Autoria e Simulação de Cenários de Redes em NS-3

DISSERTAÇÃO DE MESTRADO

José Jorge Fernandes Sousa
MESTRADO EM ENGENHARIA INFORMÁTICA

UNIVERSIDADE da MADEIRA
A Nossa Universidade
www.uma.pt

Setembro | 2011

Autoria e Simulação de Cenários de Redes em NS-3

DISSERTAÇÃO DE MESTRADO

José Jorge Fernandes Sousa
MESTRADO EM ENGENHARIA INFORMÁTICA

ORIENTAÇÃO
Paulo Nazareno Maia Sampaio

Autoria e Simulação de Cenários de Redes em NS-3

José Jorge Fernandes Sousa

Tese submetida à Universidade da Madeira para a
Obtenção do Grau de Mestre em Engenharia Informática

Funchal – Portugal

Setembro 2011

ORIENTADOR

Professor Doutor Paulo Nazareno Maia Sampaio

Professor Auxiliar do Centro de Ciências Exactas e Engenharias

CO-ORIENTADOR

Professor Eduardo Miguel Dias Marques

Professor Assistente do Centro de Ciências Exactas e Engenharias

ABSTRACT

This work is interested in the increasing need of networks to be tested and verified before the actual implementation in the real world. Variables such as investment, quality-cost study, management issues, reliability and lifetime operation are part of the main concerns associated with the existing and future networks.

This project focuses precisely on this issue, providing a solution for the optimization of network scenarios management (modeling, simulation, implementation and, monitoring) and in particular the simulation of these scenarios using the network simulator tool NS-3.

Emerging technologies are the most common targets of network simulators and this plays an important role while deciding which network simulator people choose to use to simulate their scenarios.

With this work, the main goal is to provide users with a collaborative tool that allows them to describe their own network scenarios, based on an XML-based language called NSDL and translate this XML structure to a proper C++ *script* which can be executed on an evolutionary environment, namely NS-3.

The proposed solution allows the fast development of networks scenarios through an XML structure that will be used to translate to C++, which in turn will be executed in NS-3.

The main problematic associated with this solution is the need for constant update for both validation and translation of XML documents (aiming at NS-3 representation) since the evolution or underlying technologies are also constant.

KEYWORDS

Network Scenario Description Language

Authoring of Network Scenarios

Network Simulation

Quality of Service

XML

Wireless

RESUMO

Este trabalho aborda a crescente necessidade de verificação e testes das redes antes da sua implementação no mundo real. Variáveis como o investimento, estudo qualidade-custo, questões de gestão, fiabilidade e tempo de vida útil são parte das principais preocupações associadas a existentes e futuras redes.

É sobretudo neste âmbito que este projecto opera, fornecendo uma solução para uma optimização da autoria de cenários de redes visando a sua execução na ferramenta de simulação de redes NS-3.

As tecnologias emergentes são os alvos mais comuns de ferramentas de simulação de redes e este facto é determinante ao decidir que simulador de redes usar para simular os seus cenários.

Com este trabalho, o principal objectivo é fornecer aos utilizadores uma ferramenta colaborativa que permite descrever os seus próprios cenários de redes, baseando-se numa linguagem XML denominada por NSDL e traduzir esta estrutura XML para um *script* C++ que possa ser executado num ambiente evolucionário, como é o caso do NS-3.

A solução proposta permite a rápida criação de cenários de redes através de estruturas de dados XML para posterior tradução para C++ e então, execução no NS-3. A maior problemática associada a esta solução é a necessidade de actualização constante em estruturas de validação e de tradução de documentos XML, aquando da actualização da ferramenta NS-3, o que tem vindo a acontecer muito frequentemente devido à sua própria evolução.

PALAVRAS-CHAVE

Network Scenario Description Language

Autoria de cenários de Redes

Simulação de Redes

Qualidade de Serviço

XML

Redes sem fio

AGRADECIMENTOS

Em primeiro lugar agradeço ao Professor Doutor Paulo Nazareno Maia Sampaio, por ter aceitado ser o meu Orientador para esta dissertação e por toda a ajuda, paciência e tempo disponibilizados, que só demonstraram a grande capacidade de orientação ao longo das várias fases do projecto.

Outro agradecimento especial é dirigido ao Professor Eduardo Miguel Dias Marques por todo o tempo, ajuda e material disponibilizados que contribuíram de forma decisiva para uma compreensão correcta de toda a abordagem técnica associada à autoria de cenários de redes.

Aos meus colegas que deram-me ânimo e força para a elaboração desta tese nomeadamente o colega Jorge Gonçalves que mostrou-se disponível para rever a minha escrita precária em inglês.

À minha família, por se mostrar compreensiva quanto às minhas muitas ausências em eventos familiares por necessidade de trabalhar no projecto.

DEDICATÓRIA

A Deus que muniu-me de capacidades necessárias para ultrapassar os momentos mais difíceis que enfrentei ao longo destes 5 anos de percurso académico.

Aos meus pais que sempre lutaram para que eu pudesse alcançar o que nunca lhes foi possibilitado dadas as dificuldades que passaram enquanto jovens em continuar os estudos.

À minha família mais próxima, as minhas irmãs Fanny e Corina e os seus respectivos maridos, Martinho e Lino, por me ajudarem a encontrar a motivação necessária à elaboração deste projecto.

À minha irmã falecida Vanessa que certamente estaria orgulhosa por ver-me a concluir o curso que escolhi.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

ÍNDICE DE GRÁFICOS

LISTA DE ACRÓNIMOS

3G	3rd Generation
3GPP	3rd Generation Partnership Project
4G	4th Generation
AODV	Ad hoc On-Demand Distance Vector
AP	Access Point
ARP	Address Resolution Protocol
ATM	Asynchronous Transfer Mode
AWK	Aho Weinberger Kernighan
BE	Best Effort
BS	Base Station
BSPK	Binary Phase Shift Keying
CARA	Collision-Aware Rate Adaptation
CBR	Constant Bit Rate
CS	Convergence Sublayer
CSMA	Carrier Sense Multiple Access
dBm	deciBel miliwatt
DiffServ	Differentiated Services
DSDV	Destination-Sequenced Distance Vector
DTD	Document Type Definition
E-UTRAN	Evolved-Universal Terrestrial Radio Access Network
eNB	Enhanced NodeB
FDD	Frequency Division Duplex
FEC	Forward Error Correction
FLAME	Forwarding LAyer for MEsh
FTP	File Transfer Protocol
GUI	Graphical User Interface
HWMP	Hybrid Wireless Mesh Protocol
IEEE	Institute of Electrical and Electronics Engineers
IMT	International Mobile Telecommunications
IntServ	Integrated Services
IP	Internet Protocol
LAN	Local Access Network
LENA	LTE/EPC Network simulAtor
LTE	Long Term Evolution
MAC	Medium Access Control
MAP	Mesh Access Point
Mb	Megabit
Mbps	Megabits por segundo
MHz	MegaHertz
MIMO	Multiple-Input and Multiple-Output
MP	Mesh Point
MPI	Message Passing Interface
MPLS	Multiprotocol Label Switching
MPP	Mesh Portal Point
ms	milissegundos
MVC	Model View Control
NAT	Network Address Translation
nrtPS	near-real-time Processing System
NS-2	Network Simulator 2
NS-3	Network Simulator 3

NSDL	Network Scenario Description Language
OFDM	Orthogonal Frequency-Division Multiplexing
OFDMA	Orthogonal Frequency-Division Multiple Access
OLSR	Optimized Link State Routing
OS	Operating System
OSI	Open Systems Interconnection
OSPF	Open Shortest Path First
OTcl	Object Tool Command Language
P2P	Point-to-Point
PCAP	Packet Capture
PDU	Protocol Data Unit
PHP	Hypertext Preprocessor
PHY	Physical
POO	Programação Orientada a Objectos
PMP	Peering Management Protocol
PPP	Point-to-Point Protocol
QAM	Quadrature Amplitude Modulation
QoS	Quality of Service
QPSK	Quadrature Phase Shift Keying
RIP	Routing Information Protocol
RRAA	Robust Rate Adaptation Algorithm
RREP	Route Reply
RREQ	Route Request
rtPS	real-time Processing System
RX	Received
s	segundos
SAP	Service Access Point
SNR	Signal-to-Noise Ratio
SS	Subscriber Station
STA	Station
TAR	Tecnologias Avançadas de Redes
Tcl	Tool Command Language
TCP	Transmission Control Protocol
TDD	Time Division Duplex
TX	Transmited
UAN	Underwater Acoustic Network
UDP	User Datagram Protocol
UE	User Equipment
UGS	Unsolicited Grant Service
UML	Unified Modeling Language
UMTS	Universal Mobile Telecommunication System
us	microssegundos
VLAN	Virtual Local Area Network
VND	Visual Network Descriptor
VNS	Visual Network Simulator
WAN	Wide Area Network
WEP	Wired Equivalent Privacy
Wi-Fi	Wireless Fidelity
WiMAX	Worldwide Interoperability for Microwave Access
WPA	Wi-Fi Protected Access
XBNSDL	XML Based Network Simulation Description Language
XML	Extensible Markup Language
XSD	XML Schema Definition
XSLT	eXtensible Stylesheet Language for Transformation
Yans	Yet another network simulator

1. INTRODUÇÃO

Nos últimos tempos tem sido observada uma crescente expansão das redes bem como a necessidade de um melhor planeamento antes da instalação de redes. É neste contexto que entram os simuladores de redes, plataformas que incluem diversos modelos tecnológicos relativos às tecnologias de redes, nomeadamente as mais utilizadas [Heidemann, 2003][Al-Shaer, 2011].

Apesar de os simuladores de redes não serem 100% fiáveis, permitem aos seus utilizadores investigar a rede que pretendem implementar e, acima de tudo verificar qual o custo da mudança: Questões como "E se utilizasse 1000 nós finais?" ou "Irá a rede implementada adaptar-se facilmente à tomada de outras tecnologias que não as actualmente utilizadas?" entre outras, são sempre aspectos a ter em consideração antes de proceder à concretização da rede. As principais razões pela qual é recomendável a simulação prévia à instalação da rede são [Hughes, 2009]:

- A experimentação no mundo real pode danificá-la;
- A modelação/análise requer grandes níveis de abstracções e falha na especificação de detalhes;
- Ajuda a perceber as necessidades tanto dos seus utilizadores, como das aplicações que irão operar sobre a rede, e;
- Mudanças numa rede existente são dispendiosas: Erros em más opções tecnológicas podem custar muito dinheiro a rectificá-los.

De uma forma geral, o presente documento destaca a temática de simulação de redes e todo o processo de concepção de cenários de redes, desde a modelação dos cenários à simulação dos mesmos e que conclusões podem ser extraídas perante os resultados.

1.1. MOTIVAÇÃO

Como foi referido anteriormente, um estudo precoce de uma rede com auxílio de um simulador de redes constitui um importante passo no projecto de redes informáticas.

Actualmente existem diversos simuladores de redes que se distinguem entre si mediante o conjunto de opções tecnológicas que disponibilizam, pelo custo da ferramenta (se possui licença ou é de utilização gratuita) e acima de tudo pela documentação oferecida.

Por vezes os utilizadores vêm-se forçados a optar pela utilização de diferentes simuladores relativamente àqueles que tinham por hábito utilizar, o que implica geralmente um esforço de adaptação. As opções tecnológicas oferecidas são o principal factor que leva à mudança.

Uma consequência directa destas mudanças é a necessidade de aprendizagem dos conceitos associados à nova plataforma de simulação, nomeadamente a sintaxe que esta utiliza para a descrição dos seus cenários. Nesta situação fica clara a necessidade de interoperabilidade entre os vários simuladores de redes. É neste contexto que surge a *Framework* NSDL que funciona como uma ponte comunicativa entre ferramentas de autoria textual/gráfica de cenários de redes e entre os simuladores de redes que permitem testar os cenários modelados.

Com a introdução da linguagem **NSDL** (***N**etwork **S**cenario **D**escription **L**anguage*) [Marques, 2010], torna-se muito mais simples transitar de cenários com perfis de ferramenta *X* para um cenário que visa o perfil de uma ferramenta *Y*, modificando para o processo apenas alguns aspectos da nomenclatura dos atributos. Um exemplo concreto desta facilidade é a rápida transformação de um cenário com perfil NS-2 para um cenário NS-3, que têm por linguagens alvo o OTcl [OTcl, 2011] e o C++ [Cplusplus, 2011] respectivamente.

O trabalho apresentado nesta dissertação visa a optimização da autoria de cenários de redes com a introdução de um perfil NS-3 para o NSDL.

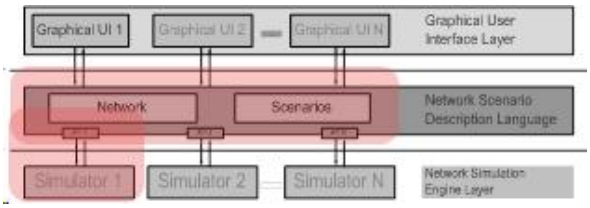

Figura 1 - Arquitectura NSDL com visão ao âmbito do projecto

Tal como podemos observar na Figura 1, a framework NSDL que é prontamente apresentada no Capítulo 4, é constituída essencialmente por 3 camadas (layers):

- **Camada superior** (*Graphical User Interface Layer)*- Inclui as ferramentas de autoria gráfica de cenários de redes;
- **Camada intermédia** (*Network Scenario Description Language*) – Possui a especificação à linguagem NSDL e os seus perfis, e;
- **Camada inferior** (*Network Simulation Engine Layer)* – Inclui um conjunto de ferramentas de redes tais como simuladores de redes e ferramentas de virtualização [Araújo, 2011] actualmente existentes.

A área sombreada constitui a principal contribuição realizada: A estipulação de um perfil NS-3 para NSDL (elementos *Network* e *Scenarios*) e o simulador a utilizar (*Simulator 1* como o NS-3).

1.2. PRINCIPAIS CONTRIBUIÇÕES

As principais contribuições realizadas com este projecto são as seguintes:

- Introdução à plataforma NS-3 com base num estudo que inclui a descrição modular desta plataforma, principais funcionalidades e uma pequena discussão sobre a vertente vantagens/desvantagens;

- Proposta de uma solução de mapeamento NSDL para NS-3 atendendo a um conjunto de tecnologias desenvolvidas na plataforma NS-3 e que são de especial interesse para a comunidade científica de redes em geral;
- Implementação da solução apresentada, e;
- Verificação e validação da solução desenvolvida com auxílio a um conjunto de cenários representativos e a respectiva discussão dos resultados perante a simulação em NS-3.

1.3. ORGANIZAÇÃO DO DOCUMENTO

Esta dissertação contém 8 capítulos organizados da seguinte forma:

O Capítulo 1 marca a introdução ao projecto referenciando áreas científicas abordadas por este trabalho bem como que razões levaram ao surgimento deste projecto e principais contribuições que serão apresentadas ao longo do documento;

Capítulo 2 apresenta os principais conceitos abordados neste projecto contando com a relevância da autoria e simulação de cenários de redes no mundo actual, uma descrição das principais tecnologias consideradas para o mapeamento entre as linguagens NSDL (XML) e NS-3 (C++) e trabalhos relacionados;

Capítulo 3 contém um estudo realizado sobre a ferramenta de simulação de redes NS-3, incluindo um enquadramento histórico da ferramenta, descrição das suas principais funcionalidades, breve apresentação dos seus módulos e um apanhado das principais vantagens e desvantagens associadas à ferramenta;

Capítulo 4 apresenta a *Framework* NSDL através das suas 3 camadas principais que contam com ferramentas de autoria de cenários de redes (camada superior), a linguagem NSDL (camada intermédia e responsável pela comunicação entre as camadas superior e inferior) e as ferramentas de simulação de redes (camada inferior);

Capítulo 5 detém a proposta de um *perfil* NS-3 para NSDL em que são apresentados os novos objectos que foram introduzidos no mundo NSDL de forma a cobrir minimamente as principais tecnologias consideradas para o efeito;

Capítulo 6 descreve a metodologia utilizada para o mapeamento entre as estruturas de dados NSDL e os scripts de NS-3;

Capítulo 7 inclui os casos de estudo com recurso a 7 cenários de redes, os primeiros 4 com a abordagem comparativa entre as plataformas NS-2 e NS-3 enquanto que os restantes cenários demonstram que outro tipo de conclusões podem ser extraídas de cenários essencialmente orientados ao NS-3, e;

Por último, o Capítulo 8 descreve as principais conclusões obtidas ao longo do desenvolvimento deste projecto, uma retrospectiva global do trabalho realizado e perspectivas futuras quanto à evolução da contribuição realizada.

2. ESTADO DE ARTE

Neste capítulo são abordados os principais conceitos relacionados com a temática do projecto desenvolvido, nomeadamente conceitos sobre autoria e simulação de cenários de redes bem como um conjunto de tecnologias que são utilizadas no processo.

2.1. INTRODUÇÃO

Actualmente a concepção de cenários de redes para simulação constitui uma actividade que engloba diversas tecnologias desde as que são utilizadas no âmbito de fornecer suporte ao desenvolvimento dos cenários, até às tecnologias que são alvo de especificação nos diversos cenários de redes.

É portanto objectivo deste capítulo apresentar sucintamente as principais tecnologias relacionadas com o desenvolvimento deste projecto.

Nas secções seguintes será feita uma descrição de cada tecnologia abordada e o seu estado actual em termos de relevância teórico-prática nos dias correntes.

2.2. ABORDAGEM CONCEPTUAL

No decorrer do desenvolvimento deste projecto, foram abordados diversos conceitos e tecnologias em que se destacam a autoria de cenários de redes, a simulação de cenários de redes, tecnologias relacionadas com desenvolvimento Web e tecnologias *Wireless*.

De seguida serão apresentados cada um destes conceitos e tecnologias de forma a facilitar um melhor entendimento sobre os mesmos e a sua relevância neste projecto.

2.2.1. AUTORIA DE CENÁRIOS DE REDES

Com a crescente importância de comunicação nos dias de hoje, é impensável desenvolver um negócio sem que para o mesmo seja projectada uma rede intranet e/ou com integração da internet: Para uma simples rede doméstica, o leque de possibilidades tecnológicas é de tal forma expansivo que um estudo prévio sobre as tecnologias a adoptar constitui uma tarefa indispensável.

Que protocolos de encaminhamento de tráfego usar, que cablagem escolher, quais os equipamentos activos devem ser adquiridos, a viabilidade de um estudo entre tecnologia/preço/necessidade são algumas das questões que devem ser devidamente

respondidas antes de proceder à implementação da rede projectada. É neste âmbito que surge o conceito de autoria de cenários de redes em que diversas aplicações foram desenvolvidas nas últimas décadas de forma a dar suporte à autoria de redes atendendo aspectos fundamentais tais como:

- Suporte representativo aos diversos dispositivos físicos que uma rede aglomera, gráfico ou textual (computadores, *switches*, *routers*, encaminhadores, servidores, pontes, pontos de acesso, cablagem, etc.);
- Facilidades de utilização, munindo os seus utilizadores de manuais de utilização que facilitem o processo de aprendizagem;
- Suporte na configuração de diversas tecnologias sob os equipamentos seleccionados sempre que estes o permitirem, e;
- Interoperabilidade, na medida em que cenários que tenham sido desenvolvidos numa determinada máquina possam ser facilmente usados/modificados por terceiros.

Uma vez modelado um determinado cenário, deve ser feita uma avaliação do seu comportamento perante a possibilidade de implementação do mesmo no mundo real. Para tal, deverá ser utilizado um simulador de redes, tema de discussão da secção seguinte.

Como exemplos de ferramentas de autoria de cenários de redes existem por exemplo o PacketTracer [Cisco, 2011], concebido pela Cisco para âmbito educacional, constituindo assim uma ferramenta gratuita e detentora de um grande conjunto de tecnologias que podem ser utilizadas pelos seus dispositivos: Entre estas tecnologias destacam-se a interface 802.11 (Wi-Fi), comutadores **VLAN** (**V**irtual **L**ocal **A**rea **N**etwork), protocolos de encaminhamento **RIP**v1 (**R**outing **I**nformation **P**rotocol), RIPv2 e **OSPF** (**O**pen **S**hortest **P**ath **F**irst), **NAT** (**N**etwork **A**ddress **T**ranslation), etc.

Outras ferramentas de modelação são o **VNS** (**V**isual **N**etwork **S**imulator) e **VND** (**V**isual **N**etwork **D**escriptor) que serão abordados nos trabalhos relacionados.

2.2.2. SIMULAÇÃO DE REDES

O processo de simulação de uma rede e análise correcta dos seus resultados pode prevenir desperdícios avultados no investimento inicial no projecto da rede. Através da simulação é possível verificar a viabilidade ou não de optar por determinadas tecnologias ao invés de outras. Utilizar um simulador de redes antes de conceber a rede no mundo real permite também definir estratégias futuros quanto à evolução da rede: Isto significa que eventuais mudanças a nível de dispositivos físicos ou até mesmo de tecnologias usadas possam existir e portanto, usar um simulador de redes no sentido de antever estas e outras possibilidades é um factor decisivo em qualquer desenvolvimentos de redes.

A escolha do simulador a utilizar irá prender-se com a possibilidade de este responder às questões anteriormente mencionadas além de, obviamente possuir suporte às tecnologias pretendidas.

De entre os vários simuladores de redes existentes, destacam-se os mais

utilizados:

Network **S**imulator **2** (**NS-2**) [VINT, 1995][Lucio, 2011][Weingartner, 2009] – Este primeiro simulador de redes constitui o simulador mais utilizado no estudo de cenários de redes em todo o mundo e a sua primeira versão foi lançada em 1996. Trata-se de uma aplicação desenhada em C++ e com recurso à especificação de cenários de redes com a linguagem OTcl;

OPNET [Lucio, 2011][OPNET, 2011] – Simulador modular com um conjunto de cerca de 400 funções próprias para a modelação de cenários de redes através de uma interface gráfica GUI (*Graphical User Interface*); Não é um simulador *open-source* - no entanto os seus parâmetros podem ser alterados, permitindo assim atingir valores mais realísticos;

QUALNET [SNT, 2001] – O simulador QUALNET representa também um simulador com recurso a uma GUI, à semelhança do OPNET mas disponibiliza a possibilidade de modelação textual dos cenários; Possui módulos para diversas tecnologias com e sem fios, permitindo uma escalabilidade numérica até 50.000 nós;

NetSim [Tetcos, 2011] – Esta ferramenta de simulação de redes é muito popular no meio académico pelo seu uso em cadeiras de redes e pelo facto de ser uma aplicação *open-source*, com suporte à modelação de diversas tecnologias e especial destaque para a funcionalidade de captura de pacotes. Baseia-se em codificação C / C++ / Java Programming e permite a modificação da configuração da GUI integrada;

Network **S**imulator **3** (**NS-3**) [Nsnam, 2011] – Este simulador é o que mais se tem destacado ultimamente, não só pelo facto de ser o sucessor do NS-2, mas pelo facto de apresentar uma distinta metodologia de desenvolvimento dos seus módulos relativamente ao NS-2. É também um simulador *open-source* que acomoda diversas contribuições de investigadores em todo o mundo. O seu desenvolvimento foi iniciado em 2006 e conta já com um conjunto amplo de tecnologias suportadas, nomeadamente tecnologias emergentes como o Wi-Fi, WiMAX e o LTE, o que também contribuiu em parte para que esta ferramenta fosse alvo de um projecto que foi concretizado ao longo desta tese.

De uma forma geral, para que um simulador de redes seja fiável na medida em que é capaz de avaliar um cenário de redes conforme a sua definição física e comportamental, este deve respeitar de entre outras, as seguintes características:

- Possuir documentação quantitativa e qualitativa, que seja regularmente actualizada perante eventuais alterações nas estruturas de dados presentes nos seus módulos;
- Um bom compromisso entre escalabilidade/recursos físicos para que cenários com grande conjunto de especificação possam ser utilizados sem comprometer os recursos disponibilizados pelo sistema operador que o hospeda;
- Grande leque de opções tecnológicas, visando a abrangência de um grande número de utilizadores e consequentemente, maior suporte para detecção de eventuais falhas/bugs na plataforma;

- Manter com regularidade os módulos constituintes da plataforma para que estes não entrem em desuso/desactualização, e;

- Interoperabilidade, nomeadamente a possibilidade de simulação em Windows, MAC OS e ambientes baseados em Linux.

Esta foi a abordagem aos simuladores de redes, que naturalmente o mais notório para o âmbito do projecto desta tese é o NS-3.

A escolha desta ferramenta em detrimento de as restantes apresentadas para a criação de um *perfil* para NSDL deve-se sobretudo às seguintes razões:

O NS-3 é o potencial sucessor da ferramenta de simulação de redes NS-2 amplamente utilizada em todo o mundo;

O NS-3 está a sofrer grande evolução o que também é interessante acompanhar o desenvolver de esta ferramenta paralelamente ao projecto realizado;

Constitui uma ferramenta com muita documentação disponibilizada que ajuda investigadores com pouca experiência a adaptarem-se facilmente aos conceitos adjacentes ao NS-3, e;

É uma ferramenta gratuita e *open-source* o que a torna mais acessível em relação a simuladores de redes que possuem licenças.

2.2.3. TECNOLOGIAS WIRELESS

No desenvolvimento de este projecto foram abordadas diversas tecnologias para serem alvo de mapeamento entre as linguagens NSDL (estrutura XML) e NS-3 (*script* C++). Entre estas tecnologias, destacam-se as que têm sido alvo de grande foco por parte da equipa de desenvolvimento NS-3 e que constitui a maioria dos tópicos sobre discussões presentes no fórum do NS-3.

As tecnologias abordam sobretudo a temática sem fios (*wireless*) pois constitui uma área de estudo emergente visto a crescente tendência de acesso à internet via dispositivos móveis.

De seguida será realizada uma breve descrição das principais tecnologias abordadas.

WI-FI

Figura 2 - Logotipo Wi-Fi [Wi-Fi Alliance, 2011]

A tecnologia **Wi-Fi** (**Wi**reless **Fi**delity) foi desenvolvida pela entidade IEEE (*Institute of Electrical and Electronics Engineer*) em meados dos anos 90 [Lehr, 2003][Prendergas, 2004]. Constitui a tecnologia mais utilizada no que diz respeito a tecnologias *wireless*. Esta tecnologia utiliza a frequência de rádio para transmissão de dados no meio. Esta por sua vez possui diversos *standards* que podem ser estabelecidos aquando da configuração dos

equipamentos activos (pontos de acesso - *Access Points* - e estações móveis - *non-Access Points*).

A inclusão de uma interface Wi-Fi que antes era opcional, agora é obrigatória.

Em termos de topologia, esta tecnologia é sobretudo utilizada em LANs (*Local Area Networks*) e em casos especiais, em WANs (*Wide Area Networks*)

De entre os diversos *standards*, destacam-se os *standards* 802.11a, 802.11b, 802.11g e 802.11n em que a sua escolha irá depender dos pré-requisitos estabelecidos para a rede a implementar. Na Tabela 1 podemos visualizar as principais características de cada variante.

Tabela 1 – Frequência, Velocidade e Distância por cada um dos principais *standards* Wi-Fi [Amped, 2011]

Wi-Fi Standards	Frequency	Wireless Speed (Max)		Wireless Distance (Max)
802.11a (1999)	5 GHz	54 Mbps		390 ft
802.11b (1999)	2.4 GHz	11 Mbps		460 ft
802.11g (2003)	2.4 GHz	54 Mbps		460 ft
802.11n (2009 - current)	2.4 GHz	1 Tx/Rx	2 Tx/Rx	820 ft
		150 Mbps	300 Mbps	

Na Tabela 1 pode ser observada a data de introdução de cada um dos principais *standards* de Wi-Fi, as velocidades máximas de transmissão que cada um suporta e as distâncias máximas permitidas entre os pontos de acesso e os dispositivos móveis.

A nível de segurança, foram realizados avanços que permitem uma maior fiabilidade perante o uso de tecnologias Wi-Fi. Na Tabela 2 são apresentados os 3 tipos de segurança utilizados pelo Wi-Fi e o respectivo nível de segurança em que se destaca o WPA2 embora o WEP seja ainda amplamente utilizado em diversas redes que utilizam Wi-Fi, predominantemente com o *standard* 802.11g.

