/2009/11/OntologyQoSNetwork.owl#Interface/N3I1"
typeof="http://www.semanticweb.org/ontologies/2009/11/OntologyQoSNetwork.owl#Interface">
<span property="net:id">N3I1</span>
</span>
```

```
</span>
<br />
<span rel="sls:includesEgressInterface"
resource="http://www.semanticweb.org/ontologies/2009/11/OntologyQoSNetwork.owl#Interface/N2I1">
<span about="http://www.semanticweb.org/ontologies/2009/11/OntologyQoSNetwork.owl#Interface/N2I1"
typeof="http://www.semanticweb.org/ontologies/2009/11/OntologyQoSNetwork.owl#Interface">
<span property="net:id">N2I1</span>
</span>
</span>
<br />
</p>
<p>Traffic Classifiers: </p>
<span rel="sls:includesTrafficClassifier"
resource="http://www.semanticweb.org/ontologies/2009/11/OntologyQoSNetwork.owl#MF/SLS1Classifier">
<p about="http://www.semanticweb.org/ontologies/2009/11/OntologyQoSNetwork.owl#MF/SLS1Classifier"
typeof="http://www.semanticweb.org/ontologies/2009/11/OntologyQoSNetwork.owl#MF">
Traffic Classifier ID:
<span property="net:id">SLS1Classifier</span>
<br />
Logic Operator:
<span rel="net:includesLogicOperator"
resource="http://www.semanticweb.org/ontologies/2009/11/OntologyQoSNetwork.owl#AND">And</span>
<br />
Layer 3 Address in source: <span property="net:includesLayer3AddressSourceSpec">10.10.0.1</span>
<br />
Layer 4 Address in source: <span property="net:includesLayer4AddressSourceSpec">12345</span>
<br />
</p>
</span>
<p>Traffic Conditioners: </p>
<span rel="sls:includesTrafficConditioner"
resource="http://www.semanticweb.org/ontologies/2009/11/OntologyQoSNetwork.owl#TokenBucketPolicer/SLS1Conditioner1">
<p about="http://www.semanticweb.org/ontologies/2009/11/OntologyQoSNetwork.owl#TokenBucketPolicer/SLS1Conditioner1"
typeof="http://www.semanticweb.org/ontologies/2009/11/OntologyQoSNetwork.owl#TokenBucketPolicer">
Traffic Conditioner ID: <span property="net:id">SLS1Conditioner1</span>
<br />
Type: Policer
<br />
Algorithm: Token Bucket<br />
sr: <span property="net:includesSR">256.0</span><br />
bs: <span property="net:includesBS">1000.0</span><br />
In Profile Action:
<span rel="net:includesInProfileLevel"
resource="http://www.semanticweb.org/ontologies/2009/11/OntologyQoSNetwork.owl#MARK/1923943192">
<span about="http://www.semanticweb.org/ontologies/2009/11/OntologyQoSNetwork.owl#MARK/1923943192"
typeof="http://www.semanticweb.org/ontologies/2009/11/OntologyQoSNetwork.owl#MARK">
MARK_PACKETS Mark:
<span rel="net:includesResultingMark"
resource="http://www.semanticweb.org/ontologies/2009/11/OntologyQoSNetwork.owl#EF">
EF
</span>
</span>
</span>
<br />
      Out of Profile Action:
<span rel="net:includesOutProfileLevel"
```

```
resource="http://www.semanticweb.org/ontologies/2009/11/OntologyQoSNetwork.owl#DROP_PACKETS">
<span about="http://www.semanticweb.org/ontologies/2009/11/OntologyQoSNetwork.owl#DROP_PACKETS"
typeof="http://www.semanticweb.org/ontologies/2009/11/OntologyQoSNetwork.owl#DROP">DROP_PACKETS</span>
</span>
<br />
</p>
</span>
<p>
Metrics
<br />
<span rel="sls:includesBandwidth"
resource="http://www.semanticweb.org/ontologies/2009/11/OntologySLS.owl#Bandwidth/816963716">
<span about="http://www.semanticweb.org/ontologies/2009/11/OntologySLS.owl#Bandwidth/816963716"
typeof="http://www.semanticweb.org/ontologies/2009/11/OntologySLS.owl#Bandwidth">
Bandwidth: <span property="sls:metricValue">1024.0</span>
<span property="sls:unit">Kbps</span>
</span>
</span>
<br />
<span rel="sls:includesDelay"
resource="http://www.semanticweb.org/ontologies/2009/11/OntologySLS.owl#Delay/1402794579">
<span about="http://www.semanticweb.org/ontologies/2009/11/OntologySLS.owl#Delay/1402794579"
typeof="http://www.semanticweb.org/ontologies/2009/11/OntologySLS.owl#Delay">
Delay: <span property="sls:metricValue">50.0</span>
<span property="sls:unit">ms</span>
</span>
</span>
<br />
<span rel="sls:includesJitter"
resource="http://www.semanticweb.org/ontologies/2009/11/OntologySLS.owl#Jitter/329335733">
<span about="http://www.semanticweb.org/ontologies/2009/11/OntologySLS.owl#Jitter/329335733"
typeof="http://www.semanticweb.org/ontologies/2009/11/OntologySLS.owl#Jitter">
Jitter: <span property="sls:metricValue">0.1</span>
<span property="sls:unit">ms</span>
</span>
</span>
<br />
<span rel="sls:includesPacketLoss"
resource="http://www.semanticweb.org/ontologies/2009/11/OntologySLS.owl#PacketLoss/2135650992">
<span about="http://www.semanticweb.org/ontologies/2009/11/OntologySLS.owl#PacketLoss/2135650992"
typeof="http://www.semanticweb.org/ontologies/2009/11/OntologySLS.owl#PacketLoss">
Packet Loss: <span property="sls:metricValue">0.1</span>
       <span property="sls:unit">%</span>
</span>
</span>
<br />
<span rel="sls:includesReliability"
resource="http://www.semanticweb.org/ontologies/2009/11/OntologySLS.owl#Reliability/542364229">
<span about="http://www.semanticweb.org/ontologies/2009/11/OntologySLS.owl#Reliability/542364229"
typeof="http://www.semanticweb.org/ontologies/2009/11/OntologySLS.owl#Reliability">
Reliability: <span property="sls:metricValue">30.0</span>
       <span property="sls:unit">min</span>
</span>
</span>
<br />
</p>
```

```
<p>
Schedule
<br />
<span rel="sls:includesSchedule"
resource="http://www.semanticweb.org/ontologies/2009/11/OntologySLS.owl#ServiceSchedule/360263043">
<span about="http://www.semanticweb.org/ontologies/2009/11/OntologySLS.owl#ServiceSchedule/360263043"
typeof="http://www.semanticweb.org/ontologies/2009/11/OntologySLS.owl#ServiceSchedule">
Start Date :<span property="sls:startDate">15:29:58.237</span>
<br />
</span>
</span>
<br />
</p>
</p>
</body>
</html>
```