Tabela 2 - Segurança em tecnologias Wi-Fi [Amped, 2011]

Wi-Fi Security	WEP Wired Equivalent Privacy	WPA Wi-Fi Protected Access	WPA2 Wi-Fi Protected Access 2
Year	1999	2003	2004
Security Strength	LOW	MEDIUM	HIGH

MESH

A tecnologia Mesh constitui uma variante particular de Wi-Fi denominada por *standard* 802.11s *draft* que se distingue dos anteriores standards mencionados pelo seu

funcionamento e topologia [Hiert, 2010].

Os dispositivos incluídos numa arquitectura Mesh típica são:

Estação ou dispositivo móvel **(STA**) – Nó que recebe ou envia tráfego mas não possui funcionalidades de reencaminhamento de tráfego;

Mesh **P**oint (**MP**) – Nó que participa activamente na descoberta de rotas através de protocolos próprios para o efeito;

Mesh **A**ccess **P**oint (**MAP**) – Trata-se de um MP agregado a um ponto de acesso e que disponibiliza serviços aos diversos STAs a si conectados, e;

Mesh **P**ortal **P**oint (**MPP**) – Representa um nó MP que funciona como um *gateway* permitindo a comunicação com redes heterogéneas (por exemplo conectividade à internet).

Relativamente à selecção de rotas, este procedimento é realizado com a utilização do protocolo de encaminhamento **HWMP** (**H**ybrid **W**ireless **M**esh **P**rotocol) que dispõe de 2 modos de funcionamento:

On-demand route discovery – Quando um nó *X* pretende comunicar com um nó *Y*, o primeiro envia mensagens **RREQ** (**R**oute **Req**uest) para os seus vizinhos até que seja encontrado *Y* e este por sua vez envie uma confirmação através da mensagem **RREP** (**R**oute **Rep**ly) que iniciará uma sessão bidireccional permitindo a transferência de pacotes entre os dois nós; Se nenhum nó contido na rede Mesh enviar um RREP o nó *X* assume que *Y* está fora da rede Mesh e procede ao MPP, e;

Proactive route discovery – A configuração de este modo de selecção de rotas exige a configuração de um MP *Root* (MP raíz) que irá representar a raíz de uma árvore de possíveis rotas e sempre que um nó *X* pretenda comunicar com um nó *Y* e *X* não possui uma rota estipulada para comunicar com *Y*, a mensagem é enviada para o MP *Root* que irá responsabilizar-se pelo correcto encaminhamento da rota mais curta a partir da sua localização. Se *Y* não se encontra nas suas ramificações assume-se que *Y* está fora da rede Mesh e procede ao MPP.

Tal como o Wi-Fi, a tecnologia Mesh é sobretudo utilizada em redes LAN e WAN.

Actualmente existem projectos em desenvolvimento com o objectivo da optimização deste *standard* e tornar o mesmo como standard de código livre (*open-source*), como é o caso do projecto *open80211s* [Open80211s, 2011].

WiMAX

WiMAX (**W**orldwide **I**nteroperability for **M**icrowave **Acces**s) é uma tecnologia que é utilizada para acesso internet de dispositivos fixos (*desktops*, *servers*) e móveis (*laptops*, telemóveis, etc.) [Gabriel, 2011][Roh, 2009][Eberle, 2011].

Figura 3 - Logotipo WiMAX [EngWeb, 2011]

O seu primeiro *standard* foi lançado pelo IEEE em 2001 como standard 802.16. Este viria a sofrer grandes

alterações ao longo do tempo em que se destaca a versão de 2004 com alguns melhoramentos em 2005 com um *standard* conhecido por IEEE 802.16e-2005.

Os dispositivos utilizados nesta tecnologia são sobretudo de 2 tipos distintos:

Base **S**tation (**BS**) – Estação responsável pelo encaminhamento do tráfego entre os SSs registados na sua tabela;

Subscriber **S**tation (**SS**) – Estação que envia ou recebe fluxos de dados para uma BS em que está registada para posterior reencaminhamento para o seu destino.

Entre as principais características desta tecnologia, destacam-se as seguintes:

Camada inferior (*Physical Layer*)

- A propagação do sinal pode ser realizada utilizando **OFDMA** (***O**rthogonal **F**requency **D**ivision **M**ultiple **A**ccess*) para ambientes sem linha em vista (**non-line-of-sight**), por **TDD** (***T**ime-**D**ivision **D**uplexing*), **FDD** (***F**requency-**D**ivision **D**uplexing*), single-carrier, etc.;
- Altas taxas de velocidade: Com a utilização da técnica **MIMO** (**M**ultiple-**I**nput and **M**ultiple-**O**utput) o suporte a *downlink* pode atingir cerca de 128 Mbps enquanto que o *uplink* poderá atingir os 56 Mbps de velocidade quando utilizada a frequência de 20 Mhz, e;
- A modulação do sinal pode ser configurada utilizando **QPSK** (***Q**uadrature **P**hase **S**hift **K**eying*), **16QAM** (***Q**uadrature **A**mplitude **M**odulation*) ou ainda **64QAM**.

Camada de *Link* (***MAC** – **M**edium **A**ccess **C**ontrol Layer*)

Acordos entre as BSs e SSs são realizados de um algoritmo de agendamento contido na BS. Uma vez respeitado este algoritmo, é fornecida à SS uma vaga de acesso (*access slot*). Uma vaga de acesso possui um tempo de expiração e especificações de **QoS** (***Q**uality **of S**ervice*) para realização de transferência de dados;

Para além de atributos específicos de QoS presentes nesta camada (como *Maximum sustained traffic rate, Minimum reserved traffic rate, Maximum latency, Tolerated jitter, Traffic priority* e *Request/transmission policy*), também pode ser definido qual classe de serviço está atribuída a uma determinada vaga. As classes de serviço existentes são:

- **UGS – U**nsolicited **G**rant **S**ervice – Aplicações com tamanho de pacote fixo;
- **rtPS – R**eal-**T**ime **P**olling **S**ervice **-** Aplicações com tamanho de pacote fixo em tempo real;
- **nrtPS – non (rtPS) -** Aplicações com tamanho de pacote variável, e;
- **BE – B**est **E**ffort **–** Alocação de banda larga (*bandwidth*) apenas quando disponível.

Camada de Aplicação (*Application Layer*)

Possibilidade de alocação de largura de banda por parte das BSs perante uma contratualização com uma SS para que esta possua garantias de serviço que complementam as eventuais especificações de QoS realizadas. Estas operações são geridas por *ServiceFlows* existentes em cada SS.

Ambas as tecnologias WiMAX e LTE (que irá ser discutida de seguida) encontram-se actualmente na categoria 3G (*3rd Generation*) e a integração destas tecnologias na 4G está agendada ainda para o ano corrente, após ambas corresponderem aos requisitos mínimos estabelecidos pelo **IMT-Advanced** (***I**nternational **M**obile **T**elecommunications Advanced*) em que se destacam os requisitos:

- ✓ Velocidade máxima de transferência para dispositivos móveis: 100Mbps
- ✓ Velocidade máxima de transferência para dispositivos fixos: 1 Gbps
- ✓ Arquitectura IP para todos os equipamentos, e;
- ✓ Largura de banda do canal escalável.

LTE

LTE (**L**ong **T**erm **E**volution) constitui a última tecnologia *wireless* abordada para o mapeamento entre NSDL e NS-3 e principal alternativa à tecnologia WiMAX anteriormente descrita [Rohde, 2008][Erbele, 2011][Hamza, 2009][Bakharev, 2010].

Figura 4 - Logotipo LTE [3GPP, 2011]

Foi desenvolvida pela associação **3GPP** (**3**rd **G**eneration **P**artnership **P**roject) no final de 2008 e como o nome sugere, é uma tecnologia com actual designação de 3G. Um grande desenvolvimento nas suas especificações têm vindo a ser conseguido e diversas experiências têm sido realizadas no mundo real de forma a testar as potencialidades desta tecnologia. Ainda não possui a dimensão de utilização por parte das principais empresas de fornecimento de soluções tecnológicas como o WiMAX, mas já grandes nomes do mercado anunciaram a sua adopção num futuro próximo; Exemplos concretos de empresas que mencionaram a adopção do LTE são a AT&T, Verizon e Cisco.

Em relação à topologia típica desta tecnologia, é considerada a **E-UTRAN** (**E**volved-**U**niversal **T**errestrial **R**adio **A**ccess **N**etwork) em que se destacam a utilização dos seguintes dispositivos:

Enhanced-**N**ode**B** (**eNB**) – Nó com funcionamento análogo ao da *BaseStation* no contexto do WiMAX, em que tem como principal função o registo e provisionamento de banda larga para os vários UEs a si conectados para posterior encaminhamento de tráfego entre os UEs, e;

User Equipment (**UE**) – São os dispositivos que geram e/ou recebem tráfego e efectuam pedidos de registo a um eNB (por norma o geograficamente mais próximo), para que este receba e encaminhe os vários fluxos de dados para o respectivo destinatário.

Uma possível representação da arquitectura descrita anteriormente pode ser observada na Figura 5, em que a E-UTRAN inclui os dispositivos utilizados em LTE (eNBs e UEs) e a possibilidade de integração com redes de outras naturezas mas sempre no âmbito do **UMTS** (**U**niversal **M**obile **T**elecommunications **S**ystem).

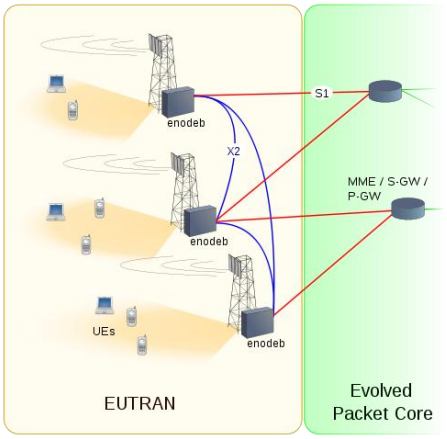

Figura 5 - Arquitectura típica de um ambiente LTE visando a abordagem E-UTRAN [Wikipedia, 2010]

À semelhança da forma como foram apresentadas as principais características que o WiMAX possui, as características principais do LTE são:

Camada Inferior (*Physical Layer*)

- A propagação do sinal suporta OFDM/OFDMA, MIMO, TDD e FDD;
- As velocidades máximas podem atingir os 172,8 Mbps (*downlink*) e 50,4 Mbps (*uplink*) a funcionar com frequência de 20 MHz e utilização de 2x2 MIMO, e;
- Os tipos de modulação de sinal passíveis de configuração são QPSK, 16QAM e 64QAM

Camada de *Link* (***MAC*** – ***M****edium* ***A****ccess* ***C****ontrol Layer*)

- Agendamento de tráfego *downlink* e *uplink* (no eNB)

Camada de Aplicação (*Application Layer*)

Os fluxos de dados são geridos por *bearers* existentes nos UEs que possibilitam associar a cada fluxo de dados atributos QoS que serão registados nos eNBs a que os UEs estão registados.

Actualmente esta tecnologia está a ser alvo de evolução de forma a respeitar os requisitos 4G mencionados na descrição do WiMAX; O lançamento de WiMAX *release* 2.0 e do LTE-Advanced (denominação atribuída à *release* 10 em desenvolvimento) está previsto para este ano de 2011.

De seguida serão apresentadas as ferramentas Web que dão suporte ao processo de mapeamento descrito no Capítulo 6.

2.3. TECNOLOGIAS DE DESENVOLVIMENTO WEB

No âmbito de desenvolvimento Web, muitas ferramentas possuem as especificações necessárias à concretização de este projecto. Entre as várias linguagens existentes, foram adoptadas as seguintes:

XML [XML, 2008] – Actualmente a tecnologia XML (E**x**tended **M**arkup **L**anguage) trata-se de uma linguagem amplamente utilizada, para as mais diversas finalidades. Entre as quais, as aplicações de XML mais frequentes são: Armazenamento de dados, transporte de informação, configuração de programas, etc.

Além de todas estas funções para o XML, este também serviu de ponte para a criação de outras linguagens de programação, neste caso em específico, o NSDL.

A criação de novas *tags* permite elaborar e estruturar a elaboração de diversos tipos de cenários sejam estes de natureza *IntServ*, *Diffserv*, MPLS, *wireless*, etc., tendo em conta os recursos disponibilizados pelo NSDL.

XSD [XSD, 2004] - A linguagem XSD (**X**ML **S**chema **D**efinition) trata-se de uma tecnologia que foi criada com o principal intuito de descrever a estrutura de um documento XML. Constitui também uma boa alternativa ao uso de DTDs (**D**ocument **T**ype **D**efinition) [DTD, 1999].

Porquê usar XSD em vez de DTD?

Tal como todas as linguagens de programação que possibilitam a estruturação de um documento XML, estas possuem vantagens e desvantagens associadas. Uma grande vantagem referente aos DTDs prende-se com o facto de que estas podem ser incluídas na declaração de um documento XML, ao contrário dos XSDs. Isto significa que só é necessário preocupar-se com um fluxo de informação, em vez de múltiplos [Chartier, 2006].

Por outro lado as XSDs permitem uma descrição dos seus elementos e atributos muito mais detalhada do que possa ser elaborado usando uma DTD. Por exemplo, é possível definir com maior rigor que valores ou conjuntos de caracteres podem ser atribuídos a determinados campos existentes no documento XML.

Esta vantagem confirma a utilização de um XSD ao invés de uma DTD devido sobretudo à grande quantidade de objectos que a linguagem NSDL possui no seu mundo léxico, bem como uma grande quantidade de atributos que para a simulação de cenários, pretende-se que estes sejam validados de forma a possuírem valores passíveis de uma simulação credível.

XSLT [XSLT, 1999] - A sigla significa e**X**tensible **St**ylesheet **L**anguage **T**ransformation e corresponde a uma linguagem baseada em XML que permite transformar documentos com sintaxe XML em um documento com o *output* pretendido conforme as *tags* que estão contidas num ficheiro XML. Esta tecnologia, detém um papel fundamental para o mapeamento descrito no Capítulo 6.

PHP [PHP, 2011] - **H**ypertext **P**rocessor (PHP), tal como o XML, constitui outra tecnologia amplamente utilizada no mundo da Web, tendo como aplicações do seu uso mais comuns as de processamento de informação (tal como o próprio nome o sugere), comunicação com bases de dados, permite manuseamento de variáveis dinâmicas e a passagem de estes argumentos entre ficheiros PHP através das funções "GET" e "POST".

2.4. TRABALHOS RELACIONADOS

Os trabalhos relacionados com o projecto desenvolvido ao longo desta tese de mestrado são essencialmente ferramentas que têm sido desenvolvidas no âmbito da *Framework* NSDL, um conjunto de ferramentas gráficas e textuais que permitem modelar cenários de redes para posterior simulação em simuladores de redes já existentes (A *Framework* NSDL será apresentada no Capítulo 4). Segue-se uma breve descrição de cada um destes trabalhos.

NSDL *perfil* NS-2 [Marques, 2010] – Este trabalho foi desenvolvido pelo Prof. Eduardo Marques e inclui a criação do conceito da NSDL *Framework* bem como um *perfil* que inclui um conjunto de objectos básicos de redes passíveis de criação de uma estrutura NSDL; A noção de *perfil* será mais aprofundada no Capítulo 4.

O *perfil* desenvolvido para a plataforma de simulação de redes NS-2 possui um conjunto de objectos que podem ser modelados dando suporte a diferentes tecnologias nomeadamente IntServ, DiffServ, métricas QoS e MPLS.

VNS [Plácido, 2010] – *Visual Network Simulator*, desenvolvido pelo colega Ricardo Plácido na Universidade da Madeira no âmbito da conclusão do seu Mestrado em Telecomunicações e Redes.

Esta ferramenta permite modelar graficamente uma rede com recurso a objectos essencialmente característicos do perfil NS-2 para NSDL; A ferramenta permite também guardar um projecto para posterior edição. Uma funcionalidade em destaque é a de possibilitar a criação de um script TCL com base na rede modelada para simulação no ambiente NS-2.

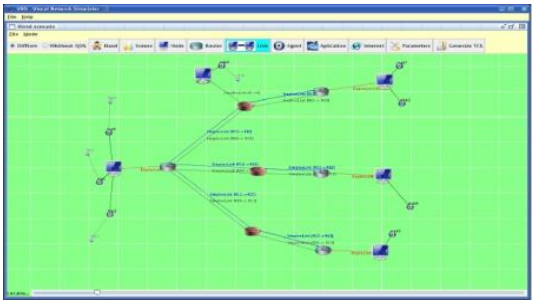

Figura 6 - Interface gráfica do VNS

Através da Figura 6 é possível verificar um cenário de redes modelado no VNS que inclui Hosts, Routers e aplicações associadas a cada Host, denotando a quantidade de fluxos de entrada/saída ocorrem em cada nó.

VND [Azevedo, 2010] – *Visual Network Descriptor* foi o projecto desenvolvido pelo colega Jesuíno Azevedo para a conclusão do programa de Mestrado em Engenharia de Informática na Universidade da Madeira. Esta ferramenta é uma GUI (*Graphic User Interface*) baseada em FLEX da Adobe [Flex, 2011] e encontra-se disponível através do *link* da referência, para modelação de cenários de redes.

Através do VND é possível configurar um ficheiro *input* XML que contém que objectos e respectivos atributos poderão ser utilizados na modelação dos cenários.

Inicialmente os elementos, objectos e atributos previstos visavam a cobertura do perfil NS-2 para NSDL, no entanto como foi mencionado este ficheiro pode ser alterado pelo próprio utilizador de modo a que este possa incluir os objectos que necessitam ser modelados na sua rede. Após modelação de um determinado cenário, o VND permite exportar esse cenário gráfico para uma estrutura NSDL.

As Figuras dos cenários 1 a 4 presentes no Capítulo 7 (Estudo de Caso) são exemplos de cenários que podem ser modelados com auxílio do VND, tendo por exemplo a possibilidade de utilizar dispositivos como *switches*, *hosts*, *routers*, cabos de fibra e *Ethernet*, etc.

Virtualização Automática de Cenários de Rede [Araújo, 2011] – Este projecto encontra-se em desenvolvimento e tem como objectivo simular cenários NSDL numa ferramenta de virtualização (*XenServer* [Citrix, 2011]), permitindo ao utilizador simular cenários num ambiente virtual a emular um ambiente real. Os cenários NSDL são traduzidos em *scripts batch* e executados no *XenServer* tornando assim automática a configuração do cenário.

Esta configuração permite a especificação de definições de interfaces, protocolos de endereçamento (IPv4/IPv6), criação de aplicações geradoras de tráfego, protocolos de encaminhamento tais como OSPF, etc. Entre os tipos de máquinas que estão a ser testadas e configuradas para o efeito, incluem-se os sistemas operativos *Vyatta* [Vyatta, 2011] e *OpenSUSE v.11* [OpenSUSE, 2011].

Destes trabalhos apresentados, realizados no âmbito da *Framework* NSDL, todos partilham a abordagem à concepção de cenários de redes: Modelação dos cenários, criação de *script* NSDL e por fim a execução num ambiente de simulação de redes já existente. Como ilustrado na Figura 1, todas as ferramentas discutidas anteriormente estão relacionadas dado o seu grau de acoplamento proporcionado através da Framework NSDL. Como podemos observar, essas ferramentas possuem funcionalidades complementares já que elas proporcionam a autoria gráfica (VND e VNS), a simulação (NS-2 e NS-3) e a virtualização de cenários de redes. É importante o leitor notar, que esse acoplamento não tem sido demonstrado na literatura por outras abordagens existentes.

Por exemplo, consideremos **XBNSDL** (***X**ML **b**ased **N**etwork **S**imulation **D**escription **L**anguage*) [Canoico, 2003] que constitui uma linguagem baseada em XML para definição de cenários de redes tal como o NSDL. Tal como a abordagem realizada ao mapeamento NSDL para NS-2/NS-3, esta linguagem utiliza também as tecnologias XML, XSLT e XSD para a autoria, transformação e validação de um documento XBNSDL, respectivamente; A transformação das estruturas XML é realizada para a linguagem OTcl para futura simulação em NS-2. Os principais esforços no desenvolver desta linguagem prenderam-se com as necessidades de interoperabilidade, o suporte à especificação de diversos tipos de cenários de redes, validação de modelos analíticos e scripts de configuração de redes.

Comparativamente ao NSDL, o XBNSDL apresenta mais semelhanças que diferenças:

- É orientado à interoperabilidade e extensibilidade;
- Utiliza as variantes XSD e XSLT para o processo de mapeamento entre XML e OTcl, e;
- Teve por alvo inicial a plataforma NS-2.

Já em termos de diferenças, especial destaque na nomenclatura utilizada que difere do NS-3 utilizando nomes muito técnicos que para utilizadores sem grandes conhecimentos de redes possa traduzir-se na necessidade de um maior esforço de adaptação.

No entanto, outros modelos, abordagens e ferramentas têm sido propostas de modo a optimizar e explorar as vantagens destas tecnologias através da simulação [Piro, 2010][Erbele, 2011][Ball, 2011]. Algumas das soluções existentes para o desenho e implementação de tecnologias como o WiMAX e LTE realizam uma abordagem para problemas e casos específicos, limitados para alguns domínios específicos e, em muitas das vezes, documentação em falta ou incompleta. É importante considerar as funcionalidades fornecidas por estas soluções. Muitas delas são muito limitadas em termos de número de funcionalidades disponíveis e têm uma aplicação demasiado específica. Outras são baseadas em grandes conjuntos de modelos e fornecem uma análise mais complexa sobre uma rede.

Se todas estas soluções pudessem operar de modo coordenado, podiam disponibilizar um ambiente sólido e útil para a optimização da gestão de redes WiMAX/LTE. Apesar disto, os formatos dos dados usados por estas soluções existentes são muito distintas e, na maioria das vezes, incompatíveis.

De modo a fornecer uma solução genérica para promover a interoperabilidade entre as várias soluções existentes, este projecto apresenta a *Framework* NSDL existente que com auxílio à linguagem NSDL conjuntamente com os seus perfis, representa uma solução viável para auxiliar os gestores de redes na optimização dos seus cenários durante o seu ciclo de vida.

2.5. CONCLUSÃO

Neste capítulo foram abordados os principais conceitos considerados ao longo do desenvolvimento do projecto.

As principais tecnologias abordadas neste projecto foram brevemente descritas relativamente aos *standards* que as definem no contexto de redes.

Foram discutidas as principais plataformas que permitem a autoria e simulação de cenários de redes com recurso aos trabalhos relacionados e ferramentas de simulação de redes, entre as quais o NS-3 que será apresentado no capítulo seguinte.

3. NETWORK SIMULATOR 3 (NS-3)

Neste capítulo é apresentado o simulador de redes Network Simulator 3 (NS-3) que constitui uma das ferramentas de simulação amplamente utilizada em todo o mundo e com grandes perspectivas de crescimento, contando com diversos colaboradores que trabalham exaustivamente para abordar as tecnologias emergentes.

3.1. INTRODUÇÃO

Antes de optar por uma determinada tecnologia para a instalação de uma rede, a simulação da mesma numa ferramenta própria, constitui uma necessidade de grande importância.

Porquê simular? – Suporte ao projecto, optimização de redes, apoio na decisão de escolha de equipamentos/tecnologias, etc.

Sendo assim, um estudo prévio sobre as várias possibilidades de implementação é fundamental. Antes de iniciar a concretização de um projecto de rede, é recomendável simular a rede em questão. Efectuar este processo permite visualizar o funcionamento da rede em termos de tráfego e valores alusivos ao seu desempenho.

A simulação prévia pode assim evitar despesas desnecessárias que resultam muitas vezes de projectos inviáveis que poderiam ter sido feitas outras opções que viabilizariam o projecto caso o seu planeamento fosse mais detalhada e se baseasse também na simulação da rede em questão.

Ao longo deste Capítulo é descrito o NS-3, nomeadamente em termos de funcionalidades, constituição modular, requisitos de instalação e por fim uma pequena análise comparativa com outras ferramentas de simulação de redes.

3.2. DESCRIÇÃO GERAL

O Network Simulator 3 (NS-3) constitui uma aplicação que permite simular diversos cenários de redes, desde os mais simples aos mais complexos, usando para tal tempos minimamente realísticos associados a eventos de diversas naturezas (por exemplo ping).

A sua origem deu-se em meados de 2006 (University of Washington - Tom Henderson e Craig Dowell; INRIA, Sophia Antipolis – Mathieu Lacage; Georgia Tech University, Atlanta – George Riley e Raj Bhattacharjea [Carneiro, 2010]) e encontra-se actualmente em constante desenvolvimento, quer a nível das suas capacidades de simulação, quer a nível de ferramentas de terceiros que permitem o estudo mais detalhado sobre os cenários de redes, como são exemplos as ferramentas de animação gráfica *nam* e ns-3-pyviz, entre outras.

Trata-se do sucessor do Network Simulator 2 (NS-2), mas no entanto não representa propriamente uma evolução do mesmo.

Os seus módulos foram desenvolvidos em C++ e apresentam algumas alterações relativamente ao NS-2, incluindo autoria de cenários de redes com auxílio opcional a *binds* de Python, ao contrário de NS-2 que foi desenvolvido em C++ mas em termos de simulação recorre às capacidades da linguagem OTcl que constitui uma extensão da linguagem Tcl.

3.2.1. PORQUÊ NS-3?

A plataforma anterior ao NS-3 (NS-2) detinha (e ainda detém) grande número de utilizadores. Ao longo da sua história foi verificado um crescente número de utilizadores, mesmo após ser descontinuado .

Na Tabela 3 podemos verificar a distribuição parcial entre as principais plataformas usadas pelos criadores de cenários de redes para simulação. Esta distrubuição está distribuída por que camadas do modelo OSI são alvo de simulação.

Tabela 3 - Plataformas para simulação de cenários de redes e distribuição das pesquisas pelas mesmas

	ns-2	OPNET	QualNet/Glomosim
$\geq$ layer 4	123 (75%)	30 (18%)	11 (7%)
= layer 3	186 (70%)	48 (18%)	31 (12%)
$\leq$ layer 2	114 (43%)	96 (36%)	55 (21%)

Embora seja visível popularidade do Network Simulator 2 (NS-2), existem vários problemas adjacentes que não puderam ser resolvidos com uma simples actualização da aplicação. Entre os vários problemas, destacam-se os seguintes:

- Utilização dual de linguagens de programação para autoria de cenários de redes (OTcl e C++);
- Escalabilidade comprometida;
- Dificuldade acrescida para iniciantes, nomeadamente estudantes;
- Agregação crescente de módulos com falta de manutenção/actualização bem como incompatibilidades entre os mesmos;
- Falta de documentação e a existente encontra-se desactualizada, e;
- Falta de correspondência entre o cenário utilizado numa simulação e o comportamento do mesmo aplicado a uma rede real.

Por estas e outras razões o desenvolvimento de uma nova ferramenta constituiu a melhor forma de ultrapassá-las. Ao longo deste documento é demonstrado como o NS-3 veio solucionar na totalidade ou em parte vários dos problemas mencionados anteriormente e outros.

3.3. MÓDULOS

Como é possível visualizar na Figura 7, a organização modular detém dependências em que os módulos de topo dependem dos módulos inferiores.

A implementação destas camadas encontra-se quase toda na pasta *src* (core em *src/core*, Simulator em *src/simulator*, e assim sucessivamente). De seguida será apresentada uma descrição geral de cada camada ou sub-módulo de camada de forma a percepcionar melhor as principais características e responsabilidades de cada componente/módulo.

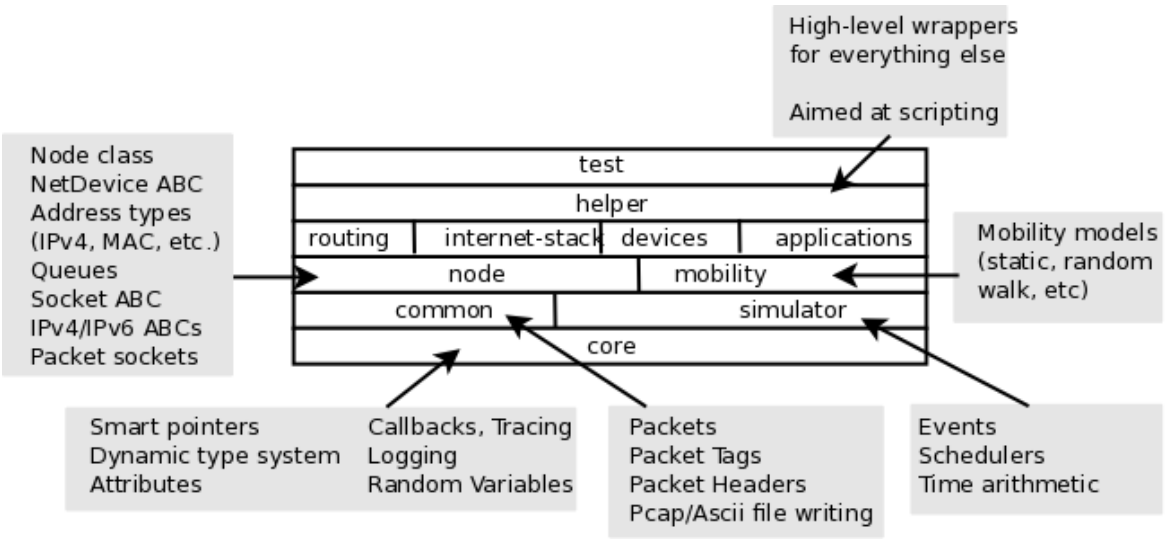

Figura 7 - Módulos NS3 [Nsnam, 2011]

Segue-se uma pequena descrição sobre os módulos que constituem a implementação de NS-3 e que se destacam comparativamente ao NS-2.

3.3.1. CORE

Este módulo constitui o módulo base, em que se encontram as diversas variáveis globais do sistema, variáveis que possibilitam o controlo do tráfego, listas de atributos, etc. Possui suporte à depuração dos cenários de redes com auxílio a funções como ***assert***, ***fatal error handler*** e ***logging***.

O core permite também a utilização de "*smart pointers*" o que optimiza de certa forma a utilização e gestão de memória.

Detém o conjunto de classes que podem ser utilizadas para "*trace*" de forma a recolher dados estatísticos das mesmas.

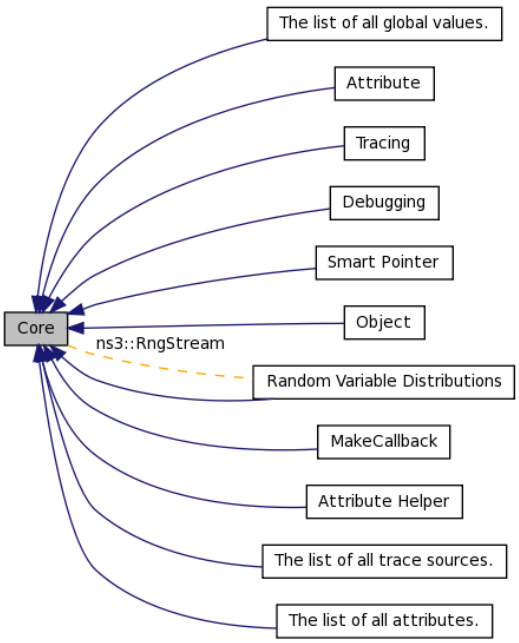

Figura 8 - Módulo Core

3.3.2. SIMULATOR

Estão descritos neste módulo especificações para a simulação dos cenários passíveis de modelação por parte do NS-3. Com as *tags* ***time*** e ***scheduler*** é possível definir tempos de início e fim para as simulações, bem como o agendamento de determinados eventos, respectivamente com a utilização de directivas de 128 bits que permitem uma maior precisão para os tempos estabelecidos [Carneiro, 2010].

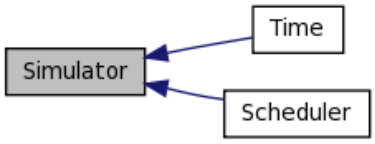

Figura 9 - Módulo Simulator

3.3.3. COMMON

Este módulo contém as definições necessárias à especificação dos pacotes que irão ser transferidos na rede elaborada. Para tal, é possível criar o pacote (***packet***), associar uma taxa de transmissão de dados (***data rate***) e agrupar pacotes consoante as suas características através do uso de ***tags*** que descriminem um determinado grupo de pacotes.

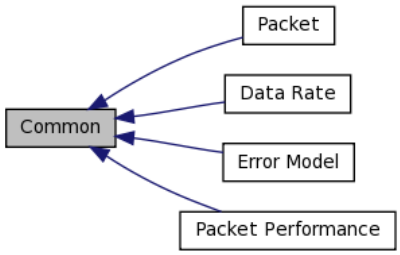

Figura 10 - Módulo Common

3.3.4. MOBILITY

O módulo ***mobility,*** por sua vez, contém as declarações necessárias a toda a abordagem de movimento dos nós na rede passível de simulação. Serve essencialmente para monitorizar todos os objectos existentes na rede em termos de posição actual e velocidade.

3.3.5. NODE

Com este módulo é possível declarar uma abstracção de ***node*** que posteriormente pode ser especificado para um outro componente de rede física em concreto (por exemplo, computador, servidor, etc.). Com a sua directiva *address* é possível estipular um endereço IP para o nó, bem como associar protocolos (***IPv4RoutingProtocol*** ou ***IPv6RoutingProtocol***) e aplicações (***Sockets*** e auxílio ao módulo ***Applications***).

A classe ***node***, uma vez instanciada, irá guardar numa lista virtual todos os nós (***NodeList***) que irão ser utilizados na simulação e também uma lista de canais criados entre os mesmos (***ChannelList***).

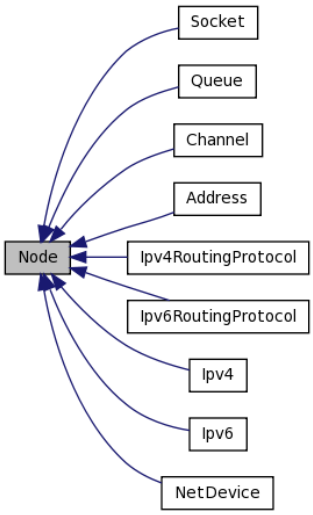

Figura 11 - Módulo Node

Com a actualização modular realizada em 25 de Maio de 2011, alguns destes objectos associados a *Node* passaram a ser considerados no âmbito de um novo módulo denominado por *Network*, como é o caso de IPv4 e IPv6.

3.3.6. APPLICATIONS

O módulo ***Applications***, tal como o seu próprio nome indica, permite a instanciação de aplicações associadas a eventuais nós pelos seus canais previamente declarados. De salientar a utilização de ***OnOffApplication*** que permite estabelecer tempos em que a aplicação em causa deve estar activa criando um tráfego CBR (***onTime***) e os tempos em que a mesma não deverá gerar qualquer tráfego (***offTime***).

Outra directiva importante deste módulo é a de ***PacketSink*** em que a ideia original foi a de complementar a declaração de ***OnOffApplication*** mas devido a ser de um nível de abstracção maior, então foi considerado separadamente: ***PacketSink*** poderá estar associado a aplicações unicast e multicast.

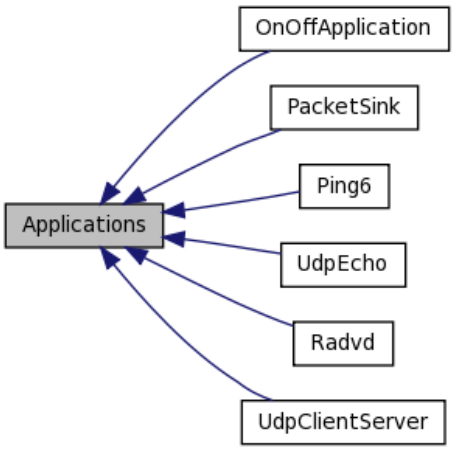

Figura 12 - Módulo Applications

3.3.7. DEVICES

Neste módulo está contida a especificação referente aos dispositivos que podem ser utilizados na autoria de cenários de redes numa óptica de topologia/tecnologia de rede. Temos disponíveis por exemplo a topologia ***Mesh***, ***Point-To-Point***, ***Wifi***, ***CSMA***, ***UAN***, ***WiMAX***, etc.

Posteriormente os vários tipos de *devices* foram considerados individualmente devido à crescente contribuição com novos tipos de dispositivos como é exemplo dos módulos ***LTE***, ***Spectrum***, ***OpenFlow Switch***, etc.

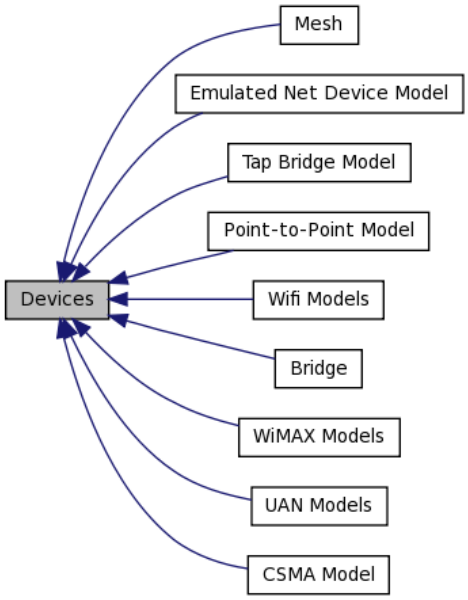

Figura 13 - Módulo Devices

3.3.8. INTERNET-STACK

Este módulo contém os principais protocolos utilizados na transmissão de dados, usados sobretudo por aplicações como ***ftp***, ***cbr***, ***telnet***, etc. Entre os vários protocolos a que o módulo cobre a sua especificação, destacam-se os seguintes, ***TCP***, ***UDP*** e ***ARP***.

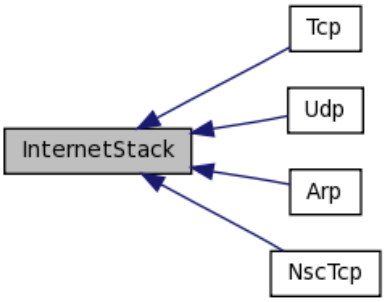

Figura 14 - Módulo InternetStack

3.3.9. ROUTING

Outro módulo detentor de grande importância em termos de autoria de cenários de redes trata-se de o módulo de ***routing*** que possui especificações necessárias para a definição de rotas para envio e recepção de tráfego.

Para rotas estáticas é possível utilizar *tags* como Ipv4StaticRouting, Ipv6StaticRouting. Para redes mais complexas ou com maior dimensão podemos utilizar encaminhamento com *tags* como ***Global routing***, ***AODV*** (***A****d hoc* ***O****n-Demand* ***D****istance* ***V****ector routing*) [Moraes, 2003], ***OLSR*** (***O****ptimized* ***L****ink* ***S****tate* ***R****outing*) [OLSR, 2008] e ***Nix-vector routing***, este último somente disponível (de momento) para Ipv4.

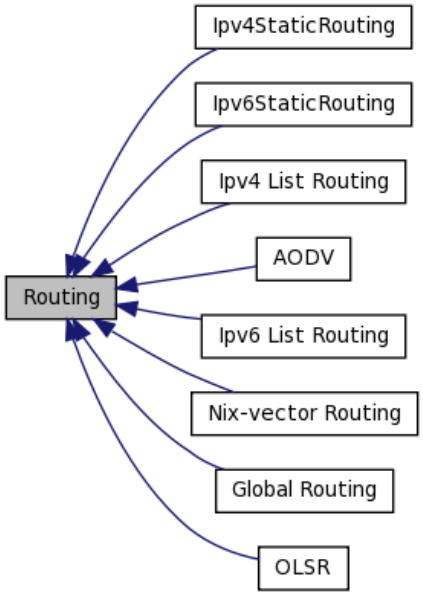

Figura 15 - Módulo Routing

Click *routing* [Suresh, 2011] e ***DSDV*** *routing* foram outras adições que levaram a que o módulo *routing* fosse, tal como o módulo *devices*, subdividido em vários módulos individuais para cada tipo de encaminhamento.

3.3.10. HELPER

Já o módulo ***Helper*** possui ajuda nomeadamente para fins de *scripting* para que o autor de cenários possa criá-los facilmente, usando para tal informações referentes aos vários módulos anteriormente apresentados.

3.3.11. TEST

Por fim, o módulo *test* contém scripts para testar o cenário desenvolvido, em termos de:

- ***Build verification test*** (verificação da construção do programa final para execução);
- ***Unit tests*** (usados para testar funcionalidades/blocos de código específicos); System tests (testes que envolvem mais do que 1 dos módulos anteriormente descritos);
- ***Examples*** (nada realmente é testado, no entanto trata-se de uma forma de garantir que o *build* to programa foi realizado com sucesso);
- ***Performance tests*** (testes para verificar se os tempos de execução de funcionalidades ou até mesmo do script na sua totalidade são aceitáveis ou não).

3.4. PRINCIPAIS CARACTERÍSTICAS/FUNCIONALIDADES

De entre as várias funcionalidades suportadas pelo NS3, destacam-se as seguintes:

Escalabilidade

Os pacotes gerados para criação de tráfego na rede podem possuir tamanho "virtual de zero bytes", o que permite obter uma maior facilidade em termos de gestão de memória visto que estes não são alocados na memória física;

Poder representativo

Nem sempre é necessário estipular coordenadas para os nós existentes na rede alvo, pelo que torna muito mais simples a codificação destes, ao contrário de OTcl que exige um posicionamento relativo ou absoluto entre os objectos para que estes não sejam sobrepostos entre si – Por exemplo, os nós que usam canais físicos entre eles não necessitam de especificação de coordenadas;

Programação orientada a objectos

A codificação para autoria de cenários de redes segue a metodologia POO, pelo que optimiza a extensibilidade e gestão da memória utilizada.

Esta boa prática de programação permite também uma fácil agregação de objectos existentes na rede a simular – Por exemplo, agregar *netdevices* aos nós e por sua vez canais, protocolos e eventuais aplicações associadas;

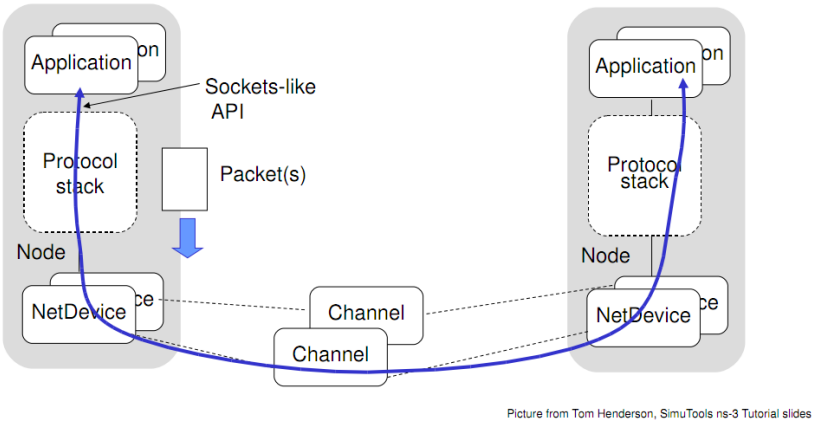

Figura 16 - Arquitectura IP em NS3 [Henderson, 2011]

Facilidades de modificação

A sua estrutura baseada em programação orientada a objectos permite que outras funcionalidades/módulos novos possam ser adicionados no futuro sem que para esse fim seja necessário alterar a estrutura da aplicação – por exemplo, a adição de modelos de gestão de energia;

Funcionalidades entre camadas

Tendo em conta o modelo OSI, a informação relativa aos pacotes existentes na rede pode ser transmitida entre as várias camadas do modelo através de *tags* que contêm informação sobre os pacotes. Esta atribuição de *tags* aos pacotes permite simultaneamente o controlo do tráfego entre as várias camadas.

Integração de funcionalidades em tempo real

Na execução das redes criadas é possível adicionar a informação existente sobre os pacotes transmitidos na rede para ficheiros do tipo PCAP, o que possibilita a futura utilização destes por plataformas terceiras (por exemplo *Wireshark*);

É possível integrar agendamento em tempo real para a simulação em que os eventos são sincronizados com um relógio;

Também em tempo real é suportada a utilização da pilha do Linux Kernel TCP/IP (com auxílio às variantes Linux 2.6.18 e Linux 2.6.26).

Desempenho

O desempenho do NS-3 traz uma grande melhoria no processamento relativamente ao NS-2 devido essencialmente à utilização de uma arquitectura *multicore*

que possibilita o processamento paralelo através da utilização do **MPI** (***M****essage* ***P****assing* ***I****nterface*) que se constitui um protocolo usado no âmbito das aplicações distribuídas.

3.5. VANTAGENS E DESVANTAGENS

De entre as várias vantagens associadas ao uso de NS-3 para autoria de cenários de redes, destacam-se as seguintes:

- Ferramenta *opensource* – Ao contrário de ferramentas utilizadas para autoria e simulação tais como OPNET e QualNet;
- Autoria dos cenários é baseada na metodologia Object-Oriented – Promove possível extensibilidade e facilidades de modificação;
- Actualização dos módulos que constituem a sua arquitectura – Módulos tais como *core* e *node* sofreram grandes alterações de modo a realçar o realismo dos cenários criados, e;
- Existência de grande documentação referente aos vários módulos do programa bem como uma grande comunidade de colaboradores que asseguram um desenvolvimento contínuo da ferramenta.

Já em termos de desvantagens, podemos destacar a seguinte:

- Para utilizadores de NS-2, a utilização do NS-3 exige uma familiarização com a linguagem C++ bem como algum conhecimento sobre programação orientada a objectos, e;
- Ainda se encontra num estado inicial de desenvolvimento pelo que ainda não cobre um vasto leque de tecnologias, comparativamente a outros simuladores de redes.

3.6. REQUISITOS DE INSTALAÇÃO

Para instalar o Network Simulator 3, é necessário que um conjunto de requisitos seja respeitado. Segue-se uma lista dos requisitos para instalação e execução do NS-3 [Ns3Install, 2011].

✓ O sistema operativo (OS) em que o *software* deverá ser instalado deve ser um dos seguintes:

- Ubuntu/Debian
- Fedora/RedHat
- Gentoo
- Mac OS X (Snow Leopard)

- ✓ Para o núcleo do NS-3 (ns-3 core) é necessária a pré-instalação do gcc/g++ versão 3.4 ou superior
- ✓ Python 2.4 ou superior

Ainda relativamente ao sistema operativo alvo de instalação, é necessário ter em conta se o funcionamento do *software* irá ser apropriado ao que se pretende realizar com o mesmo. Atendendo à Tabela 4, podemos visualizar que funcionalidades são possíveis executar com cada uma das variantes.

Tabela 4 - Variantes e funcionalidades suportadas [Ns3Install, 2011]

Option status

Option	Linux gcc-4.x,gcc-3.4.x	OS X	Cygwin
Optimized build			X [1]
Python bindings			X [2]
Threading			
Real-time simulator		X	X
Emulated Net Device		X	X
Tap Bridge		X	X
Network simulation cradle	note[3]	X	X
Static builds			

Key: **( )**(empty space) = supported; **X** = not supported; **?** = unknown; **dev** = support in ns-3-dev (next release)

Notas:

1. Funciona com gcc4
2. *Cygwin* possui esta limitação
3. NSC funciona bem com as versões de gcc 3.4, 4.2 ou superior. Devem ser evitadas versões compreendidas entre as *releases* de 4.0 e 4.1. devido a problemas de compatibilidade verificados com estas versões.

Para todos os efeitos, uma máquina que tenha por base o Linux é a ideal para a instalação do NS-3.

3.7. CONCLUSÃO

Neste capítulo foi realizada uma apresentação do simulador de redes NS-3. Para tal, foram apresentadas as suas principais características e funcionalidades, a sua constituição modular, requisitos de instalação e uma pequena apreciação de vantagens e desvantagens que a ferramenta apresenta actualmente.

No capítulo seguinte será apresentada a *Framework* NSDL na qual o NS-3 faz parte como plataforma destino no âmbito da autoria de cenários de redes.

4. NSDL (FRAMEWORK)

Neste capítulo é apresentada a Framework ***N**etwork **S**cenario **D**escription **L**anguage (**NSDL**)*, proposta e desenvolvida no âmbito do trabalho de doutoramento do Dr. Eduardo Marques na Universidade da Madeira, de forma a proporcionar a interoperabilidade entre diferentes ferramentas de gestão e simulação de redes, e em consequência a optimização da autoria de cenários de redes, desde a sua especificação até à sua simulação.

4.1. INTRODUÇÃO

Cada vez mais surgem novas ferramentas relacionadas com a autoria de cenários de redes e poucas apresentam interoperabilidade entre si. Com a finalidade de diminuir este problema, o NSDL tem por principais objectivo criar pontes entre ferramentas de concepção e monitorização de cenários de redes (especificação textual e/ou gráfica (GUIs) como é o exemplo do VNS e VND) e ferramentas de simulação (NS-2, NS-3, etc.).

De seguida serão apresentadas as principais características do NSDL que permitem confirmar os objectivos previamente mencionados.

4.2. CONCEITOS BÁSICOS

A definição da linguagem NSDL tem por base a linguagem XML que apresenta atributos de qualidade favoráveis às finalidades do NSDL: Entre estes atributos destacam-se a (1) simplicidade, (2) extensibilidade e (3) grande poder de abstracção:

(1) – Uma especificação NSDL é de fácil compreensão pois consiste numa estrutura XML organizada e semanticamente válida, quando seguidas as boas práticas de programação (i.e. indentação);

(2) – O NSDL permite a introdução de extensões que visam abordar a especificação de objectos existentes e/ou até de objectos novos no contexto de uma determinada plataforma de simulação final. O NSDL como extensão do XML detém um conjunto de *tags* que formam o mundo lexical de *tags* permitidas na validação NSDL. Este conjunto de *tags* é incluído no *perfil* base pois são utilizadas para especificar elementos que são comuns à maioria das ferramentas de simulação de cenários de redes. O conceito de *perfil* foi introduzido para as eventuais extensões ao NSDL base propostas, como é o exemplo do trabalho realizado no âmbito desta tese (i.e. *perfil* NS-3) que será descrito no capítulo 5. Uma extensão da linguagem NSDL inclui o *perfil* base e um conjunto de *tags novas* que permitem especificar objectos que existam exclusivamente no contexto da plataforma destino (NS-3 por exemplo), e;

(3) – Com a utilização do NSDL é possível especificar os vários elementos existentes em um determinado *perfil* com um alto nível de abstracção. Por outro lado, permite também detalhar exaustivamente um certo objecto, especificando para tal os diversos atributos que o perfil prevê para o objecto em causa.

A extensibilidade mencionada anteriormente também leva ao surgimento de um outro atributo muito importante no âmbito da autoria de cenários de redes: A interoperabilidade.

Quando dois *perfis* possuem elementos de rede coincidentes, é possível obter facilmente dois *scripts* para serem utilizados nas ferramentas relativas aos *perfis* através de uma única especificação NSDL: Além de ser uma forma rápida e optimizada de poder simular o mesmo cenário de rede em diversas plataformas finais, permite também que o utilizador não tenha necessariamente grandes conhecimentos de programação nomeadamente as linguagens utilizadas nas ferramentas de simulação de redes, como é o exemplo de OTcl para NS-2, C++ para NS-3, especificação *batch* para ambientes Linux, etc., pois o utilizador somente necessita programar XML.

De seguida será apresentada a arquitectura que reflecte a abordagem a todos estes atributos de qualidade referidos.

Uma especificação NSDL permite caracterizar diversos objectos existentes numa rede, em grupos contextuais diferentes mas que complementam a robustez de um determinado cenário de redes: Estes grupos são essencialmente rede, visualização e simulação.

4.3. ARQUITECTURA

A arquitectura da *Framework* NSDL é constituída essencialmente por 3 camadas (Figura 17):

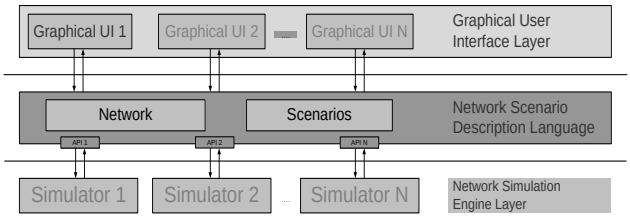

Figura 17 - Arquitectura da Framework NSDL por camadas

Camada superior (*Graphical User Interface Layer*) – Inclui as diversas ferramentas de autoria gráfica, monitorização e visualização de cenários de redes. Estas ferramentas, uma vez compatíveis com o NSDL, podem ser utilizadas para importação e exportação de uma estrutura NSDL adequada; A exportação visa a criação de uma estrutura NSDL consoante os objectos graficamente representados na ferramenta. Por

outro lado, a importação de um ficheiro NSDL tem por finalidade poder observar graficamente os vários objectos especificados no ficheiro fornecido e eventualmente possibilitar a sua edição. São exemplos de ferramentas gráficas no âmbito da *Framework* NSDL, o VNS e o VND, ambas apresentadas no Capítulo 2. De momento o VND aborda a exportação de um cenário gráfico para um ficheiro NSDL mas o processo de importação também é algo que está a ser considerado como uma futura contribuição. Quanto ao VNS, esta ferramenta permite modelar cenários de redes com o objectivo de obter um script passível de execução no ambiente de simulação NS-2.

Camada intermédia (*Network Scenario Description Language*) – Esta camada é detentora da linguagem NSDL que serve como uma *ponte* entre as ferramentas gráficas e as ferramentas de simulação, permitindo assim uma optimização do processo de autoria dos cenários de redes. Esta linguagem é constituída essencialmente por 2 elementos: *Network* e *Scenarios* que irão ser descritos com maior detalhe na secção 4.4. Nesta camada são incluídas ainda metodologias de validação e de transformação das estruturas NSDL (maior detalhe no Capítulo 6) que permitem obter *scripts* passíveis de execução em uma das ferramentas de simulação incluídas na camada inferior.

Camada inferior (*Network Simulation Engine Layer)* – Por último, a camada inferior aglomera as ferramentas existentes de simulação, detentoras de um *perfil* NSDL para que possa existir então uma ligação entre esta camada e a intermédia. Destacam-se essencialmente o NS-2 e o NS-3 por serem os simuladores que actualmente detêm um perfil para NSDL. Como fora mencionado, outras ferramentas poderão eventualmente ser alvo de integração nesta *Framework* NSDL, bastando para tal a criação de um perfil que aborde os objectos que a ferramenta permite simular e os seus respectivos atributos. Como exemplo de contribuições realizadas neste âmbito por parte de colegas da Universidade da Madeira temos por exemplo o trabalho que está a ser realizado pela colega Joana Araújo que consiste em incorporar um perfil de virtualização na linguagem NSDL que permita simular uma rede virtual com auxílio a máquinas virtuais como o *XenServ* e o *Vyatta*.

De seguida será apresentada a estrutura detalhada no NSDL incluído na camada intermédia.

4.4. ESTRUTURA NSDL

Como foi mencionado na secção 4.3, uma estrutura NSDL é constituída por 2 elementos: *Network* e *Scenarios*.

O elemento *Network* inclui todos os objectos que fazem parte de um cenário de rede, sejam estes equipamentos activos (*computers*, *routers*, *stations*, etc.) sejam passivos (*links*, *switches*, *bridges*, etc.). Já o elemento *Scenarios* é responsável pela definição do comportamento dinâmico da rede (com o sub-elemento *Simulations*) bem como a definição da posição espacial dos objectos incluídos na rede (com o sub-

elemento *Visualizations*).

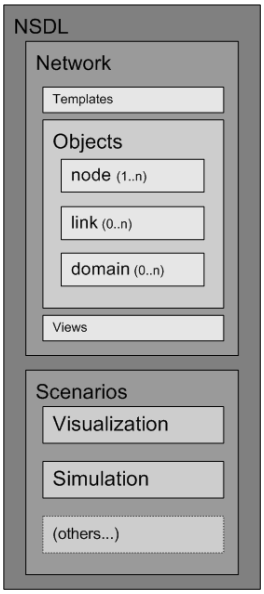

Figura 18 - Estrutura NSDL

Na Figura 18 é ilustrada a estrutura típica de um documento NSDL. Podemos constatar que o elemento *Network* é composto por *Templates*, *Objects* e *Views*. Os *Templates* são de uso opcional e possibilitam especificar objectos de rede de uma forma muito mais fácil e flexível bastando para o efeito referenciar o *template* que deve ser utilizado para enriquecer a especificação de um determinado objecto.

Os *Objects* incluem equipamentos passivos/activos e domínios. Ainda na Figura 18 podemos observar os elementos de maior nível de abstracção (*node*, *link* e *domain*) e que fazem parte do perfil base (Figura 19).

As *Views* são utilizadas para agrupar um conjunto de objectos de rede. Esta funcionalidade é útil quando é apenas pretendido destacar um conjunto específico de objectos de uma rede e/ou alterar uma definição que todos partilham na sua especificação.

O elemento *Scenarios* inclui ainda dois sub-elementos denominados por *Simulations* e *Visualizations,* em que o primeiro detém especificações que irão determinar o comportamento da simulação do cenário em "*run-time*" (tempo de execução) tais como os tempos de início e fim da simulação, versão do simulador, ficheiros de output de dados, etc. Enquanto o segundo é responsável pela definição da localização espacial dos objectos existentes na rede, o que pode ser muito útil na utilização deste documento numa ferramenta gráfica.

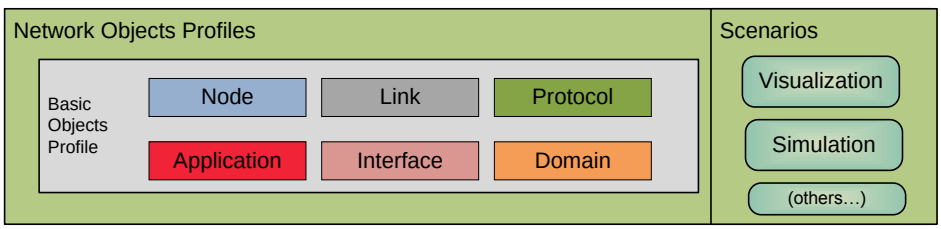

Figura 19 - Perfil base do NSDL

De seguida serão apresentados os principais objectos que constituem os elementos de *Network* e de *Scenarios* incluídos na Figura 18.

4.4.1. NETWORK

No elemento *Network* são descritos todos os objectos de rede (mais exactamente no elemento *Objects*) o que faz com que este elemento detenha maior importância comparativamente ao elemento *Scenarios*. De entre os principais objectos, destacam-se os seguintes:

Node – Trata-se do tipo de nó com maior nível de abstracção, conhecendo como principais especificações os *computers*, *routers*, *switches*, entre outros;

Link – Ainda no âmbito da definição física da rede, o elemento Link constitui uma ligação simples entre dois nós (*nodes*) e que suporta diversas tecnologias, entre as quais Ethernet, DSL, *optic fibre*, spectrum, etc.;

Domains - São utilizados para estabelecer pontes entre redes heterogéneas que possam existir no mesmo cenário de rede, como por exemplo a inclusão de um domínio *DiffServ* e um domínio *IntServ*, ligados ao domínio da *internet*.

Ainda na especificação do elemento *Node*, podem ser instanciados diversos elementos que irão afectar o seguimento da simulação do cenário criado. São sobretudo elementos que abordam a dinâmica do cenário entre os quais podemos assinalar os seguintes:

Application – Responsável pela geração de tráfego consoante um determinado protocolo e parâmetros específicos de configuração. Entre as diversas especificações de aplicações previstas temos o FTP, CBR, Telnet, Pareto que podem funcionar sob TCP ou UDP;

Protocol – Este elemento possui uma denotação contextual muito extensa pelo que as diversas especificações de protocolo podem ser de várias naturezas: protocolos de encaminhamento, protocolos de transporte, protocolos de endereçamento, etc. IPv4 para endereçamento e TCP/UDP para transporte são os mais utilizados, e;

Interface – Este elemento representa dispositivos que permitem a comunicação entre nós de rede. No contexto das redes com fios (*wired networks*) pode até representar um *link* de comunicação mas já nos ambientes sem fio (*wireless*) o conceito de *link* não seria o mais correcto, pois trata-se de um canal virtual e não físico, daí na realidade os fluxos de dados são transmitidos entre *interfaces*.

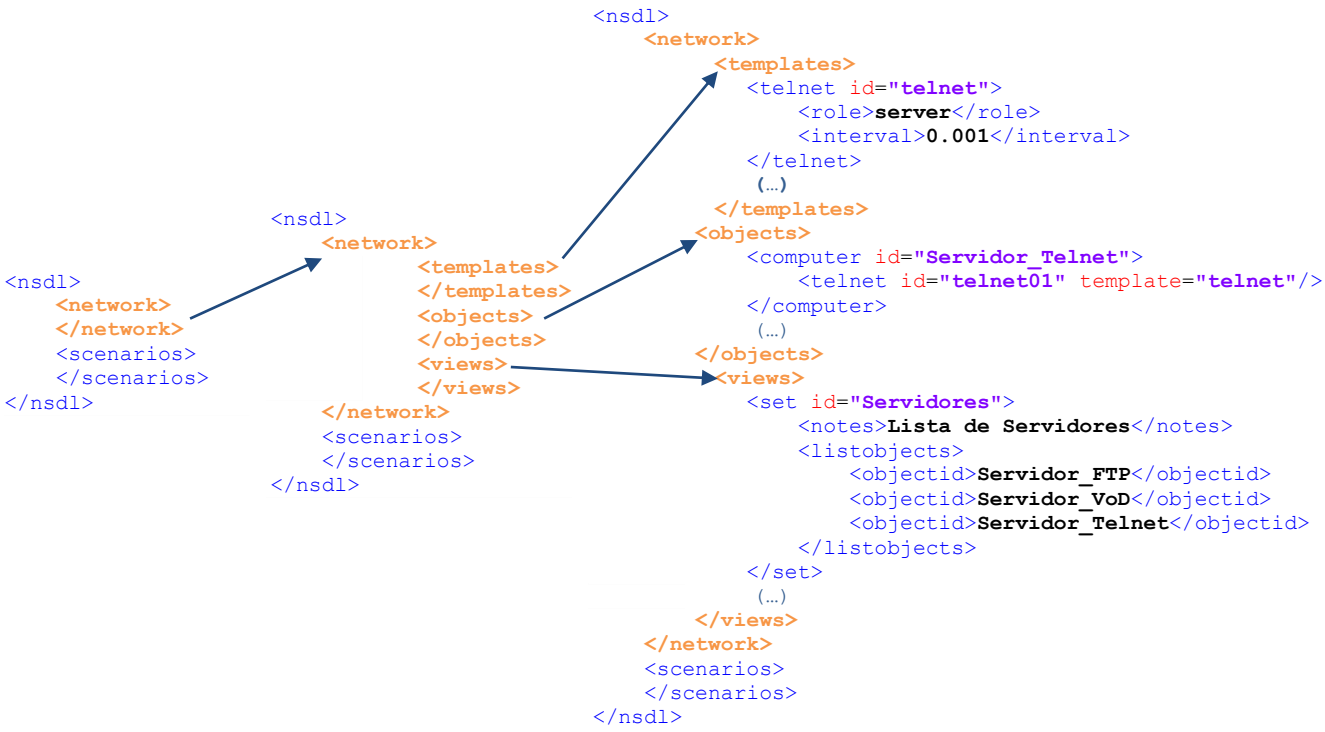

Figura 20 - Exemplo de codificação NSDL com realce ao elemento network

Através da Figura 20 podemos observar o *expandir* da codificação de um cenário NSDL representativo aos vários elementos contidos na *tag network*: *templates*, *objects* e *views* e um exemplo declarativo de como é programado na prática. Podemos observar que a instância de *telnet* incluída no objecto *computer* utiliza o *template* previamente definido. Com este exemplo fica clara a usabilidade dos *templates* que torna o código mais curto e organizado.

4.4.2. SCENARIOS

O elemento *Scenarios*, como já foi referido, inclui dois sub-elementos denominados por *Simulations* e *Visualizations*.

Simulations – Este elemento inclui a especificação de eventos que devem ocorrer em tempo de simulação sob um objecto referenciado, definidos através de tempos de início (*Start*) e o respectivo instante de paragem (*Stop*); Neste elemento são também especificadas algumas informações ao nível do simulador destino como por exemplo a versão, tempo de paragem da simulação global, versão do simulador, etc. Por fim, o elemento *Simulations* também inclui a definição opcional de variáveis específicas da plataforma destino contidas no elemento *extra* para o qual não existem quaisquer regras de especificação semântica e lexical, apenas precisam de estar previstas nos esquemas de tradução (XSLTs).

Visualizations – Neste elemento são definidas as coordenadas *X*, *Y* e *Z* de cada objecto existente na rede, bem como a definição de que tipo de mobilidade é atribuído a cada dispositivo. No caso concreto do mapeamento NSDL para NS-3, apenas posições fixas foram previstas (classe *ConstantVelocityMobilityModel*), sem suporte à mobilidade em tempo de execução.

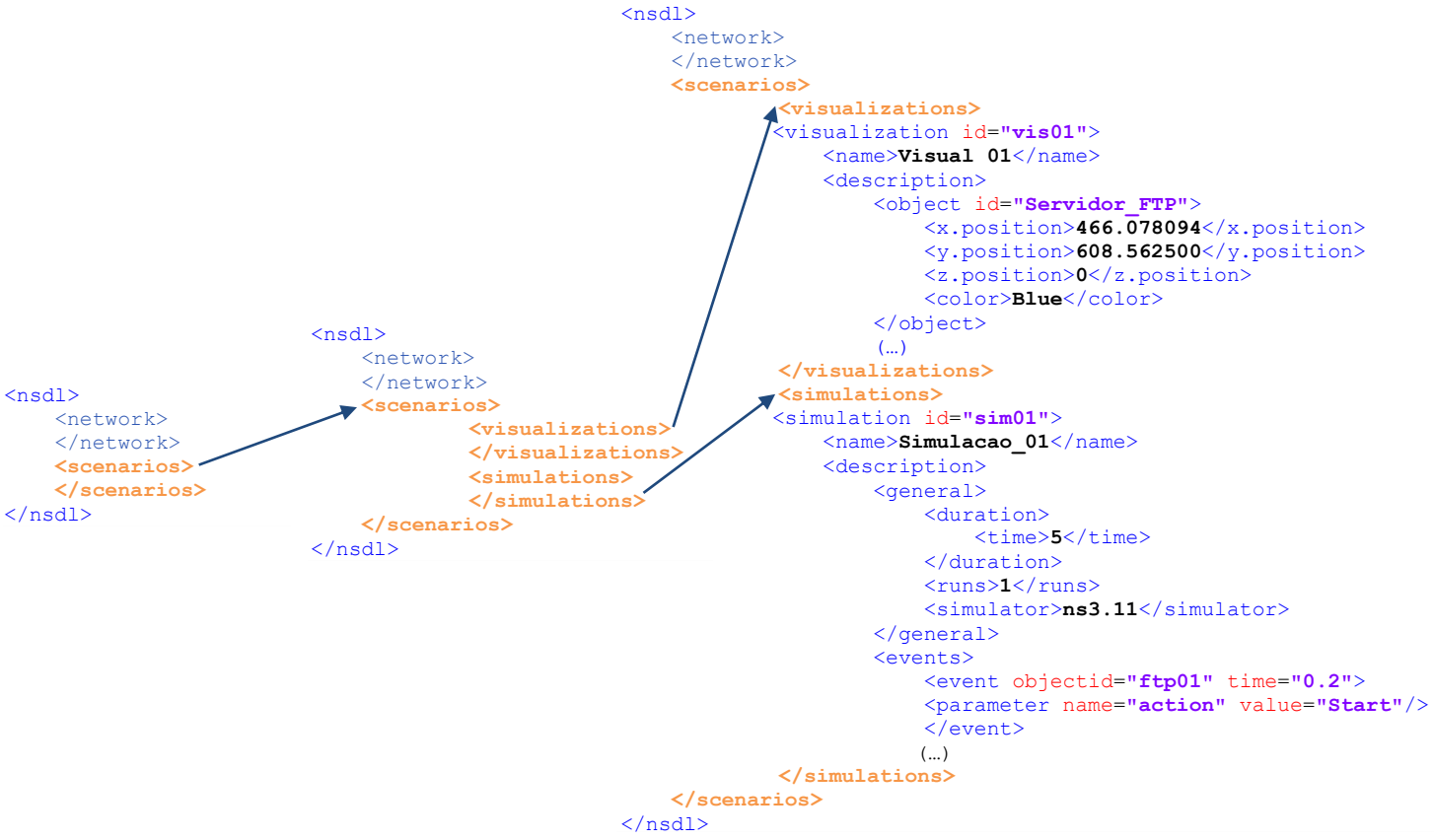

Figura 21 - Exemplo de codificação NSDL com realce ao elemento scenarios

Analogamente à Figura 20, a Figura 21 ilustra um excerto de codificação NSDL em que são expandidos no sentido da esquerda para a direita os elementos incluídos em *Scenarios* com um exemplo de especificação de objectos que são permitidos. Podemos verificar a especificação das coordenadas *X*, *Y* e *Z* de um objecto referenciado (`Servidor_FTP`) no elemento *visualization*.

Na tag *simulation* podemos verificar ainda a definição de tempos da simulação global (*tag* `<time>`), versão do simulador (*tag* `<simulator>`) bem como um exemplo de evento com a aplicação `"ftp01"` a possuir a definição de tempo de início aos `"0.2"` (segundos).

4.5. CONCLUSÃO

Neste capítulo foi apresentada a *Framework* NSDL que constitui uma solução para a integração de diversas aplicações visando a optimização da autoria e gestão de cenários de redes. Em especial, foi apresentada a linguagem NSDL que representa a camada intermédia da *Framework* NSDL e que permite estipular cenários de redes com uma estrutura própria de acordo com um conjunto de *tags*. Estas *tags* podem fazer parte do conjunto de objectos apresentados no *perfil* base como também podem pertencer a uma extensão de NSDL ou seja, um *perfil* específico para uma dada plataforma de simulação de redes.

No próximo capítulo será apresentada a proposta de extensão à linguagem NSDL, com a introdução de um novo *perfil* de forma a abordar a plataforma de simulação discutida no Capítulo 3, o NS-3.

5. EXTENSÃO À LINGUAGEM NSDL

No presente capítulo serão apresentados as tecnologias desenvolvidas no âmbito do simulador NS-3 que foram alvo de introdução na linguagem NSDL, acompanhadas de uma pequena descrição funcional.

5.1. INTRODUÇÃO

Como vimos no Capítulo 3, a linguagem NSDL possui um conjunto de elementos base que qualquer extensão desta linguagem deve conter. Numa fase inicial, este facto foi tido em conta para que o mapeamento para o novo *perfil* (i.e. *perfil* NS-3) contenha estes elementos e o respectivo suporte à sua tradução.

Posteriormente, outros elementos foram adicionados para que, de uma forma muito simples, representem uma das seguintes situações: Sejam variáveis globais específicas do simulador NS-3; Sejam tecnologias desenvolvidas no âmbito do NS-3 e por consequência, não existam no contexto do NS-2, ou então; Sejam apenas limitações/exigências da plataforma para que determinado elemento seja especificado correctamente (por exemplo o caso em que objectos previamente existentes necessitem de um atributo extra como é o caso dos ficheiros de *output* que necessitam de uma fonte de dados (`<source/>`).

É necessário ter em conta que nem todos os objectos existentes para o perfil NS-2 foram abordados para o perfil NS-3, naturalmente, devido ao facto de que muitas das tecnologias existentes no ambiente NS-2 ainda não terem sido desenvolvidas para NS-3 (como é o caso das arquitecturas do tipo ***I**ntegrated **S**ervices* (IntServ) [RFC 1633] e ***Diff**erentiated **Serv**ices* (DiffServ) [RFC 2475]) ou então encontram-se em desenvolvimento (como é o caso do ***M**ulti**P**rotocol **L**abel **S**witching* - MPLS [RFC 3031]).

Neste capítulo são mencionados que novos objectos foram desenvolvidos para o *perfil* NS-3 e qual a sua finalidade. Os atributos detalhados em termos de mapeamento NSDL para C++ encontram-se em anexo (Anexo II).

De acordo com a Figura 22, o perfil NS-3 é constituído essencialmente por 3 conjuntos de objectos de rede: *Generic Objects* (descrito no Capítulo 4) que contém os objectos com maior nível de abstracção e são por norma objectos coincidentes com qualquer perfil que seja eventualmente desenvolvido para NSDL; *Base TCP/IP Objects* que irá ser abordado de seguida na secção 5.2. e *Wireless Objects* que por sua vez se subdivide em outros 3 conjuntos de objectos, fazendo cada um alusão a um ambiente tecnológico distinto (Wi-Fi, WiMAX e LTE que irão ser abordados nas secções 5.3., 5.4., e 5.5. respectivamente).

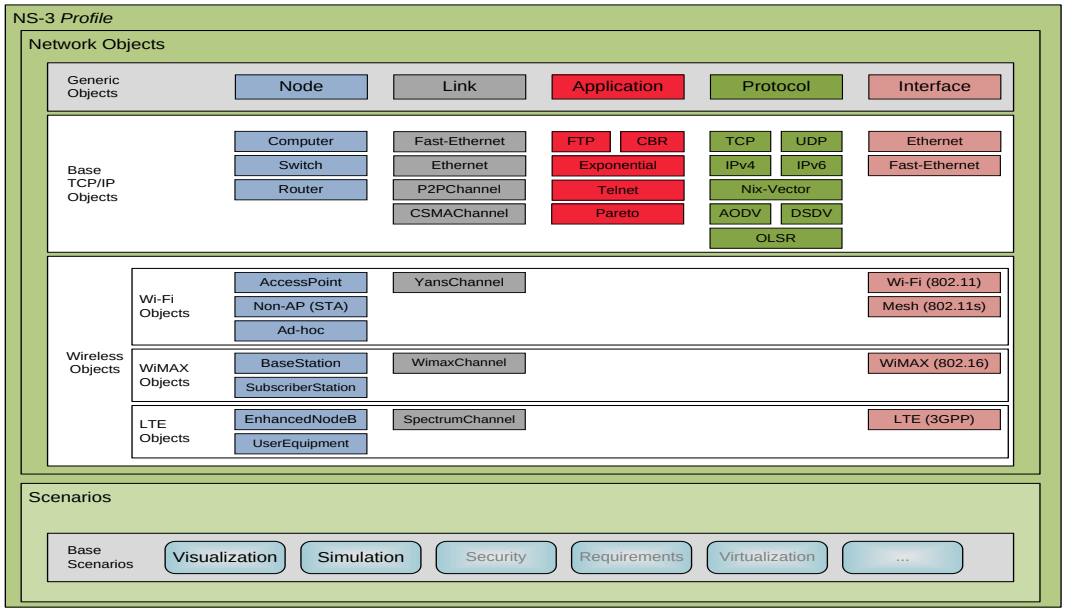

Figura 22 - Perfil NS-3 para extensão em NSDL

5.2. EXTENSÃO AO PERFIL BASE

A extensão ao *perfil* base (*Generic Objects*), apresentada nesta secção, é ilustrada na Figura 22 no conjunto denominado por *Base TCP/IP Objects*. A este nível de abstracção, apenas as colunas referentes às especificações de *Link* e *Protocol* sofreram extensão. Os restantes objectos foram previamente previstos para o *perfil* existente NS-2 e portanto, a sua inclusão no perfil NS-3 foi realizada sem grandes dificuldades, com algumas restrições a nível de atributos (alguns em falta, outros adicionados consoante o funcionamento do simulador NS-3).

Ao nível do *Link*, foi adicionado o elemento:

- ✓ **CSMAChannel** (`<link.csma/>`) – Embora não exista uma emulação real de um barramento do tipo **C**arrier **S**ense **M**ultiple **A**ccess (CSMA), o NS-3 permite ao utilizador a configuração de canais de transmissão com características similares ao de um CSMA real. É uma alternativa ao uso de canais do tipo P2PChannel (`<link/>`) (**P**oint-**to**-**P**oint) para cenários com cablagem física (*wired scenarios*).

Já ao nível dos protocolos (coluna *Protocol*), os elementos incluídos foram o protocolo de endereçamento IPv6 e outros protocolos de encaminhamento:

- ✓ **IPv6** (`<ipv6/>`) – Este protocolo, embora não explorado exaustivamente neste trabalho, foi abordado na tradução do endereçamento IPv6 para os vários nós de rede. IPV6 não foi explorado dada a sua incompatibilidade com algumas das tecnologias aplicadas para os cenários de redes (por exemplo, o protocolo de encaminhamento *Nix Vector*);

Qualquer um dos protocolos seguintes pode ser especificado ao nível global da simulação ou então somente ao nível de um nó em concreto (por exemplo um *router* que utiliza DSDV *routing* para a descoberta das rotas seguintes).

- ✓ **AODV** (`<aodv/>`) – O protocolo de encaminhamento **A**d hoc **O**n-Demand **D**istance **V**ector [RFC 3561][Klein-Berndt, 2011][Narra, 2011] funciona de acordo com a descoberta de rotas para o(s) "vizinho(s)" (*next hop(s)*) em que mensagens com a *string* "HELLO" são transmitidas entre um determinado nó e o(s) seu(s) respectivo(s) vizinho(s) em um intervalo de tempo (`<interval/>`). A ideia é encontrar o caminho mais curto para o caminho (*path*) a percorrer pelo pacote. As principais características deste tipo de protocolo são as seguintes: (i) Novas rotas serão encontradas somente quando necessário; (ii) Apenas conhece as rotas para o(s) nó(s) "vizinho(s)" ao invés de uma rota inteira (rota entre a fonte e o destinatário), e; (iii) Usa mensagens "HELLO" periodicamente para localizar eventuais "vizinhos";

- ✓ **DSDV** (`<dsdv/>`) – Já o **D**estination-**S**equenced **D**istance **V**ector [Narra, 2011][Perkins, 1994] trata-se de um protocolo de encaminhamento pró-activo que se baseia numa tabela de rotas em que o factor decisivo na escolha da rota a seguir é um vector de distância. Este vector é calculado com base na contagem de nós (*hop count*) que um certo pacote necessita de "percorrer" de forma a concretizar a rota fonte – destino. Naturalmente, a rota com uma contagem de nós menor será a escolhida para a transmissão. Este protocolo é considerado um precedente do protocolo AODV, tanto que em termos de desenvolvimento para NS-3, a codificação deste foi considerada como base para o desenvolvimento do DSDV;

- ✓ **Nix-Vector Routing** [Lee, 2005] (`<nix.vector/>`) – O Nix-Vector cujo nome provém da ideia utilizada no seu funcionamento (*N-index* Vector) utiliza um vector que inclui um conjunto de índices pertencentes aos nós existentes entre a fonte e o destino; Para alcançar este objectivo, é realizada uma procura eficiente entre os índices existentes em cada nó (*hop*) – Cada índice irá determinar que dispositivo de rede ou *Gateway* irá ser utilizado para a próxima transmissão. Este processo de transmissão e leitura de índice do próximo nó é realizado desde a fonte até ao destino.
  Este protocolo é ainda alvo de modificações relevantes ao seu funcionamento no que diz respeito a sua implementação NS-3. Por esta razão, ainda não são suportadas tecnologias como IPv6 nem qualquer abordagem a eventuais erros de transmissão no entanto suporta IPv4 e P2P/CSMA *links*, e;

- ✓ **OLSR** (`<olsr/>`) - **O**ptimized **L**ink **S**tate **R**outing Protocol) [RFC 3626] [Narra, 2011] constitui um protocolo frequentemente utilizado para autoria de cenários de redes e por consequência foi considerado na inclusão do *perfil* NS-3.
  É um protocolo pró-activo tal como o DSDV, em que múltiplos *broadcasts* são efectuados em toda a rede de forma a conhecer as rotas entre os eventuais nós. Devido ao facto de estar constantemente neste processo de *broadcast*, foi introduzido o conceito de **M**ulti**P**oint **R**elay (MPR) que estipula que só os nós

que receberem a notificação do nó *source*, irão efectuar um novo *broadcast*; Isto acontece de acordo com a disponibilidade anunciada previamente pelos nós "vizinhos" (<willingness/>) para retransmitir o sinal; O processo repete-se até atingir o destino. Tal como o protocolo AODV, transmite periodicamente mensagens com a *string* "HELLO" para descoberta e actualização de rotas para nós "vizinhos".

5.3. DESCRIÇÃO NSDL PARA WI-FI

Considerando o conjunto de objectos para wireless (*Wireless Objects* na Figura 22), inicialmente o domínio apresentado refere-se ao **Wi**reless **Fi**delity (**W**i-**F**i) com as suas respectivas variantes, em que se destaca a 802.11s (Mesh).

A extensão realizada para a descrição do domínio Wi-Fi contempla novos elementos para os objectos *Node*, *Link* and *Interface*.

Em relação aos objectos *Node*, os seguintes elementos foram propostos:

- ***A**ccess**P**oint* (***AP***) – Papel que caracteriza um ponto de difusão do sinal;
- *Non-AP* (*STA*) – Normalmente os receptores do sinal emitido pelos APs, e;
- ***Ad-hoc*** – Independente dos papéis anteriores, no entanto pode exercer funções de ambos.

Adicionalmente, estes papéis podem ser especificados no elemento de interface.

Os objectos *Link* constituem a camada física são estendidos para Wi-Fi pelo elemento:

✓ **YansChannel** (<link.yans/>) – ***Y**et **A**nother **N**etwork **S**imulator* foi desenvolvido com o objectivo de emular o modelo de propagação descrito em [Lacage, 2006]. Tal como descrito na sua classe NS-3, este tipo de link dá a possibilidade de emular um modelo de propagação em que a definição de um modelo de atraso (<delay.model/>) e/ou de um modelo de perdas (<loss.model/>) é opcional e não são instanciados quaisquer modelos por omissão aquando da instanciação da classe referente a este tipo de *link*. Este tipo de *link* é predominantemente utilizado nos ambientes *wireless*, mais concretamente em transmissão de dados entre *interfaces* do tipo Wi-Fi e Mesh.

A *Interface* é estendida para a descrição dos elementos:

✓ **Wi-Fi** (<wlan802.11/>) - O modelo Wi-Fi apresentado pela equipa de desenvolvimento NS-3 é constituído sobretudo por 2 componentes:

1. (non)QosWifiMac (*Upper level* - <nonqoswifi/>/<qoswifi>) – Instancia os principais atributos da camada de gestão dos fluxos de dados e atribui ao nó associado o seu tipo de responsabilidade; Estas

responsabilidades são denominadas por: ***A**ccess **P**oint* (***AP***), *Non Access Point* (*STA*) e ***Ad-hoc** Point* (***Adhoc***), mencionadas na extensão ao elemento *Node*;
Existe ainda a possibilidade de estabelecer um *standard* (`<standard/>`) em específico; As opções apresentadas pelo NS-3 são: *802.11a, 802.11b, 802.11g, 802.11_10MHz, 802.11_5MHz, holland, 802.11p_CCH e 802.11p_SCH* (estes dois últimos obsoletos);

2. YansWifiPhy *(Lower Level* - `<wifi.physical/>`) – Trata-se da componente física de transmissão de dados, incluindo o canal de comunicação Yans apresentado anteriormente.

Já a nível de gestão da *interface* do tipo Wi-Fi, é utilizado um gestor remoto (*Remote Manager* - `<remote.station.manager/>`) que é responsável pela atribuição de um estado aos pacotes existentes, utilizando para tal um dos vários algoritmos disponíveis:

- ***AARF** (Rate Adaption)* - (`<aarf/>`);
- ***AARFCD** (Efficient Collision Detection for Auto Rate Fallback Algorithm)* - (`<aarfcd/>`);
- ***AMRR*** - (`<amrr/>`);
- ***ARF*** - (`<arf/>`);
- ***CARA** (**C**ollision-**A**ware **R**ate **A**daptation)* - (`<cara/>`);
- ***ConstantRate*** - (`<constantrate/>`);
- ***Ideal*** - (`<ideal/>`);
- ***ONOE** (Rate control algorithm developed by Atsushi Onoe)* - (`<onoe/>`);
- ***RRAA** (**R**obust **R**ate **A**daptation **A**lgorithm)* - (`<rraa/>`), e;

✓ **Mesh** (`<wlan802.11s/>`)[Andreev, 2010] - As redes de natureza Mesh (Standard IEEE 802.11s) abordadas na óptica do NS-3 são constituídas essencialmente por duas camadas:

1. MAC (L2 – Layer 2)
2. **PHY** (**Phy**sical – Transport Layer)

A definição de uma interface do tipo Mesh (*MeshPointDevice*) pode ser tipicamente de duas variantes (`<stack.type/>`):

1. **Draft s** (IEEE 802.11s (mesh) draft standard implementation)
2. **FLAME** (**F**orwarding **La**yer for **Me**shing Protocol)

Em termos de definição MAC, é análogo ao do *standard* 802.11 (Wi-Fi) pois o standard 802.11s trata-se de uma extensão do primeiro (inclui definição de *standard* e *remote manager* previamente descritos). Em específico para o *standard* 802.11s, são necessários definir 2 componentes:

1. PMP (**P**eer **M**anagement) - Usado para abrir, manter e fechar *links* de comunicação entre estações mesh "vizinhas";
2. HWMP (**H**ybrid **W**ireless **M**esh **P**rotocol) – Apresenta dois modos de funcionamento: (i) *On-demand route discovery, e; (ii) Proactive route*

discovery. Ambos obrigatórios e por consequência criados na instanciação de *MeshPointDevice*.

Para a camada física (PHY - `<mesh.physical/>`) é utilizado um canal de transmissão de dados do tipo Yans (tal como para Wi-Fi).

5.4. DESCRIÇÃO NSDL PARA WiMAX

A extensão realizada para a descrição do domínio WiMAX também contempla novos elementos para os objectos *Node*, *Link* e *Interface*.

Em relação aos objectos *Node*, os seguintes elementos foram propostos:

- ***BaseStation (BS)*** – Estação emissora do sinal recepcionado, enviado pelos SSs emissores e retransmitido para os respectivos SSs receptores; Funciona analogamente a um *router* com tratamento "*last-mile*", e;
- ***SubscriberStation (SS)*** – Estação que subscreve à BaseStation para envio ou recepção de fluxo(s) de dados; Cada SS está associado a um único BS em que este irá realizar a gestão dos respectivos SSs associados.

No âmbito dos objectos *Link*, o seguinte elemento foi proposto:

✓ **WimaxChannel** (`<link.wimax/>`) – Este elemento foi desenvolvimento com o principal objectivo de dar suporte aos ambientes WiMAX, emulando para tal um canal de transmissão virtual que envia blocos FEC (**F**orward **E**rror **C**orrection) em modo *burst* [Farooq, 2009]. À semelhança do tipo de *link* Yans, o *link* Wimax permite a emulação de um modelo de propagação em que podemos especificar um modelo de perdas (`<loss.model/>`).

Em relação aos objectos *Interface*, WiMAX foi definida com o elemento:

✓ **WiMAX** (`<wlan802.16/>`) - Já o modelo para **W**orldwide **I**nteroperability for **M**icrowave **Acces**s apresentado pela equipa do NS-3 é composto essencialmente por 3 camadas [Farooq, 2009][Sievanen, 2003]:

1. **Covergence Sublayer (CS)** – Subdividido ainda por outras 2 camadas:
   - **Packet CS** – Responsável pela classificação dos pacotes e reencaminhamento para as devidas conexões (ou ***serviceflows***). Opera sobre o Protocolo de Internet IP, IPv4, IPv6, **P**oint-to-**P**oint **P**rotocol (PPP) e também sobre links *Ethernet* (IEEE standard 802.3), e; Permite a especificação de requisitos QoS (`<service.qos/>`);
   - **ATM CS** [eTutorials, 2011] – O ***A**synchronous **T**ransfer **M**ode* tem como principais funções aceitar a conexão a eventuais nós ATM existentes na rede, classificá-los e o reencaminhamento de PDUs (**P**rotocol **D**ata **U**nit) [Sievanen, 2003] para os respectivos

MAC-SAP (***M**edium **A**ccess **C**ontrol Layer – **S**ervice **A**ccess **P**oint*).

2. **MAC Sublayer** – Principal camada do *standard* 802.16 Suporta PMP (**P**oint-to-**M**ulti**p**oint) e o registo e identificação de todas as estações: A ***Base Station*** (BS) que controla e reencaminha os fluxos para os ***Subscriber Stations*** (SSs) existentes, e; as SSs que são responsáveis pelo "*framing*" dos pacotes. Além da gestão efectuada sobre os SSs, a BS deve manter um controlo sobre os fluxos de dados que são transmitidos (***ServiceFlows***) – Esta tarefa é realizada com auxílio de algoritmos próprios de gestão:
    - **UGS – Unsolicited Grant Service** – Aplicações com tamanho de pacote fixo;
    - **rtPS – Real-Time Polling Service -** Aplicações com tamanho de pacote fixo em tempo real;
    - **nrtPS – non (rtPS) -** Aplicações com tamanho de pacote variável, e;
    - **BE – Best Effort –** Alocação de banda larga (*bandwidth*) apenas quando disponível.

Nota: A maioria destas funcionalidades encontra-se ainda muito limitadas (por exemplo, o modelo actual só permite 1 ***ServiceFlow*** por SS).

3. **PHY Layer (Physical Layer)** – 2 tipos de camadas físicas:

A primeira camada simples reencaminha em modo "*burst*" sem considerar variáveis de controlo de tráfego (possibilitado pela classe *SimpleWimaxPhy*). Por outro lado a segunda camada converte os pacotes em blocos FEC utilizando a divisão do tipo OFDM (**O**rthogonal **F**requency-**D**ivision **M**ultiplexing) – Utilização da classe *OfdmWimaxPhy* - possibilitando o uso de canais de comunicação virtuais por cada uma das camadas para transmissão de dados: *SimpleWimaxChannel* e *OfdmWimaxChannel* para as camadas 1 e 2, respectivamente.

5.5. DESCRIÇÃO NSDL PARA LTE

A extensão realizada para a descrição do domínio LTE contempla novos elementos para os objectos *Node*, *Link* e *Interface*.

Em relação aos objectos *Node*, os seguintes elementos foram propostos:

- ***E**nhanced**N**ode**B*** (***eNB***) – Funcionamento análogo ao do *Base Station* descrito para a tecnologia WiMAX; Sendo assim, a sua principal função é a de reencaminhar tráfego para os respectivos dispositivos (UEs), e;
- ***U**ser**E**quipmen*t (***UE***) – Dispositivos móveis conectados ao eNB para transmissão de dados (*downlink* e *uplink*).

No âmbito dos objectos *Link*, o seguinte elemento foi proposto:

- ✓ **SpectrumChannel** (`<link.spectrum/>`) – Este último tipo de link desenvolvido, trata-se de uma especificação de *link* que visa a emulação de propagação de espectros, tal como acontece nos ambientes **UTRAN** (**U**MTS **T**errestrial **R**adio **A**ccess **N**etwork). Este *link* é utilizado para transmissões de dados entre *interfaces* do tipo LTE (**L**ong **T**erm **E**volution), que irá ser apresentado na secção de "*Interfaces*" que se segue. Tal como o *link* Yans, este permite especificar um modelo de propagação (modelos de perdas – (`<loss.model/>`) e de atraso (`<delay.model/>`)).

Como Interface o elemento proposto é descrito:

- ✓ **LTE** (`<lte/>`) - Como última tecnologia abordada (ambos em aspectos de tecnologias com e sem fios) pelo mapeamento realizado em termos de NSDL para NS-3, o módulo LTE (**L**ong **T**erm **E**volution) desenvolvido para o NS-3 detém a seguinte especificação [NS3-LTE, 2011]:

O módulo NS-3 tem como finalidade emular uma rede **E-UTRAN** (**E**volved-**U**niversal **T**errestrial **R**adio **A**ccess **N**etwork) da família de infra-estruturas **3GPP UMTS** (**3**rd **G**eneration **P**artnership **P**roject **U**niversal **M**obile **T**elecommunications **S**ystem) em que se destacam tipicamente dois tipos de nós, os eNBs e os UEs, que podem ser visualizados na Figura seguinte:

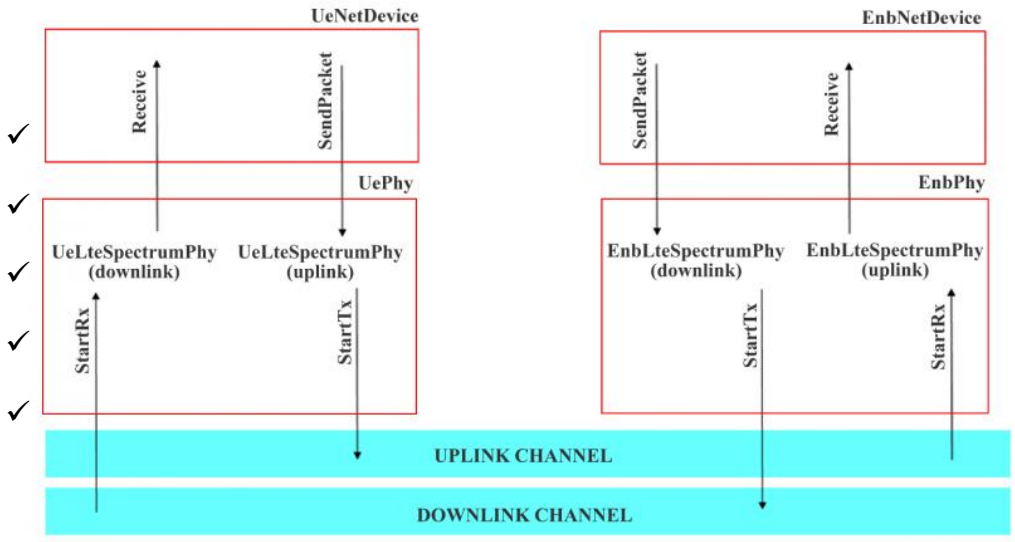

Figura 23 - Transmissão DL e UL no módulo LTE [NS3-LTE, 2011]

Em que as classes *UeNetDevice* e *EnbNetDevice* representam os dispositivos associados aos nós EU e eNB respectivamente. As transmissões realizadas entre ambos os dispositivos são definidas ao nível da camada física (PHY Layer) com a instanciação da classe *SpectrumChannel* que representa a criação virtual de um espectro de comunicação entre os vários dispositivos interconectados.

5.6. VARIÁVEIS GLOBAIS

As variáveis apresentadas de seguida, embora possam ser características comuns de outros simuladores que não o NS-3, no âmbito do NSDL foram introduzidas para suporte exclusivo ao NS-3:

- ✓ **Real Time** (`<realtime/>`) – A definição desta variável booleana, determina se a simulação irá ser executada em tempo real ou não, e;

- ✓ **CheckSum** (`<enable.checksum/>`) – Por questões de performance, o cálculo dos *checksums* está desactivado por omissão no NS-3. Caso o utilizador queira activá-lo, poderá definir esta variável booleana como *true* para posterior constatação em ficheiros do tipo *.pcap*.

De seguida é apresentada uma possível representação do *perfil* NS-3 adoptado para NSDL em que se encontram, muito resumidamente, os principais objectos desenvolvidos, muitos dos quais foram previamente apresentados, que estipulam a distinta identidade deste *perfil*. Como podemos verificar, não inclui especificações de *node* (ao contrário da Figura 22) pois actualmente o papel específico de cada nó (seja este uma *Base Station*, um *Access Point*, etc.) é estipulado na respectiva especificação de *interface*. Resumidamente, o perfil NS-3 para NSDL sofre alterações nomeadamente a nível de *links* existentes e tipos de *interface*.

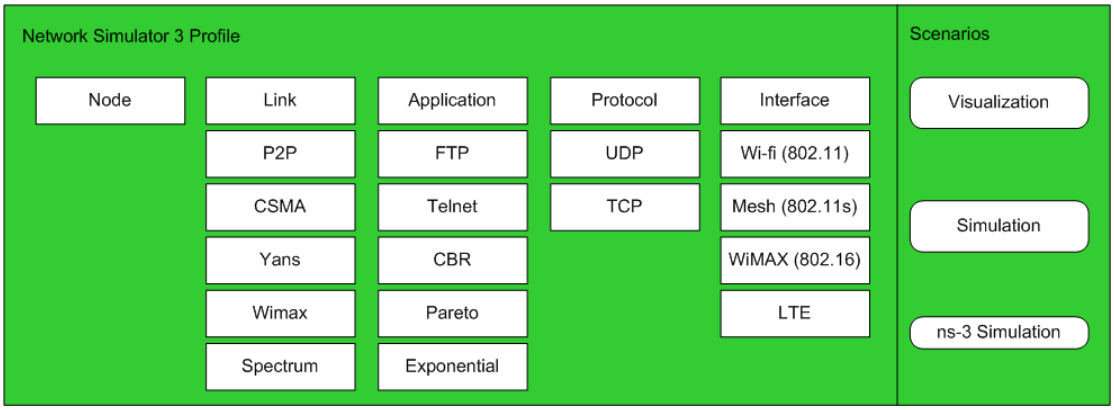

Figura 24 - Perfil NS-3 com alusão aos principais objectos abordados

5.7. CONCLUSÃO

Neste capítulo foram apresentados os principais elementos adicionados ao domínio lexical do NSDL, acompanhados de uma descrição informal sobre o funcionamento de cada elemento no contexto do NS-3, que irão permitir aos utilizadores

da *Framework* NSDL criar cenários de redes mais ricos a nível de tecnologias utilizadas e respectivas especificações.

No seguinte capítulo irá ser descrita que abordagem ao mapeamento, destes e outros elementos previamente existentes relativamente ao *perfil* NS-3, foi adoptada para a tradução de cenários NSDL (XML) para cenários NS-3 (C++).

6. ABORDAGEM AO MAPEAMENTO NSDL -> NS-3

Neste capítulo é apresentada a abordagem para o mapeamento de cenários de Rede descritos em NSDL para a sua representação em C++, a ser utilizada em NS-3.

6.1. INTRODUÇÃO

A abordagem de mapeamento NSDL para NS-3 surge a partir da experiência prévia adquirida no âmbito do Projecto NSDL no mapeamento NSDL para NS-2 [Marques, 2010]. O mapeamento NSDL para NS-2 proporcionou a base para o tratamento de Ficheiros NSDL, permitindo futuros mapeamentos baseados em NSDL serem optimizados.

Além duma visão geral da abordagem de mapeamento proposta, neste capítulo serão também apresentamos os principais requisitos funcionais e não funcionais identificados, casos de utilização, diagramas de actividades e a arquitectura da solução implementada.

6.2. REQUISITOS

De seguida são apresentados os principais requisitos, funcionais e não-funcionais relativos ao mapeamento NSDL -> NS-3. Os requisitos relacionados a um sistema representam um conjunto de premissas que o sistema, uma vez operacional, deve respeitar. Existem diversas formas de os categorizar sendo a mais frequente a de classificar os requisitos por funcionais e por não-funcionais como irá ser apresentado nas subsecções seguintes.

6.2.1. REQUISITOS FUNCIONAIS

Os requisitos funcionais apresentados de seguida são relativos à plataforma de mapeamento NSDL para NS-3 e englobam as principais funcionalidades que a aplicação deve fornecer ao utilizador (normalmente autores de cenários de redes) como também a eventuais colaboradores para manutenção/desenvolvimento da *Framework* NSDL. Os requisitos funcionais descrevem que conjunto de acções que um sistema deve ser capaz de realizar, abstraindo-se de aspectos relacionados com recursos físicos; Tratam essencialmente fluxos de dados de entrada ou saída na interacção do utilizador com o sistema, que tipicamente são representados com auxílio aos casos de utilização que irão ser apresentados na secção 6.3. [RSC, 2001][Sommerville, 2004].

Sendo assim, os principais requisitos funcionais assinalados foram os seguintes:

- O sistema deve ser capaz de traduzir eficiente e correctamente qualquer cenário descrito em NSDL para um cenário descrito em C++ desde que este possua na sua especificação elementos (*tags*) do domínio lexical do NSDL (*Eficiência e correcção*);

- O sistema deve fornecer suporte à especificação de cenários NSDL visando a obtenção de um cenário NS-2 e/ou NS-3, disponibilizando para tal documentação alusiva aos diversos elementos incluídos nos perfis NS-2 e NS-3 respectivamente;

- Em termos de formulários de submissão de ficheiros para eventuais validações/traduções, o sistema deve ser capaz de identificar campos inválidos e/ou em falta e devolver o respectivo *feedback* para o utilizador de forma a regularizar o problema detectado (*Tratamento de erros*);

- Ainda na óptica de *feedback* por parte do sistema, este deve fornecê-lo aquando de uma validação de um ficheiro que resultou em 1 ou mais erros detectados para posterior depuração por parte do utilizador, e;

- Em caso de sucesso do processo de mapeamento NSDL para NS-2 / NS-3, o sistema deve disponibilizar um *link* para que o utilizador possa efectuar o download do ficheiro de output resultante.

6.2.2. REQUISITOS NÃO-FUNCIONAIS

Os seguintes requisitos foram denominados por não-funcionais por não afectarem directamente as funcionalidades oferecidas pela aplicação, mas sim que recursos físicos são utilizados pelo uso da aplicação por parte dos utilizadores. São sobretudo requisitos que descrevem atributos do sistema e do ambiente.

De entre os requisitos não-funcionais, podemos identificar como os mais importantes os seguintes [Sommerville, 2004]:

- *Portabilidade*

  A solução a desenvolver deve suportar a sua apresentação nos diversos navegadores existentes, nomeadamente aqueles são considerados como os mais populares: Mozilla Firefox, Internet Explorer (v.6 ou superior), Chrome, Opera, etc.;

- *Consistência*

Toda a metodologia usada para o desenvolvimento da solução deve ser feita de forma a evitar eventuais conflitos entre os vários módulos e funcionalidades existentes (sobretudo a nível de processamento, partilha de recursos, etc.);

- *Redundância*

De forma a efectuar uma verificação exaustiva sobre o documento NSDL eventualmente submetido pelo utilizador, uma dupla verificação deve ser operada, caso isto não implique a sobrecarga dos recursos disponibilizados;

- *Disponibilidade*

A aplicação deve estar disponível para consulta bem como apresentar suporte em caso de falha do sistema, com um intervalo de reactivação da aplicação num intervalo máximo de 1 dia útil (*Recuperação*).

- *Tratamento de requisições*

Embora não muito relevante a nível académico, o servidor detentor da aplicação desenvolvida deve ser capaz de lidar com o mínimo de 15 requisições simultâneas.

6.3. CASOS DE UTILIZAÇÃO

Para que seja mais perceptível para os utilizadores da aplicação (na óptica do que pode ser realizado com a ajuda da ferramenta) bem como para eventuais desenvolvedores/gestores (manutenção/modificação da aplicação), foi desenvolvida uma pequena representação alusiva aos casos de utilização possíveis no uso da aplicação desenvolvida (Figura 25).

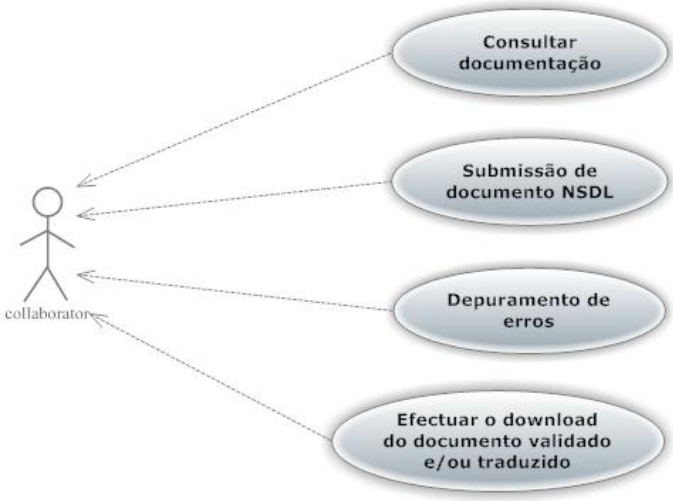

Figura 25 - Casos de uso possíveis através da interacção com a aplicação de mapeamento

Na solução proposta é prevista a existência de apenas um utilizador comum que tem por objectivo criar os seus cenários de redes com o auxílio da aplicação desenvolvida, constituindo um colaborador.

De entre os possíveis casos de utilização identificados para o colaborador, estão:

- *Consultar documentação* - A consulta de documentação relativa ao perfil NS-3 pode ser realizada visitando o *link* seguinte: http://apus.uma.pt/nsdl/xml/docs/nsdl_main.html. Após consulta e estudo da documentação, o colaborador poderá desenvolver o seu documento NSDL através de ferramentas externas, em que se destacam o VND descrito no Capítulo 2 para autoria gráfica de cenários de rede e as ferramentas oXygen [Oxygen, 2011] e Notepad++ [Notepad++, 2011] para desenvolvimento textual;
- *Submissão de documento NSDL* - Através da interface gráfica, o utilizador terá disponíveis as opções de mapeamento NSDL para NS-2 e para NS-3 em que a escolha de cada perfil será determinante na selecção dos esquemas de validação (XSDs) e de tradução (XSLTs) a utilizar;
- *Tratamento de erros* - Quando for submetido o ficheiro, se houver erros em termos de sequência/sintaxe por parte do documento NSDL, a aplicação irá devolver os eventuais erros de forma a facilitar a tarefa de depuração de erros por parte do utilizador. É de salientar no entanto que a tradução do ficheiro é realizada mesmo que a validação não seja totalmente conseguida isto é, mesmo na eventual existência de erros, a aplicação traduz o ficheiro com base na correspondência entre as *tags* existentes no documento NSDL e os *templates* existentes nos esquemas XSLT;
- *Efectuar o download do documento validado e/ou traduzido* - Por fim é fornecido ao utilizador um *link* que permite efectuar o download do documento resultante após aos processos de validação e de tradução. Este documento poderá apresentar, em complemento com os erros XML, ou seja, comentários alusivos a erros que possam existir na sintaxe NSDL ou até mesmo inconsistências nos valores dos atributos.

6.4. DIAGRAMAS DE ACTIVIDADE

O diagrama de actividade ilustrado na Figura 26 proporciona uma visão geral dos 4 casos de utilização apresentados no próximo capítulo sobretudo devido à natureza sequencial em que os mesmos ocorrem na autoria de qualquer cenário de rede. É certo que com a experiência acumulada pelo utilizador, passos como a verificação da documentação e depuração de erros poderão tornar-se obsoletos, mas numa fase inicial de interacção são fundamentais.

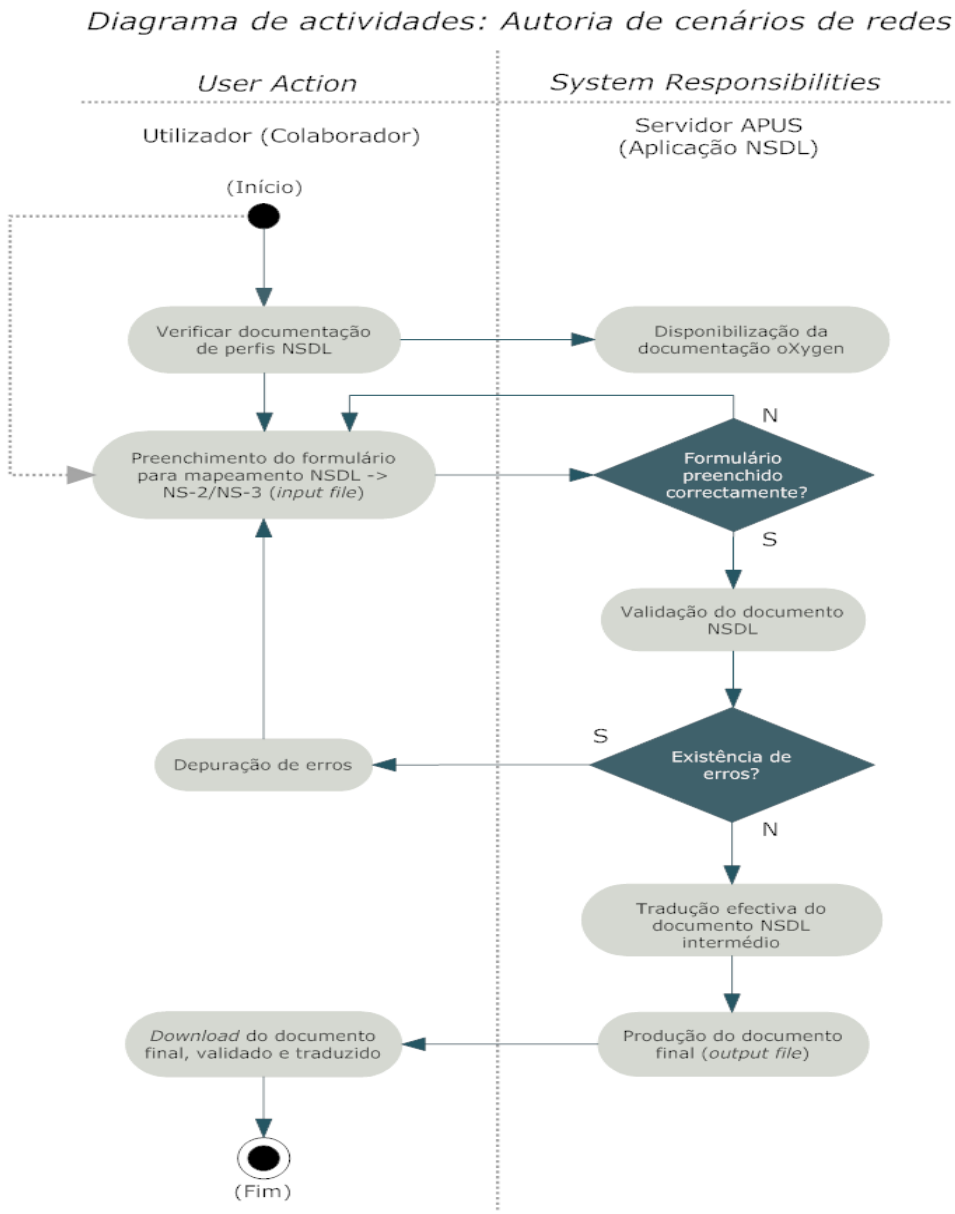

Figura 26 - Diagrama de actividades complementar ao conjunto de casos de uso

Na Figura 26 é possível observar quais as acções que o utilizador pode realizar, permitidas pelo sistema e que responsabilidades são atribuídas a este. De salientar que não é decisiva a resposta derivada da questão "*Existem erros?*" para que seja iniciado o processo de tradução da estrutura NSDL fornecida, ou seja, sendo a resposta "*Sim*" ou "*Não*", a tradução irá ser realizada. A representação ilustrada define o caso que deve ser idealmente seguido, pois caso existam erros, é recomendável que estes sejam corrigidos e que uma nova submissão do ficheiro NSDL seja realizada.

6.5. DIAGRAMA DE CLASSES

O mapeamento realizado entre a estrutura de dados NSDL e um *script* final C++ implica a utilização de um conjunto de classes de módulos NS-3 e por vezes, surgem classes que instanciam outras classes do mesmo módulo em que se encontram ou até mesmo de outros módulos. Esta última situação não será perceptível visto que os módulos serão apresentados individualmente de forma a optimizar a compreensão do respectivo funcionamento.

A apresentação dos vários módulos segue o conjunto de módulos NS-3 apresentados no Capítulo 3, com a particularidade de apenas os que foram abordados na totalidade ou parcialmente, possuem uma representação UML.

Um aspecto em ter em conta relativamente à representação UML dos seguintes módulos é o de que nem todos os atributos/métodos são explicitamente especificados/utilizados: Isto significa que aqueles atributos que não são considerados fundamentais ou que a sua configuração por omissão seja a ideal para o funcionamento do objecto em questão, não são modificados no processo de mapeamento. Por outro lado, existem diversos métodos que embora não sejam utilizados explicitamente no *script* C++, são invocados aquando da sua execução no ambiente NS-3 (por exemplo, o método *DoSend()* apresentado em quase todos os *NetDevices*, que é utilizado para enviar fluxos de dados).

Devido ao facto de alguns dos diagramas UML não serem suficientemente visíveis em formato A4, estes foram colocados na pasta “Diagramas UML” presente no CD da tese.

6.6. ARQUITECTURA DO SISTEMA

Para a descrição da arquitectura de implementação do sistema foi adoptada a arquitectura **MVC** (**M**odel-**V**iew-**C**ontroller) [Hyfinity, 2004][Gilzar, 2002], tipicamente utilizada no desenvolvimento de aplicações em Web. Esta arquitectura divide as responsabilidades do sistema tipicamente em 3 contentores (*containers*) distintos, cada um fazendo referência aos módulos do MVC: Um *container* (***Controller***) irá representar a lógica de negócio (os principais modelos de operação no sistema), um segundo *container* (***View***) irá englobar os ficheiros responsáveis pela disponibilização de informação (a interface gráfica entre o sistema e o utilizador final) e um terceiro e último *container* (***Model***) que detém um conjunto de ficheiros de controlo que são responsáveis pela definição do comportamento da aplicação e onde são codificadas as principais decisões do sistema (no caso concreto da abordagem ao mapeamento NSDL, decisões de validação de formulários, validação de documentos XML (Esquemas XML – XSDs), tradução (XSLTs), etc.). A arquitectura MVC foi adoptada no âmbito deste trabalho devido ao facto de na sua definição abordar uma organização eficiente dos vários ficheiros que fazem

parte da aplicação em categorias distintas (ficheiros modulares, ficheiros de apresentação e ficheiros de controlo) bem como fornece suporte à manutenção/modificação do sistema por eventuais colaboradores, aspecto que foi assinalado na secção dos requisitos funcionais (5.2.1.) como condição a ser respeitada.

A Figura 27 tem por finalidade a organização dos ficheiros utilizando a arquitectura MVC.

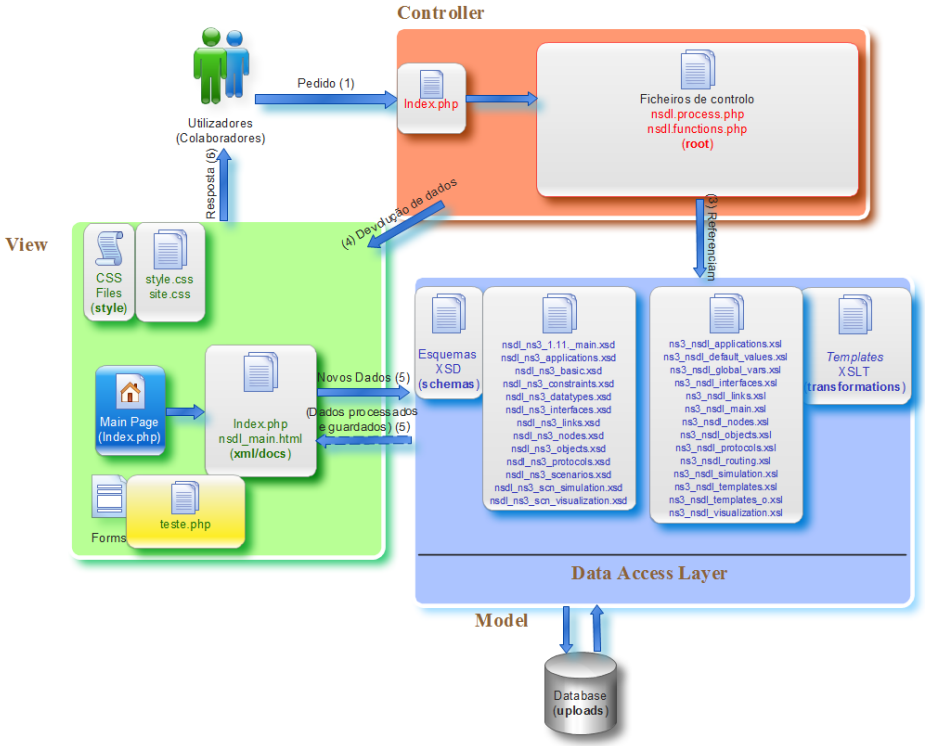

Figura 27 - Arquitectura MVC da aplicação de mapeamento NSDL

Na Figura 27 é apresentada uma divisão ternária que separa os ficheiros segundo as suas responsabilidades perante o sistema. Na divisão à esquerda (***View***) foram incluídos os ficheiros responsáveis pela definição da interface gráfica que irá ser apresentada ao utilizador: Inclui a página inicial (*Index.php*), formulários e respectivas chamadas aos ficheiros com a definição de estilos e formatação (ficheiros CSS). Já na divisão apresentada à direita superior (***Controller***) foram incluídos os ficheiros que afectam o comportamento do sistema perante as acções realizadas pelo utilizador – ficheiros PHP detentores das funções presentes no Anexo I que por sua vez referenciam os ficheiros de módulo incluídos na 3ª e última divisão (inferior à direita - ***Model***) que são os ficheiros responsáveis pela validação (XSDs) dos ficheiros submetidos para tradução por parte dos utilizadores para posterior tradução consoante as definições presentes nos ficheiros de transformação (XSLs). Já a base de dados guarda os ficheiros submetidos bem como as versões intermédias dos mesmos (resultantes da aplicação da função **function** nsdl_uniform), bem como os ficheiros traduzidos (ficheiros de *output*) para disponibilizá-los ao utilizador para *download*.

Os ficheiros apresentados pela arquitectura MVC encontram-se no CD na pasta denominada por "*Server*" e pode ser simulada com uma ferramenta própria (por exemplo XAMPP) ou alojada num servidor na internet.

6.7. REPRESENTAÇÃO DA ABORDAGEM DE MAPEAMENTO NSDL/NS-3

De forma a realizar uma tradução atendendo às várias validações adjacentes a qualquer tradução entre diferentes linguagens (como validações lexicais, semânticas, etc.), a Figura 28 ilustra os principais passos da abordagem de tradução NSDL/NS-3.

Os principais passos da abordagem proposta são discutidos a seguir:

- ***Input de ficheiro***

Inicialmente é necessário que um ficheiro NSDL (XML) seja fornecido no formulário presente no ficheiro PHP (*teste.php*);

- ***Validação lexical e semântica***

Para estes dois passos são necessários ter em conta dois ficheiros: o ficheiro que contém um conjunto de regras referentes à estrutura a que o documento carregado deve obedecer ou seja, a *Definição de Esquema XML* (XSD) – no caso específico do mapeamento NSDL -> NS-3 (nsdl_ns3_1.11._main.xsd) e o ficheiro responsável por efectuar as transformações (um ficheiro XSLT) consoante os *templates* invocados – para NSDL -> NS-3 (ns3_nsdl_main.xsl);

- ***Criação de um ficheiro output***

Finalmente, após os processos de validação e tradução, é então criado um ficheiro.tcl/.cc para então ser executado em NS-2/NS-3.

É necessário ter em conta que este ficheiro final poderá eventualmente conter comentários que são gerados quando é verificada alguma inconsistência no ficheiro NSDL, como por exemplo atributos inexistentes/valores atribuídos fora do intervalo permitido, etc.

Após executados estes passos de tradução de ficheiros NSDL -> NS-3 foram elaborados alguns cenários que utilizam objectos já previstos para o NS-2 bem como novos objectos introduzidos no âmbito do NS-3, de forma a validar a abordagem realizada ao mapeamento descrito neste capítulo. Os cenários mencionados serão alvo de estudo no capítulo seguinte.

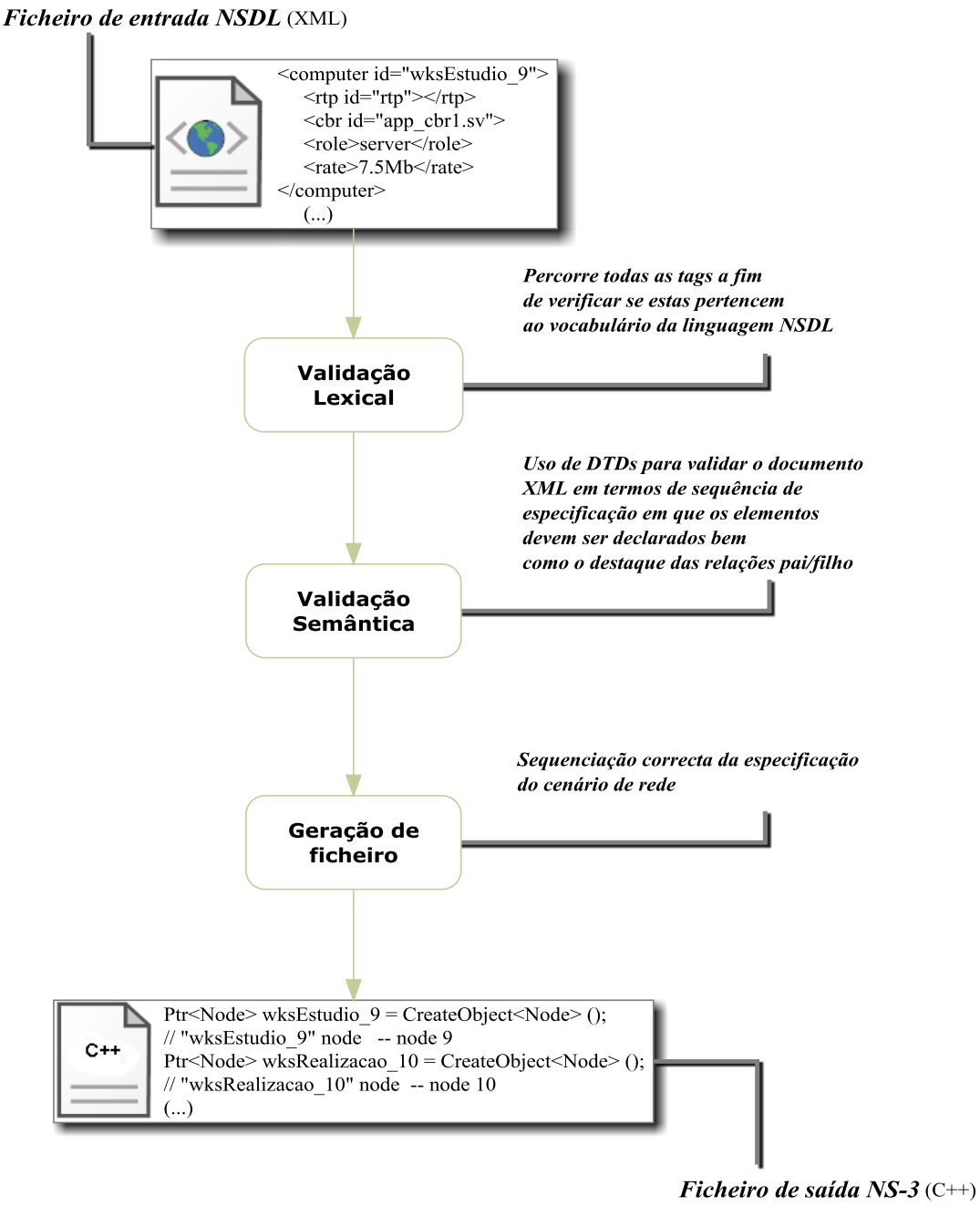

Figura 28 - Metodologia de mapeamento NSDL -> NS3

6.8. OPÇÕES TECNOLÓGICAS

No decorrer do desenvolvimento da solução disponibilizada, as tecnologias utilizadas foram essencialmente:

- ✓ XML para especificação de cenários NSDL;
- ✓ XSD para definição de que objectos NSDL são permitidos num documento XML e em que ordem (sequência de especificação) estes devem ser apresentados;

- ✓ XSLT para inclusão dos diversos *templates* de codificação a aplicar quando existir correspondência entre as *tags* existentes no documento XML e as *tags* identificadoras dos *templates* em XSL, e;
- ✓ PHP para fornecer suporte ao processo de uniformização de documentos XML submetidos pelos colaboradores bem como gerir todo o processo de mapeamento NSDL -> NS-2/NS-3 (i.e. o processo descrito na secção anterior – 6.6).

No que diz respeito às ferramentas de desenvolvimento utilizadas, para edição de ficheiros de linguagens como o XML, XSLT e PHP foi utilizado o Notepad++ [Notepad++, 2011] em que pode ser seleccionada que linguagem de programação está a ser editada bem como possui suporte a funcionalidades populares como é o caso do "*auto-complete*" embora saiba que um editor próprio para cada uma das linguagens fosse o ideal, a escolha do Notepad++ deve-se sobretudo à simplicidade de interacção com a plataforma relativamente, por exemplo, ao Dreamweaver [Dreamweaver, 2011].

Para edição dos esquemas de validação (XSD) foi utilizada a ferramenta oXygen [Oxygen, 2011] pois possui grande suporte à especificação da árvore de validações para documentos XML (tanto gráfica como textual) e permite, já numa fase final de edição, elaborar a documentação necessária ao utilizador para suporte à autoria de cenários de redes NSDL.

6.9. CONCLUSÃO

Neste capítulo foram apresentados os principais requisitos da aplicação de mapeamento desenvolvida bem como os objectos que foram alvo de desenvolvimento no âmbito do *perfil* NS-3 para autoria de cenários de redes NSDL, bem como os processos adjacentes como os de validação e tradução de ficheiros e que tecnologias foram usadas em prol de conseguir alcançar estes objectivos.

De uma forma geral, foram também representadas a arquitectura do sistema, os vários casos de uso que podem ser realizados pelo utilizador na interacção com a aplicação Web bem como que opções tecnológicas foram adoptadas para desenvolvimento da aplicação e respectivas ferramentas de suporte à sua edição.

Finalizada a descrição do mapeamento NSDL para NS-3, no próximo capítulo serão apresentados alguns cenários de redes modelados com a linguagem NSDL e que comprovam a representatividade e validade desta no contexto de autoria de cenários de redes.

7. ESTUDO DE CASO

Os cenários de redes apresentados neste capítulo têm por finalidade demonstrar como o NSDL é capaz de especificar diversos tipos de cenários de redes atendendo a aspectos como diferentes topologias, tecnologias, número de nós, etc. Também é objecto de estudo, a comparação de resultados (*outputs*) gerados por ambas plataformas de simulação de redes, Network Simulator 2 (NS-2) e Network Simulator 3 (NS-3).

7.1. INTRODUÇÃO

De forma a confirmar minimamente a veracidade dos requisitos apresentados na secção 5.1, é necessário desenvolver um método de validação. Para este efeito, foram criados diversos cenários alusivos às tecnologias presentes no actual modelo de tradução NSDL -> NS3 de forma a comparar variáveis que são frequentemente alvo de estudo por parte de autores de cenários de redes, sejam estes para simulação ou até para implementação no mundo real. De entre essas variáveis, destacam-se o atraso (*delay*), a variação de atraso (*jitter*) e a quantidade de pacotes perdidos (*loss*) durante as transmissões de dados.

Estas variáveis mencionadas anteriormente foram observadas numa óptica comparativa entre o desempenho de um determinado cenário no ambiente NS-2 e o mesmo cenário (análogo em termos de especificação) em NS-3 de forma a verificar e analisar as diferenças verificadas por ambos os simuladores concorrentes.

Os casos de estudo são apresentados na seguinte ordem: Cenários 1 ao 4 seguem a abordagem comparativa mencionada anteriormente. Enquanto que o cenário 1 trata-se de um pequeno cenário que utiliza apenas 2 nós comunicantes, os cenários 2, 3 e 4 foram elaborados por grupos de alunos da cadeira Tecnologias Avançadas de Redes (TAR) do ano lectivo 2009/2010 leccionada pelos docentes Prof.ª Dr.ª Laura Rodríguez, pelo Prof. Dr. Paulo Sampaio e com formação adicional por parte do Prof. Eduardo Marques relativamente ao NSDL. A escolha destes cenários foi feita usando critérios como criatividade dos cenários apresentados, perspectiva realística ou seja, se seriam definições de redes passíveis de implementação no mundo real e complexidade do cenário (entenda-se por complexidade a intersecção de elementos como a topologia de rede adoptada, quantidade de nós, tipos de interfaces, quantidade e diversidade de fluxos de dados, etc.).

Já os cenários 5, 6 e 7 têm por principal finalidade a de demonstrar como a abordagem ao mapeamento NSDL -> NS3, descrita no capítulo 5, é capaz de suportar tecnologias emergentes como é o caso de tecnologias como Wi-Fi (cenário 5), WiMAX e LTE (cenários 6 e 7).

É de salientar que estes cenários, apresentados de seguida, além de servirem para validar o mapeamento entre XML e C++ realizado, contribuíram também para a depuração de alguns erros encontrados no processo de mapeamento, nomeadamente

pormenores como definições de aplicações e abordagem ao posicionamento dos nós no âmbito da visualização.

7.2. DESCRIÇÃO DA METODOLOGIA DE ESTUDO

Para realizar o estudo dos vários cenários apresentados anteriormente, foram utilizadas ferramentas que permitem a extracção de dados relevantes nos ficheiros de *output* gerados em tempo de execução pelos *scripts* detentores dos cenários de redes.

Segue-se uma descrição sobre os métodos utilizados para a obtenção de dados amostrais para ambos os simuladores NS-2 e NS-3, respectivamente.

7.2.1. DADOS NS-2

Em NS-2 foram utilizados os ficheiros de trace (.tr) que com auxílio de *scripts* AWK [Close, 1995][Ed-Dbali, 2011] específicos, foi possível extrair os dados para as amostras apresentadas nos gráficos referentes ao ambiente NS-2.

Os ficheiros de trace permitem descrever eventos ocorridos ao longo da simulação como por exemplo que pacotes foram enviados, que pacotes foram descartados e em que nó, em que instante é realizada a transmissão nó fonte/nó destino, etc. Já os ficheiros AWK permitem operar sobre os ficheiros do tipo .tr e extrair apenas os dados que interessam, bem como realizar operações sobre os mesmos.

Foram basicamente utilizados 3 ficheiros para a extracção dos dados de perdas de pacotes (*measure-loss.awk*), atraso (*measure-delay.awk*) e *jitter* (*measure-jitter.awk*) - que se encontram no CD da tese (Encontra-se na pasta "*Estudo de Caso*"), de forma a extrair somente os dados que interessam (neste caso os eventos associados à *flag* "r" que denota *reception*) e então efectuar o cálculo das variáveis *delay* e *jitter*.

Para o atraso, o cálculo utilizado foi o seguinte:

- Quando um pacote é recepcionado, no seu destino final (*destination*) é registado um evento com *flag* "r" e o instante (em segundos (s)) em que este evento ocorre é denominado por *end_time[packet_id]* ou seja, o tempo que demorou ao pacote *X* ser enviado da fonte (*source*) ao seu destino (*destination*). A este tempo subtraem o tempo de início do envio (*start_time[packet_id]*) e obtemos então um conjunto de atrasos individuais:

  start = start_time[packet_id];
  end = end_time[packet_id];
  packet_duration = end - start;

E este cálculo é efectuado para todos os pacotes transmitidos na rede para posterior inclusão num ficheiro *trace* intermédio que servirá para popular as tabelas de

Excel.

Já para o cálculo do *jitter*, este utiliza o mesmo cálculo utilizado para o *delay* individual de cada pacote, definindo assim o último *delay* obtido;

Posteriormente são comparados os valores de início de transmissão do pacote e término do mesmo:

if (start < end) {

O que obviamente será sempre verdadeiro devido aos somatórios dos atrasos intermédios existentes no transporte de um pacote (processamento do pacote, transmissão do mesmo no canal físico, etc.) e então é calculada a diferença entre o *atraso* actual a ser calculado, com o último atraso:

Cálculo do último atraso:
delay_diff = packet_duration - last_delay;

Cálculo do atraso actual:
seqno_diff = pkt_seqno[packet_id] - last_seqno;

Caso *seqno_diff* seja nulo, significa que não houve um atraso associado à última transmissão logo o *jitter* é de 0; Caso contrário o *jitter* será calculado seguindo a seguinte expressão:

jitter = delay_diff/seqno_diff;

De uma forma geral, a fórmula matematicamente correcta usada para o cálculo do *jitter* é a seguinte [Junqueira, 2007]:

```
jitter=(((packet_receive_time_array[i])-(packet_send_time_array[i]))-
((packet_receive_time_array[j])-(packet_send_time_array[j])))/(i-j);
```

Ou seja, trata-se do mesmo cálculo apresentado acima explicado em detalhe.

7.2.2. DADOS NS-3

Os dados obtidos a partir do NS-3 foram gerados em tempo de simulação, utilizando para tal, funções próprias desenvolvidas com o intuito de obter os principais dados necessários a uma correcta e justa comparação entre as plataformas NS-2 e NS-3.

Estes dados são sobretudo registos de eventos nas aplicações geradoras de tráfego e nos receptores do mesmo, com eventos do tipo “transmitido – TX” e “recepcionado – RX” respectivamente, com o registo adicional do instante em tempo de execução em que este evento ocorreu.

Adicionalmente foi utilizada a geração de um ficheiro XML com alguns dados estatísticos que permitem verificar alguns dos dados obtidos, bem como endereços de

fontes/destinos. A geração deste ficheiro utiliza o método desenvolvido pelos investigadores Gustavo Carneiro, Pedro Fortuna e Manuel Ricardo [Fortuna, 2010] em que um módulo denominado por *FlowMonitor* (já integrado no NS-3) extrai dados relativos aos fluxos de dados criados no cenário a ser simulado e guarda-os num ficheiro XML para posterior estudo. Entre as variáveis medidas e registadas neste ficheiro, destacam-se as seguintes:

- ***timeFirstTxPacket*** (o momento exacto em que o primeiro pacote de um determinado fluxo foi enviado);
- ***timeLastTxPacket*** (momento em que fora transmitido o último pacote de um determinado fluxo);
- ***jitterSum*** (mede o somatório dos atrasos referentes a um determinado fluxo desde o momento da sua geração até à sua entrega no respectivo destino ou descarte);
- ***lastDelay*** (atraso ponto-a-ponto), e;
- ***lostPackets*** (constitui o valor de pacotes perdidos de um determinado fluxo).

7.3. CENÁRIOS DE REDES

Tal como foi mencionado na Secção 3 do Capítulo 6 (Casos de utilização), a autoria de cenários de redes pode ser realizada com auxílio de várias ferramentas. No caso concreto dos cenários aqui apresentados, os cenários 1 a 4 foram elaborados graficamente com auxílio da ferramenta VND desenvolvida pelo colega Jesuíno Azevedo no âmbito da sua dissertação de Mestrado em Engenharia Informática [Azevedo, 2010]. Esta ferramenta permite que o utilizador modifique o conjunto de objectos (e respectivos atributos) no ficheiro de configuração de *input* (objectos disponibilizados), pelo que a modelação de cenários não conhece qualquer limitação. Infelizmente a ferramenta não suporta a geração textual NSDL para qualquer tipo de tecnologia, nomeadamente tecnologias que foram alvo da abordagem apresentada neste trabalho para o mapeamento de NSDL -> NS-3. Neste sentido, apenas tecnologias desenvolvidas por ambos os simuladores NS-2 e NS-3 são desenvolvidas no VND para posterior mapeamento.

Uma vez adquirido o ficheiro NSDL contendo o cenário desenvolvido, este poderá ser traduzido para NS-2/NS-3, conforme as necessidades do utilizador. É precisamente esta abordagem que foi seguida para os cenários 1 a 4 pois possuem tecnologias compatíveis no contexto da *Framework* NSDL. Já os cenários 5 a 7 abordam exclusivamente a sua autoria em ferramentas de codificação textual (qualquer editor XML) visando a sua simulação somente no NS-3.

O estudo referente a cada um dos cenários a ser apresentados inicia-se com uma pequena descrição de topologia do cenário e que responsabilidades aplicacionais detêm cada um dos nós mais importantes, acompanhada de uma figura e uma tabela que ilustra a topologia usada no cenário e as especificações das aplicações utilizadas respectivamente.

Por fim, são criados gráficos com amostras comparativas e elaborada uma pequena conclusão que inclui uma apreciação sobre os resultados obtidos.

A especificação de todos os cenários apresentados encontram-se no CD da tese, pasta denominada por "Cenário *X* (*X* de 1 a 7)" a partir da pasta "*Estudo de Caso*" (ambas as codificações em NSDL e NS-3).

7.3.1. CENÁRIO 1

Este cenário consiste no caso mais simples: apenas dois nós em que existe um que desempenha o papel de servidor (***node0***) e o segundo de cliente (***node1***). No servidor são gerados dois fluxos de dados (***pareto1*** e ***exponential1***) que são enviados para o cliente.

Para facilitar o entendimento deste e dos restantes cenários, a sua duração é de 12 segundos, com as medições de ***Throughput***, ***Delay*** e ***Jitter*** a serem efectuadas em intervalos de 0,2 segundos.

Em relação às aplicações, os tempos de início da geração de tráfego é de 0.2 segundos para aplicações a funcionar sob TCP e 0.6 segundos para aplicações sob UDP; Já o término destas aplicações ocorre 10 segundos após ao início da simulação. Este aspecto também deve ser considerado nos restantes cenários.

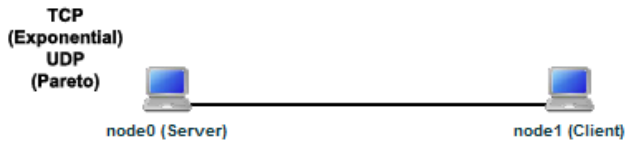

Figura 29 - Cenário 1 (sem abordagem a QoS)

Neste cenário não é considerado qualquer aspecto de Qualidade de Serviço (QoS), e mais detalhe sobre as aplicações usadas podem ser verificadas na Tabela 5, a seguir:

Tabela 5 - Requisitos das 2 aplicações do cenário 1

Aplicação	Agente utilizado	Tamanho dos pacotes (bytes)	Taxa de transmissão (Mbps)	Atraso (ms)
Pareto	UDP	X	X	X
WEB (Exponential)	TCP	X	X	X

Os resultados obtidos foram os seguintes:

A nível de perdas (***loss***) de acordo com os pacotes enviados e os recebidos, o resultado é o que se verifica na Tabela 6.

Tabela 6 - Pacotes enviados/perdidos do Cenário 1

		Pacotes enviados	Pacotes perdidos
NS-2	TCP	168	0
	UDP	180	0
NS-3	TCP	785	0
	UDP	534	0

Comparando os valores de fluxo efectivo (***Throughput***), o gráfico obtido foi o seguinte:

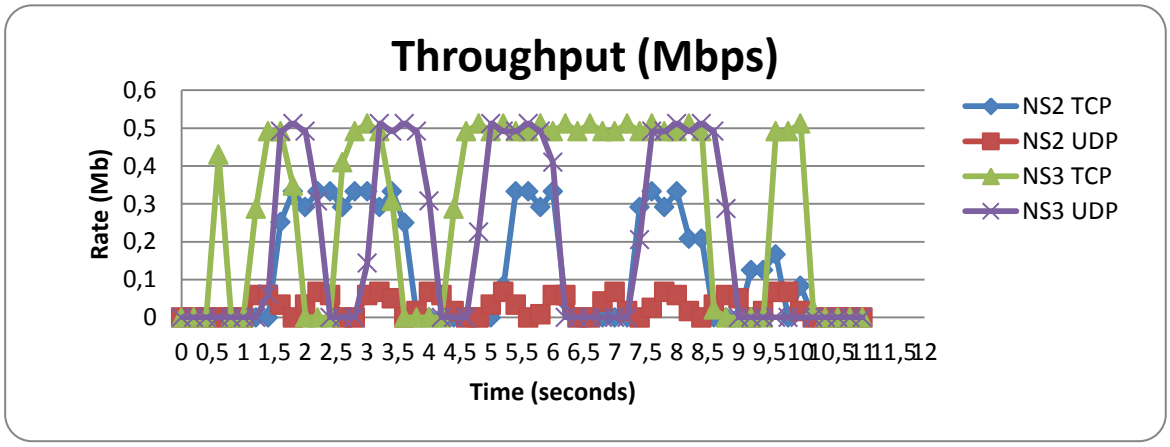

Gráfico 1 - Comparação entre Throughput em NS-2 e NS-3 (Cenário 1)

Finalmente os resultados alusivos aos valores do atraso (***delay***) foram os seguintes (auxílio ao ficheiro *measure-delay.awk e measure-delayNS3.awk*):

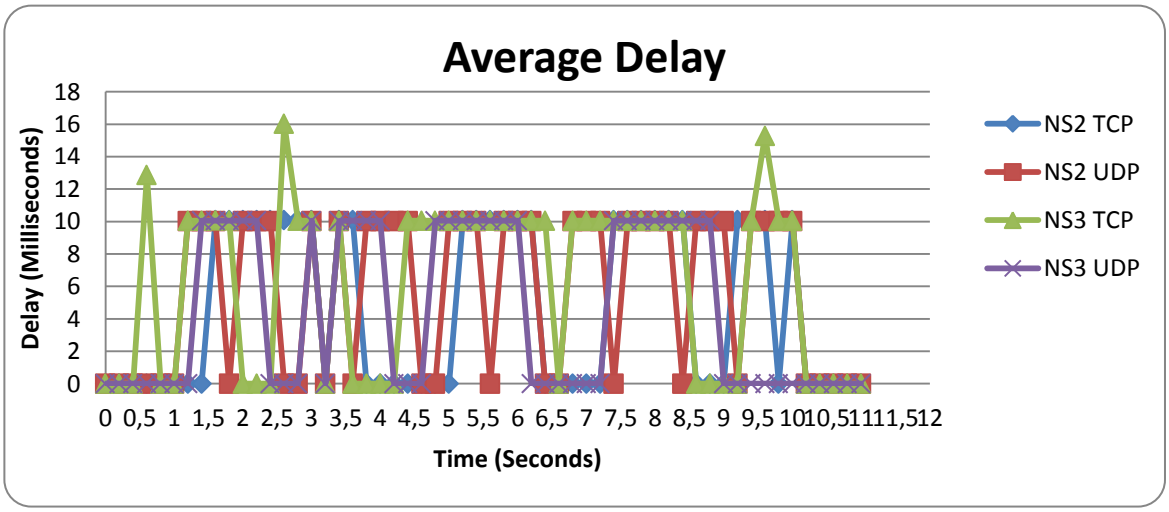

Gráfico 2 - Comparação entre delay em NS-2 e NS-3 (Cenário 1)

7.3.1.1. CONCLUSÕES ALUSIVAS AO CENÁRIO 1

Em termos de *perdas*, ambos os resultados obtidos deste cenário em NS-2 e NS-3 mostram que o número de pacotes perdidos é de **0** pacotes o que é compreensível visto tratar-se de um cenário muito simples de apenas dois nós conectados por um simples *link point-to-point*. Os resultados relativos ao ***throughput*** confirmam a maior quantidade de pacotes transmitidos em NS-3 comparativamente ao NS-2 bem como a confirmação da diferença do tamanho dos pacotes em ambos os ambientes por omissão: Um pacote UDP possui 210 bytes em NS-2 e 512 bytes em NS-3. Em geral, maiores valores de fluxo TCP foram observados em relação ao UDP. Por outro lado, os valores de ***delay*** foram consideravelmente semelhantes, com apenas alguns picos para o TCP em NS-3 que atingiu aos 2,6 segundos o valor máximo de 16 ms, o que é perfeitamente aceitável.

No gráfico de ***delay***, os valores da série a 0 significa que no intervalo de tempo em que se encontra, não foram transmitidos pacotes.

Para este primeiro cenário não foi apresentado o gráfico relativo aos valores de ***jitter*** pois as variações de atraso foram pouco significativas.

7.3.2. CENÁRIO 2

O cenário 2 foi desenvolvido pelos colegas Jorge Sousa e Jorge Gonçalves do Mestrado em Engenharia Informática no âmbito da disciplina Tecnologias Avançadas de Redes (TAR).

O principal objectivo com a realização deste cenário não foi o de proporcionar uma rede extremamente realista de uma empresa actualmente operacional, mas sim utilizar o maior número de tipos de agentes e de aplicações possível de modo a demonstrá-los, ou seja, utilizar ao máximo os conhecimentos adquiridos durante as aulas de TAR.

Para este cenário foi utilizada a topologia ilustrada na Figura 30 constituída por 4 servidores (computadores à esquerda) que geram fluxos de acordo com a Tabela 7. Estes fluxos são encaminhados por *switches* (1 a 4) para os *routers* centrais (1 e 2) e posteriormente para um outro conjunto de *switches* (5 a 10) que se encarregam de efectuar o *broadcast* do fluxo recebido para os computadores destino associados (*Hosts* de 1 a 6).

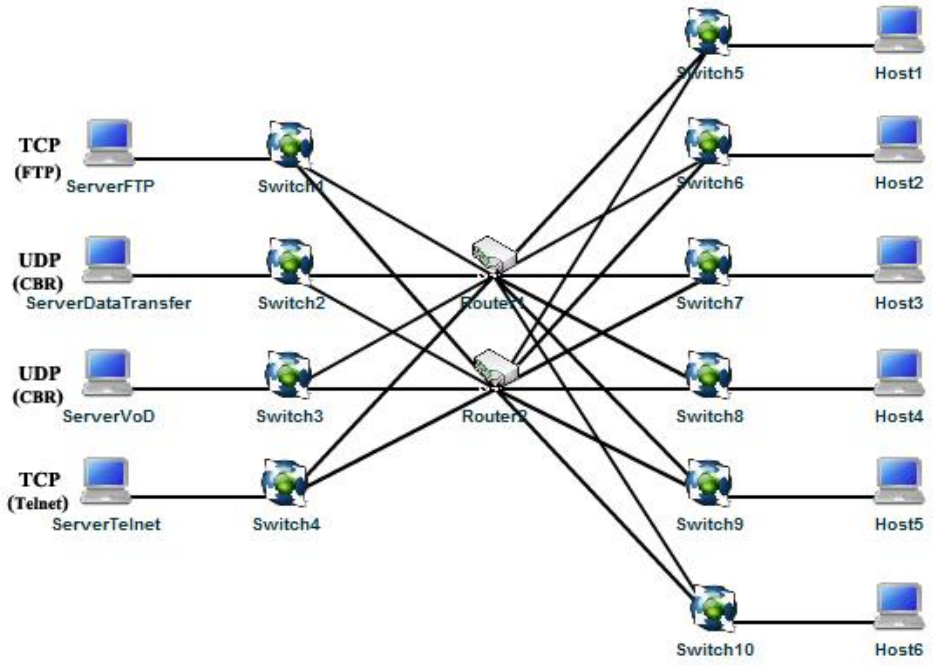

Figura 30 - Cenário 2 (Jorge Sousa e Jorge Gonçalves) sem QoS

A conectividade deste cenário é proporcionada por *links* de igual *data rate* (Taxa de transmissão de dados), correspondente a 100 Mbps e com *delay* (atraso) de 10 milissegundos (ms).

Tabela 7 - Requisitos das diversas aplicações do 2º cenário de rede

Aplicação	Agente utilizado	Tamanho dos pacotes (bytes)	Taxa de transmissão (Mbps)	Atraso (ms)
FTP	TCP	X	10	450
WEB (Transmissão de dados - CBR)	UDP	1000	5	150
Vídeo on Demand (CBR)	RTP	210	9	10
Telnet	TCP	X	10	400

Os fluxos gerados possuem uma descrição que utiliza o rótulo de aplicação / protocolo, a especificação dos principais atributos que descrevem o comportamento dos fluxos na rede criada. É realizada a definição de *packet size* (tamanho dos pacotes) para aplicações CBR, *data rate* (taxa de transmissão) e *delay* (atraso).

Após simulação dos cenários em NS-2 e NS-3, os resultados obtidos foram os ilustrados na Tabela 8 em termos de pacotes enviados e perdidos durante a transmissão.

Os Gráficos 3, 4 e 5 mostram os valores observados para as variáveis *Throughput*, *Delay* e estimação de *Jitter*, respectivamente.

Tabela 8 - Pacotes enviados/perdidos do Cenário 2

		Pacotes enviados	Pacotes perdidos	Pacotes perdidos (%)
NS-2	TCP	413937	196	0,05
	UDP	2272828	36	0,0002
NS-3	TCP	7176	24	0,33
	UDP	168693	720	0,43

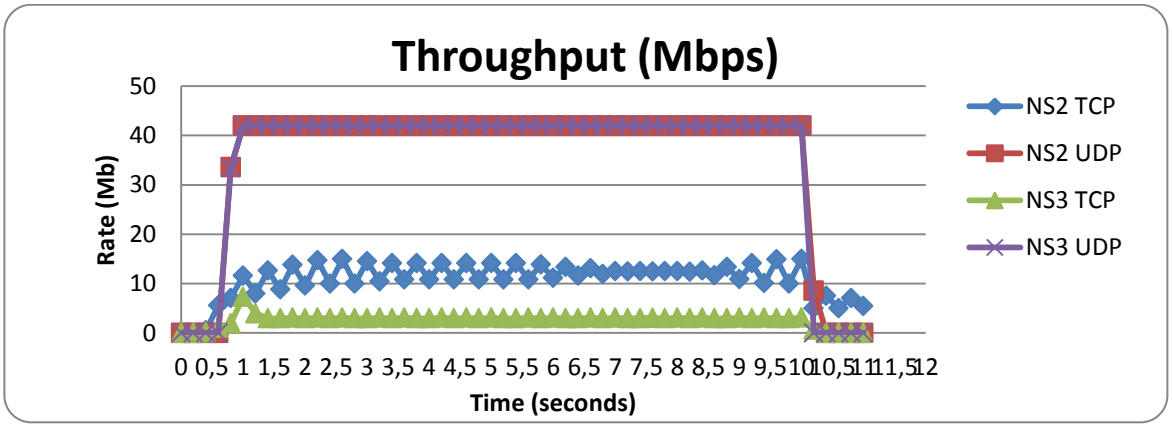

Gráfico 3 - Comparação entre Throughput em NS-2 e NS-3 (Cenário 2)

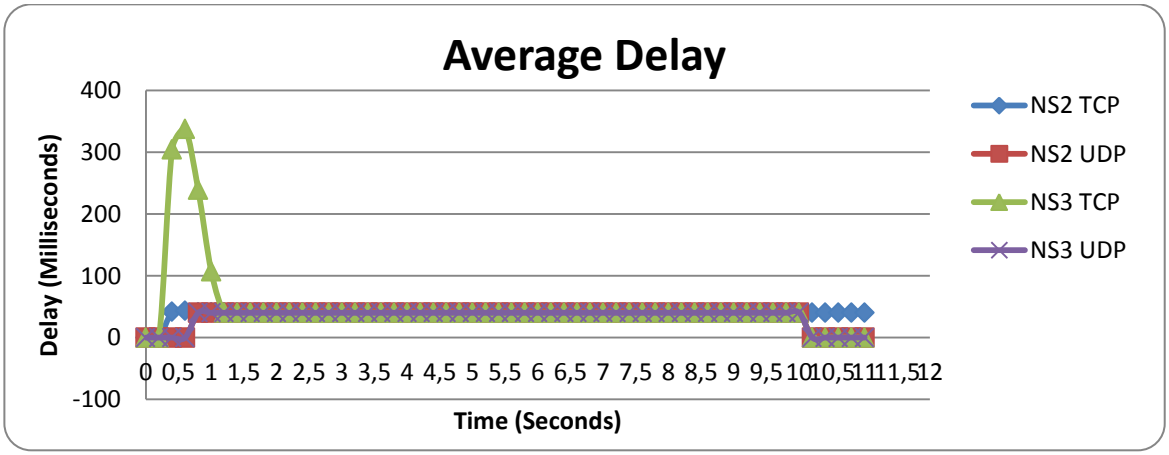

Gráfico 4 - Comparação entre delay em NS-2 e NS-3 (Cenário 2)

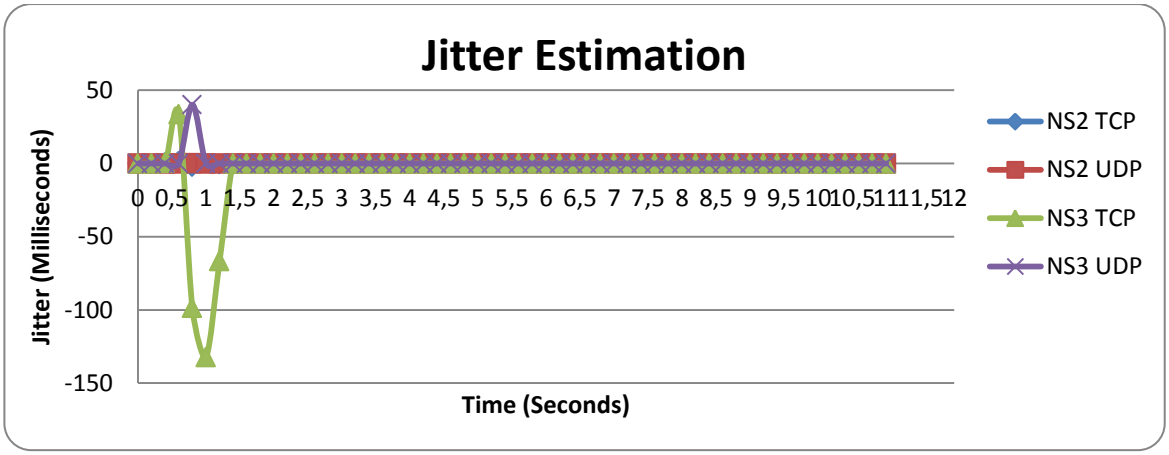

Gráfico 5 - Comparação entre Jitter em NS-2 e NS-3 (Cenário 2)

7.3.2.1. CONCLUSÕES ALUSIVAS AO CENÁRIO 2

Os dados obtidos através de este segundo cenário de redes vêm demonstrar uma vez mais a grande diferença de envio de pacotes entre o NS-2 e NS-3. Foi observado no primeiro cenário que o perfil "*burst*" de envio de pacotes era superior no NS-3 relativamente ao NS-2. Quando a quantidade de aplicações aumenta, a situação se inverte e neste segundo cenário é claramente o NS-2 que possui maior número de pacotes enviados.

Relativamente às perdas, estas foram aceitáveis tendo em conta a quantidade de tráfego gerado. No entanto, observamos um grande número de perdas para aplicações sob UDP no NS-3. Uma possível explicação para este facto seria a necessidade de conhecimento de rotas que é determinada pelo *IPv4GlobalRouting* em tempo de execução e no processo são descartados em dispositivos que ainda não possuem as tabelas de encaminhamento devidamente populadas.

Em relação ao *throughput* para os tráfegos TCP, é nítido o impacto de no NS-3 não ser ainda possível definir um intervalo (***interval***) para novo envio de pacotes, o que torna os valores de *throughput* ligeiramente superiores no NS-2. Por outro lado, a actual possibilidade de estipulação de tamanho de pacote (***PacketSize***) e de taxa de transmissão a usar (***DataRate***) para UDP, permite a verificação de valores semelhantes para o throughput de UDP, rondando os 42 Mbps após início das aplicações por volta dos 0,6 segundos. Os 42 Mbps são facilmente explicáveis dado que temos 6 aplicações UDP, 3 das quais operam a 5 Mbps (total de 15 Mbps) e as restantes 3 operam a 9 Mbps (total de 27 Mbps + os 15 Mbps anteriores = 42 Mbps).

Os valores de atraso (*delay*) verificados são aceitáveis visto que o grande pico de atraso foi observado no início da geração de tráfego, atingindo os 337 ms para TCP em NS-3 mas posteriormente as 4 aplicações estabilizaram em torno dos 40 ms. Da mesma forma, os valores de *jitter* reflectem o grande valor de atraso observado nos instantes iniciais das aplicações, onde podemos ver uma grande variação de atraso por parte do TCP em NS-3 entre os 0,8 e 1,2 segundos de tempo de execução da simulação e os restantes

valores sempre muito próximos de 0 devidos à estabilização mencionadas do comportamento das aplicações.

7.3.3. CENÁRIO 3

O cenário 3 foi desenvolvido pelas colegas Branca Almeida e Nélia Fernandes, também do Mestrado em Engenharia Informática no âmbito da disciplina Tecnologias Avançadas de Redes (TAR). A topologia usada pelas colegas foi a que podemos observar na Figura 31.

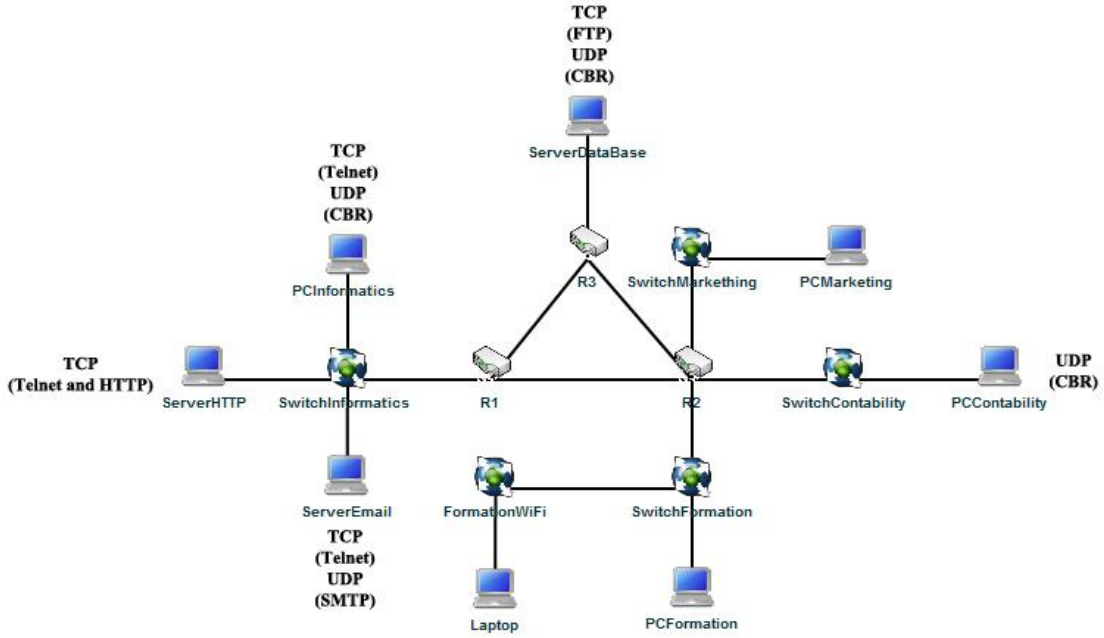

Figura 31 - Cenário 3 (Branca Almeida e Nélia Fernandes) sem QoS

Na Figura 31 é possível visualizar as combinações de aplicação / protocolo usadas na rede, bem como o conjunto de *hosts* receptores dos fluxos gerados que englobam não só os nós sem geração de tráfego (*Laptop*, *PCFormation* e *PCMarketing*) mas também os restantes nós, ou seja, qualquer nó é receptor dos fluxos gerados.

Com o desenvolvimento deste cenário, as colegas tinham como principal objectivo verificar o comportamento dos vários tipos de tráfego em relação à topologia implementada, nomeadamente ao nível de variáveis tais como *bandwidth* (largura de banda) definida, *delay* (atraso) e *jitter* (variação dos atraso).

Na Tabela 9, temos em detalhe a especificação dos principais atributos relativos às aplicações usadas neste cenário, à semelhança do que tínhamos em relação aos cenários até então apresentados.

Tabela 9 - Requisitos das diversas aplicações do 3º cenário de rede

Aplicação	Agente utilizado	Tamanho dos pacotes (bytes)	Taxa de transmissão (Mbps)	Atraso (ms)
FTP	TCP	200	3 Mbps	450 ms
Video on Demand	UDP	500	5 Mbps	7ms
Web	UDP	200	1 Mbps	30 ms
Mail	TCP	200	1 Mbps	400 ms
Telnet	TCP	200	0,5 Mbps	40 ms
VoIP	UDP	300	2 Mbps	10 ms
Video Conferência	UDP	500	4 Mbps	10 ms

Na sequência da simulação deste cenário utilizando as suas especificações em NS-2 e em NS-3, os resultados obtidos foram os seguintes:

Tabela 10 - Pacotes enviados/perdidos do Cenário 3

		Pacotes enviados	Pacotes perdidos	Pacotes Perdidos (%)
NS-2	TCP	37524	0	0
	UDP	182183	0	0
NS-3	TCP	48996	4167	8,5
	UDP	40730	167	0,4

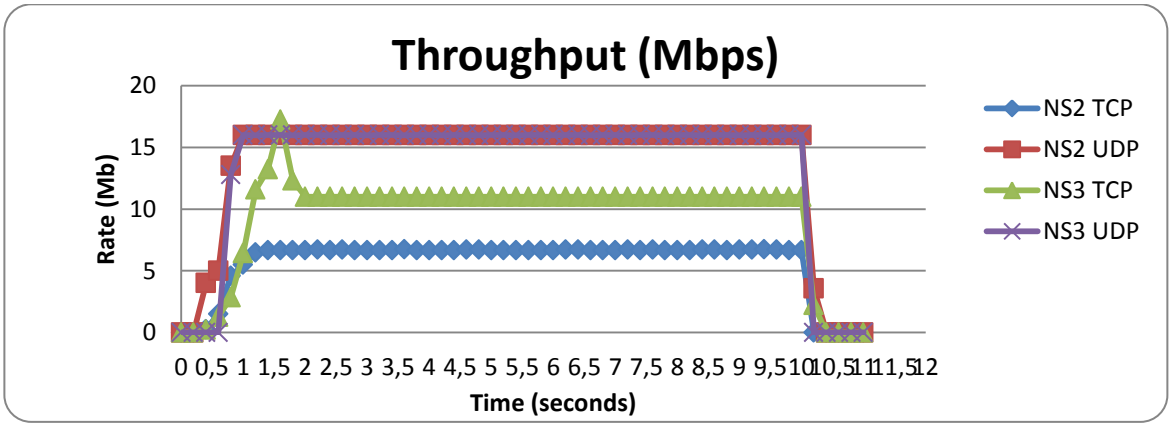

Gráfico 6 - Comparação entre Throughput em NS-2 e NS-3 (Cenário 3)

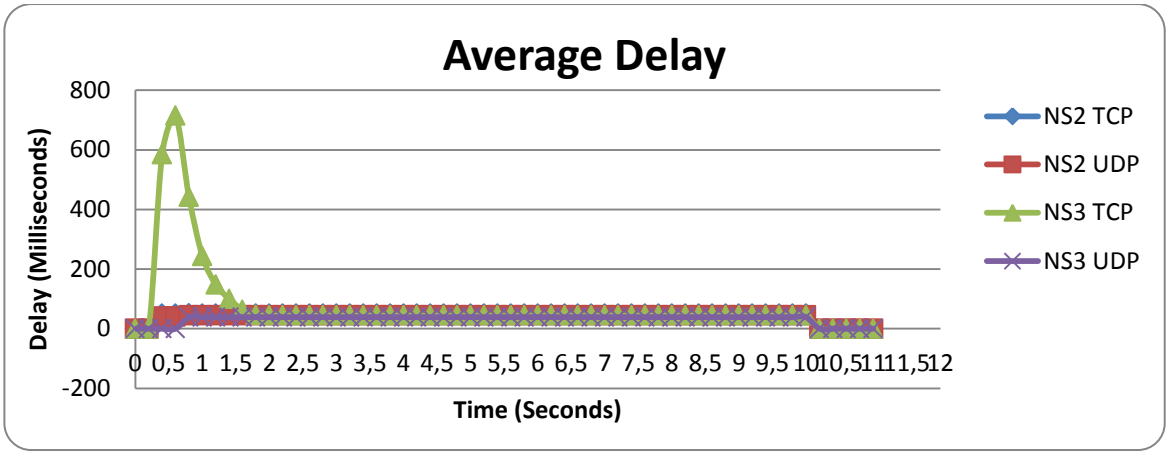

Gráfico 7 - Comparação entre delay em NS-2 e NS-3 (Cenário 3)

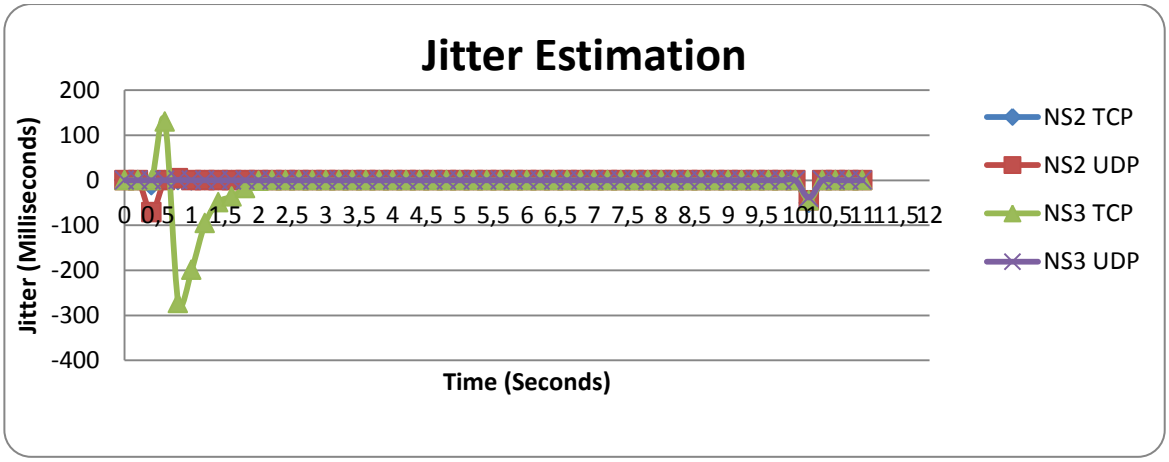

Gráfico 8 - Comparação entre jitter em NS-2 e NS-3 (Cenário 3)

7.3.3.1. CONCLUSÕES ALUSIVAS AO CENÁRIO 3

Relativamente a este terceiro cenário, os valores de perdas foram nulos no ambiente NS-2, já para o NS-3 as perdas foram de 8,5 % (TCP) e de 0,4 % (UDP).

Os valores observados de *throughput* comprovam uma vez mais a grande similaridade dos módulos do protocolo UDP existentes no NS-2 e NS-3. Os valores de UDP rondam os 16 Mbps como esperado. Por outro lado, o tráfego TCP foi superior para o NS-3, rondando os 12 Mbps enquanto que no NS-2 os valores rondaram os 6,1 Mbps.

Para o *delay* foram observados valores aceitáveis para a transmissão dos diversos fluxos de dados, tendo as aplicações sob UDP apresentado valores de atraso entre os 38 e 44 ms. Já os fluxos sob TCP, após estabilização dos momentos iniciais, apresentaram valores compreendidos entre os 44 e os 51 ms.

Em função dos valores de atraso, os valores de *jitter* apresentam também alguma instabilidade nos intervalos de tempo iniciais e posteriormente apresentando valores estáveis a partir dos 1,8 segundos de simulação.

7.3.4. CENÁRIO 4

O cenário 4 foi desenvolvido pelos colegas Nélio Sousa e Fernando Rodrigues do Mestrado em Engenharia Informática no âmbito da disciplina Tecnologias Avançadas de Redes (TAR). Para este 4º cenário, os colegas usaram uma abordagem semelhante ao do cenário anterior relativamente à topologia de rede em que temos 3 *routers* centrais responsáveis pelo reencaminhamento do tráfego e, por cada *router*, a associação de um conjunto de redes LAN (à excepção do servidor de *Backup e Database*).

Os principais requisitos a serem respeitados com este cenário foram:

- Desenvolvimento de uma rede de alta velocidade;
- Identificação e especificação dos principais parâmetros de rede;
- Para o cenário modelado, este deverá possuir vários tipos de tráfego, com diferentes requisitos em termos de *data rate*, *delay* e *jitter*.

Mais especificamente, para os fluxos utilizados na rede, temos basicamente 5 servidores principais (*WksRedaction03*, *WksStudio*, *WksRealization*, *ServerMultimedia* e *ServerWebFTP*) que utilizam as combinações aplicação/protocolo que podemos visualizar na Figura 32.

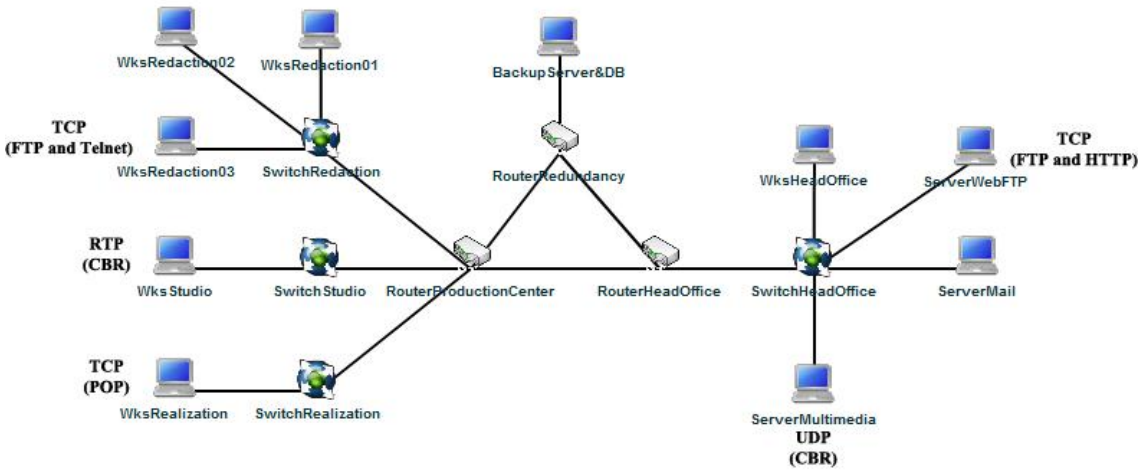

Figura 32 - Cenário 4 (Nélio Sousa e Fernando Rodrigues) sem QoS

Na Tabela seguinte é possível visualizar um maior detalhe na especificação de cada fluxo em que são definidos valores para os principais atributos caracterizadores do comportamento do tráfego em questão.

Tabela 11 - Requisitos das diversas aplicações do 4º cenário de rede

Aplicação	Agente utilizado	Tamanho dos pacotes (bytes)	Taxa de transmissão (Mbps)	Atraso (ms)
POP/Mail-FTP	TCP	10	0.1	500
CBR/Video-on-Demand	UDP	500	7.5	6
CBR/Vídeo RT/HDTV	UDP	1000	5	3
Telnet	TCP	50	0.1	60
FTP	TCP	50	2	500
HTTP-Tráfego Exponencial	TCP	800	2.5	50
FTP1	TCP	50	1.5	500

Com base nos dados apresentados, os resultados obtidos na simulação NS-2/NS-3 foram os seguintes:

Tabela 12 - Pacotes enviados/perdidos do Cenário 4

		Pacotes enviados	Pacotes perdidos	Pacotes Perdidos (%)
NS-2	TCP	50603	33879	66,9
	UDP	301112	0	0
NS-3	TCP	102066	7557	7,4
	UDP	20561	109	0,53

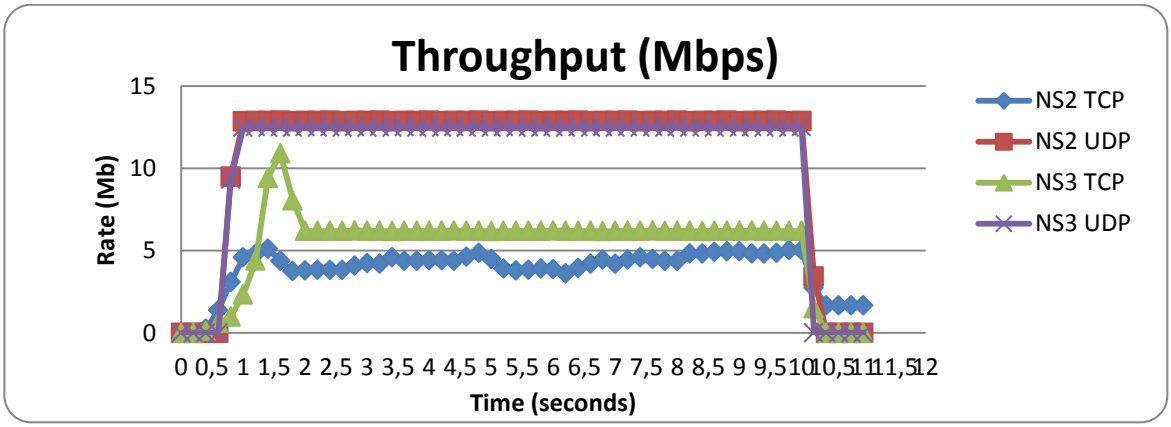

Gráfico 9 - Comparação entre Throughput em NS-2 e NS-3 (Cenário 4)

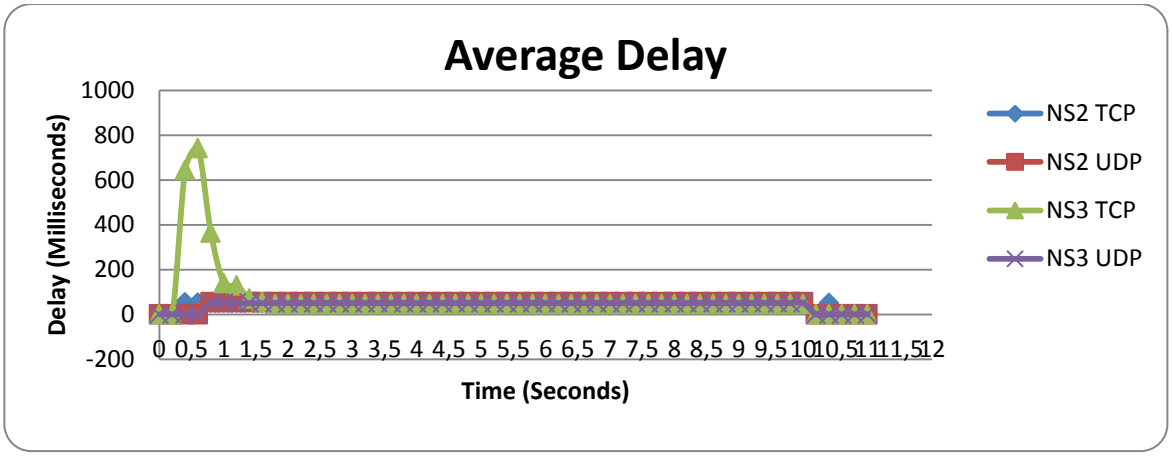

Gráfico 10 - Comparação entre delay em NS-2 e NS-3 (Cenário 4)

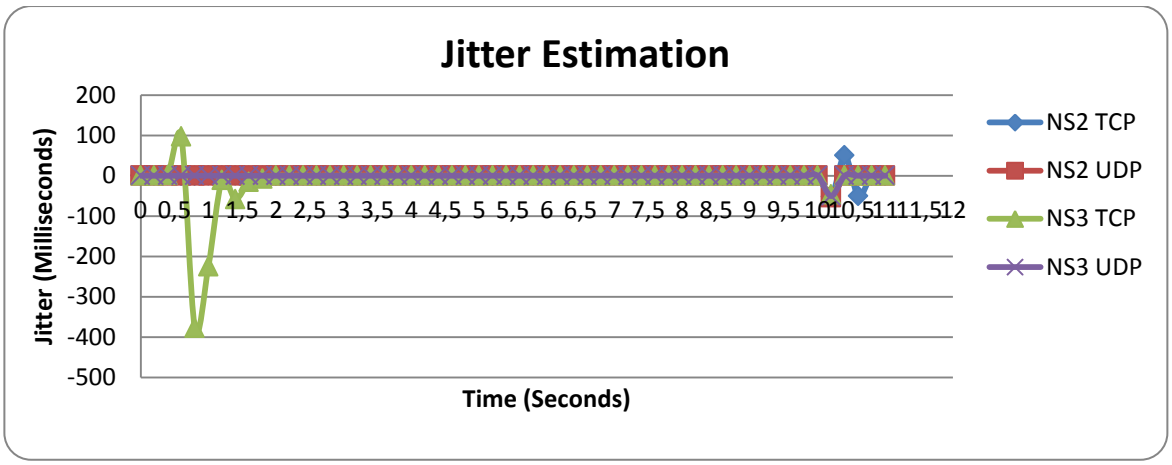

Gráfico 11 - Comparação entre jitter em NS-2 e NS-3 (Cenário 4)

7.3.4.1. CONCLUSÕES ALUSIVAS AO CENÁRIO 4

Perante os resultados apresentados, é possível concluir que os fluxos UDP apresentam comportamentos semelhantes em ambos os simuladores NS-2 e NS-3. Ao contrário dos cenários anteriores, em que os valores de *throughput* de UDP foram praticamente coincidentes, neste quarto cenário foi ligeiramente inferior no NS-3 devido sobretudo às perdas observadas.

Também em relação às perdas, estas foram significativamente altas para os fluxos TCP em NS-2 em que as causas plausíveis para este facto são as de estipulação de filas com tamanho fixo de 40 pacotes por fila e de custos associados a determinadas rotas, definidos pelos colegas de TAR (nomeadamente entre as ligações existentes nos *routers*).

Como tem sido frequente nos cenários anteriores, o fluxo TCP em NS-3 apresenta nos momentos iniciais de simulação grandes valores de *delay* (aos 0,6 segundos chegou a atingir os 741 ms de atraso) e cerca de 1,2 segundos após início da simulação, estabiliza para os 46,5 ms. O mesmo fluxo em NS-2 apresenta valores de atraso

ligeiramente superiores comparativamente ao TCP NS-3, na ordem dos 50 aos 52 ms. Já no que diz respeito aos fluxos UDP, foram observados atrasos na ordem dos 54 ms em NS-2 e 50 ms em NS-3.

A nível de *jitter* podemos observar a grande variação de atraso no intervalo de 0,6 a 1,2 segundos, seguindo-se valores na ordem dos micros segundos (µs) que não constituem um grande impacto na performance da rede.

7.3.5. CENÁRIO 5 (Wi-Fi)

Para este quinto cenário foi adoptada uma topologia de rede em malha com um considerável número de nós quando comparado com os cenários anteriores. A topologia de rede bem como as várias aplicações existentes na rede podem ser observadas na Figura 33.

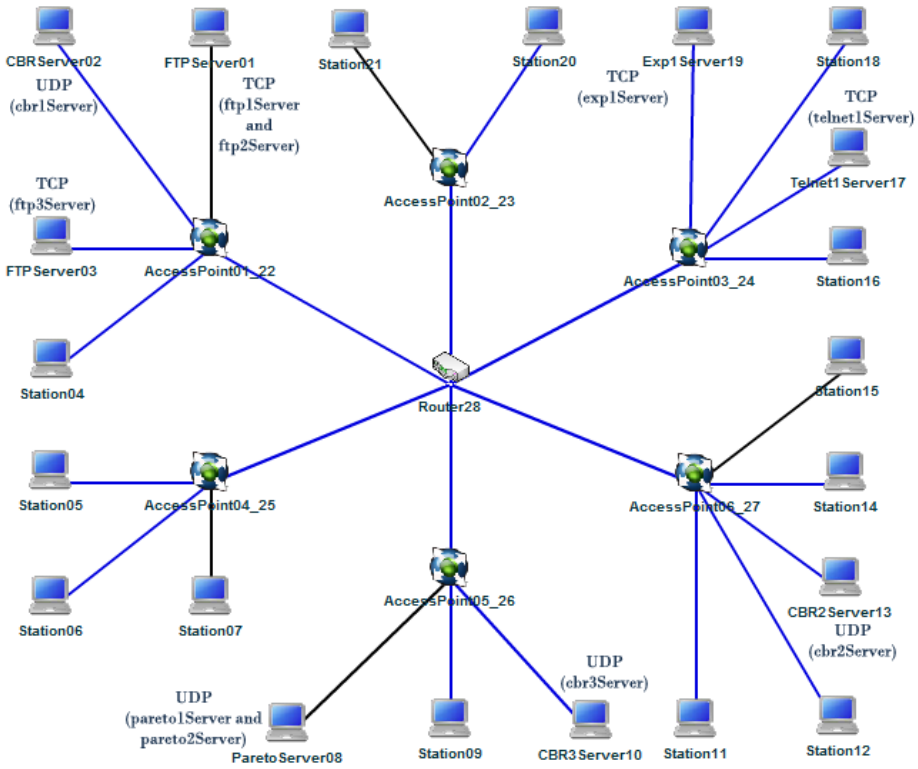

Figura 33 - Cenário 5 (Wi-Fi)

Em termos de *links*, estes são maioritariamente definidos por um *YansChannel* e por *links P2PChannel* e *CSMAChannel*, estes dois últimos com 100 Mbps de *DataRate*, 10 ms de atraso para *P2PChannel* e 20 ms para *CSMAChannel*.

Relativamente à topologia, este cenário é constituído por um *router* central (***Router28***) que está ligado virtualmente a 6 *Access Points* (APs) que por sua vez comunicam com um conjunto de máquinas finais (*non-APs/STAs*) por *links* virtuais (*YansChannel*) ou físicos (*P2P* e *CSMA Channels*).

As aplicações presentes na Figura 33 não possuem qualquer especificação de atributos, pelo que a sua instanciação utiliza os valores por omissão. O tempo de início e término das aplicações segue a abordagem até então seguida de 0,2 e 0,6 segundos para o início das aplicações TCP e UDP, respectivamente, e a paragem de geração de tráfego para ambos os fluxos dá-se aos 10 segundos. A simulação geral termina aos 12 segundos.

Este cenário e os seguintes não seguem a óptica comparativa mencionada na introdução, pelo que apenas é pretendido demonstrar o suporte do mapeamento desenvolvido entre NSDL e a plataforma NS-3 para tecnologias como o Wi-Fi neste caso concreto.

Como exemplos de dados que podemos extrair temos os presentes nos Gráficos 12, 13 e 14. Através do Gráfico 12 podemos verificar que o *standard* que melhores valores de *throughput* regista é o *80211a* acompanhado do standard *holland* em que ambos apresentaram dados muito próximos. Por outro lado, como *standard* com menor performance, o *standard 5Mhz* que não chegou a ultrapassar os 3,4 Mbps.

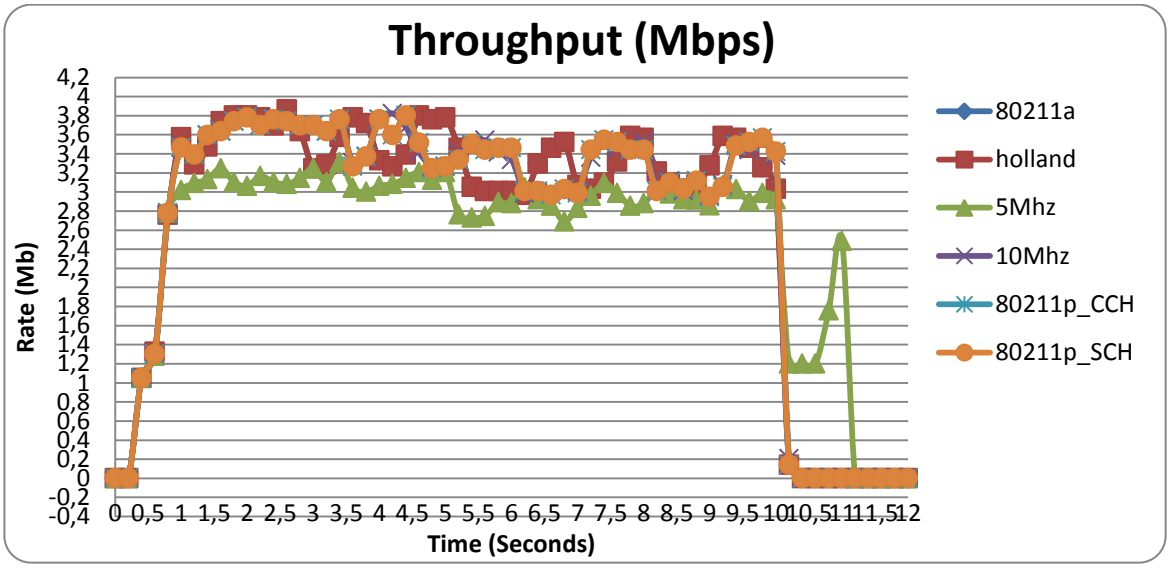

Gráfico 12 - Throughput por Standard do modelo Wi-Fi (Cenário 5)

Relativamente à comparação do *throughput* por algoritmo de gestão (Gráfico 13), todos apresentaram valores muito semelhantes com destaque para o algoritmo *MINSTREL* que no intervalo compreendido entre os 7 e os 9 segundos apresentou uma taxa de transmissão estável nos diversos *PacketSinks* do cenário de rede.

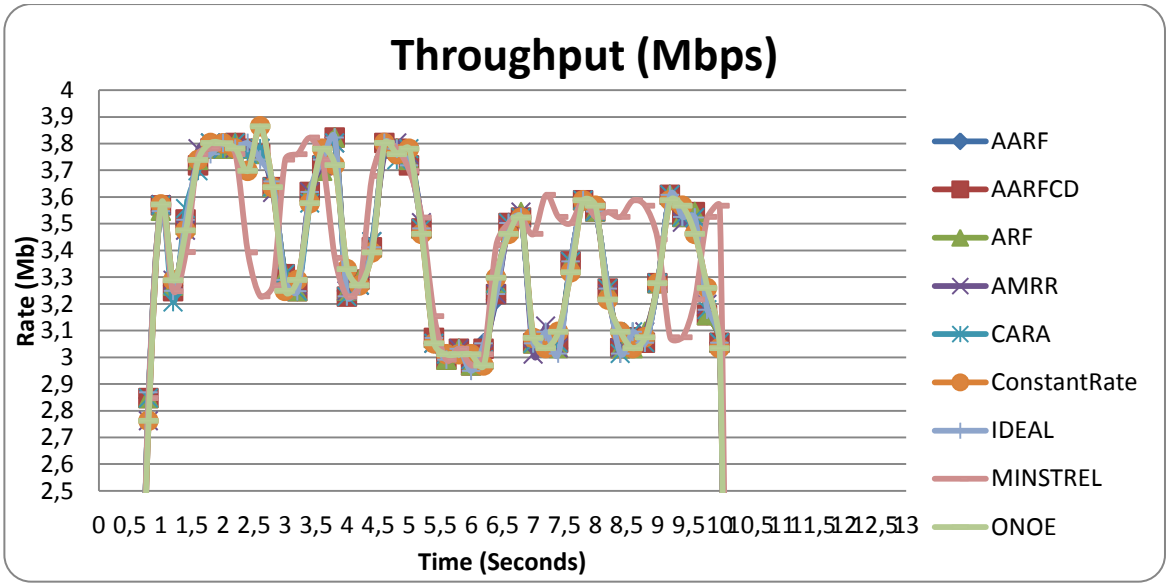

Gráfico 13 - Throughput for algoritmo de gestão do modelo Wi-Fi (Cenário 5)

Outros dados interessantes que podem ser medidos através deste quinto cenário, são a força (em decibéis), o sinal (*signal*), o respectivo ruído (*noise*) e a relação sinal- -ruído (*Signal-to-noise* - SNR). Podemos constatar a diferença entre o sinal que teoricamente deveria ser registado (SNR – *Signal-Noise*) e aquele que foi medido em tempo de execução (SNR – Real).

Para o intervalo de tempo estável da simulação, o SNR real é de 30 a 40 dBm o que nos leva a concluir que este cenário seria perfeitamente passível de introdução no mundo real visto que são valores de SNR aceitáveis para uma rede Wi-Fi.

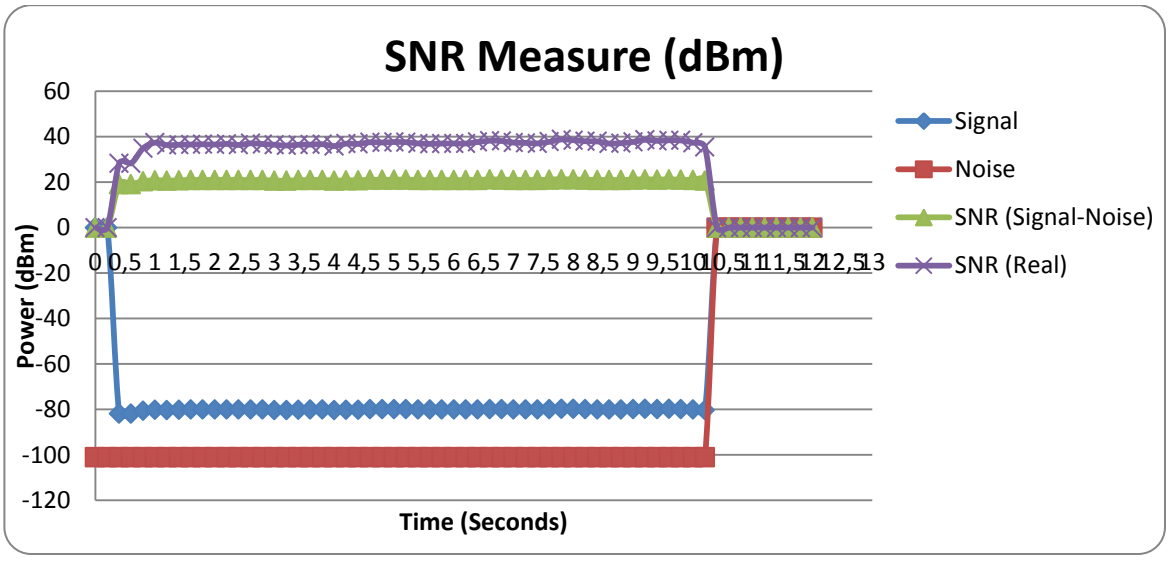

Gráfico 14 - Medida do SNR com base no Sinal (Signal) e Ruído (Noise) (Cenário 5)

7.3.6. CENÁRIO 6 e 7 (WiMAX Vs LTE)

Para estes dois últimos cenários de redes, foi adoptada a mesma topologia para ambas as tecnologias LTE e WiMAX (Figura 34). Podemos verificar que a topologia ilustrada consiste em um nó central (*BaseStation* em WiMAX e *Enhanced-NodeB* em LTE) que recebe e reencaminha tráfego entre os restantes nós (*SubscriberStations* em WiMAX e *UserEquipments* em LTE).

Relativamente às aplicações, apenas 1 aplicação (fonte ou destino) pôde ser definida por nó, dada a limitação em WiMAX por parte da classe que gere os fluxos de dados a transmitir (*ServiceFlow*) de lidar com múltiplos fluxos em 1 único nó. Sendo assim, existem 3 aplicações servidor e 3 clientes que também podem ser visualizados na Figura 34. As várias aplicações não possuem qualquer especificação de atributos logo os valores utilizados são os instanciados pela classe *OnOffApplication*.

Tal como em todos os cenários anteriores, as aplicações sob TCP iniciam aos 0,2 segundos enquanto que as aplicações sob UDP iniciam aos 0,6 segundos. O término das aplicações dá-se aos 10 segundos e a paragem da simulação aos 12 segundos.

Os *links* comunicantes diferem conforme a tecnologia: Em WiMAX o *link* utilizado foi o usado pela classe *SimpleOfdmWimaxChannel* enquanto que para o ambiente LTE o *link* utilizado foi *SingleModelSpectrumChannel*.

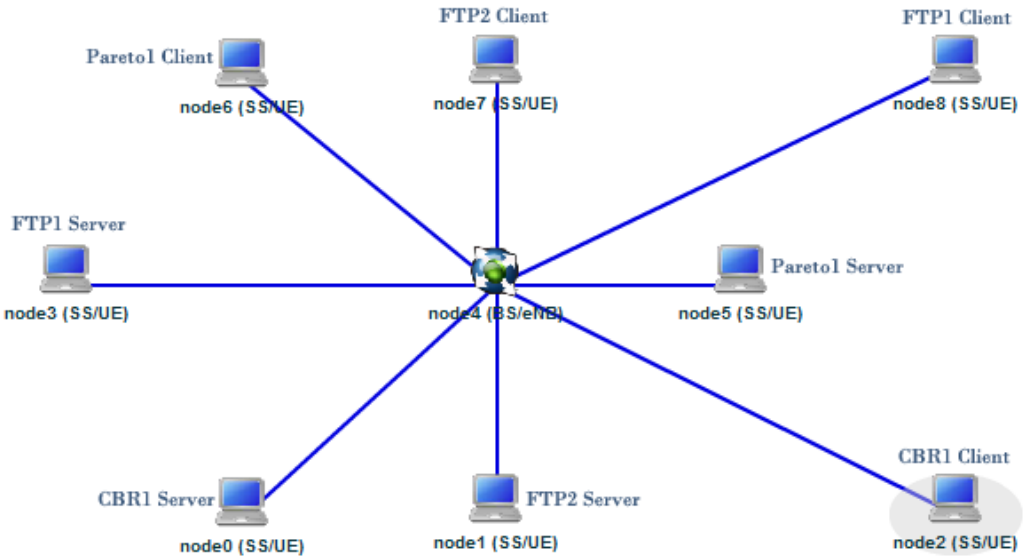

Figura 34 - Cenário WiMAX/LTE

Embora existam actualmente grandes limitações por parte do modelo desenvolvido para LTE no NS-3, nomeadamente a funcionalidade de uplink por parte dos UEs, é possível medir o throughput nos nós destino. Trata-se de uma medida em parte injusta pois os

fluxos LTE são gerados no eNB enquanto que os fluxos WiMAX são gerados em SSs e com destino a outros SSs logo percorrem 2 trajectos (SS->BS->SS) e em LTE apenas 1 trajecto é percorrido (eNB->EU). Este facto é visível no Gráfico 15 em que no cenário LTE foram registados valores de throughput superiores aos do WiMAX também porque este último ficou mais susceptível a perdas de pacotes como podemos constatar na Tabela 13.

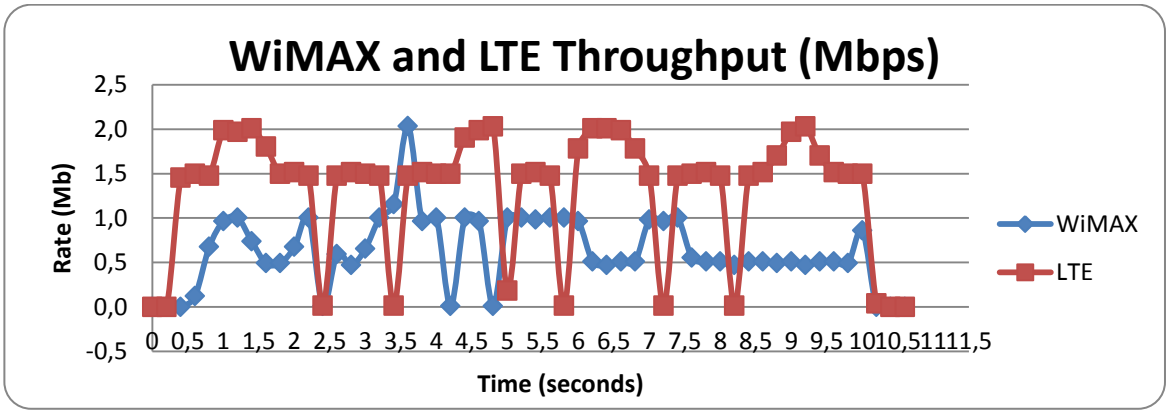

Gráfico 15 - Throughput para os ambientes WiMAX e LTE

Tabela 13 - Pacotes enviados/perdidos dos Cenários 6 e 7 (WiMAX e LTE)

	Pacotes enviados	Pacotes perdidos	Pacotes perdidos (%)
WiMAX	3819	2086	54,6
LTE	3880	0	0

Para qualquer comparação entre estes dois cenários, os resultados serão favoráveis ao cenário LTE pelas razões apresentadas anteriormente mas que surgem das limitações existentes por parte do LTE que de momento são incontornáveis, até que seja introduzido no NS-3 o projecto LENA [LENA, 2011] que está actualmente a ser parcialmente usado por utilizadores do NS-3 mas ainda não está oficialmente integrado na plataforma.

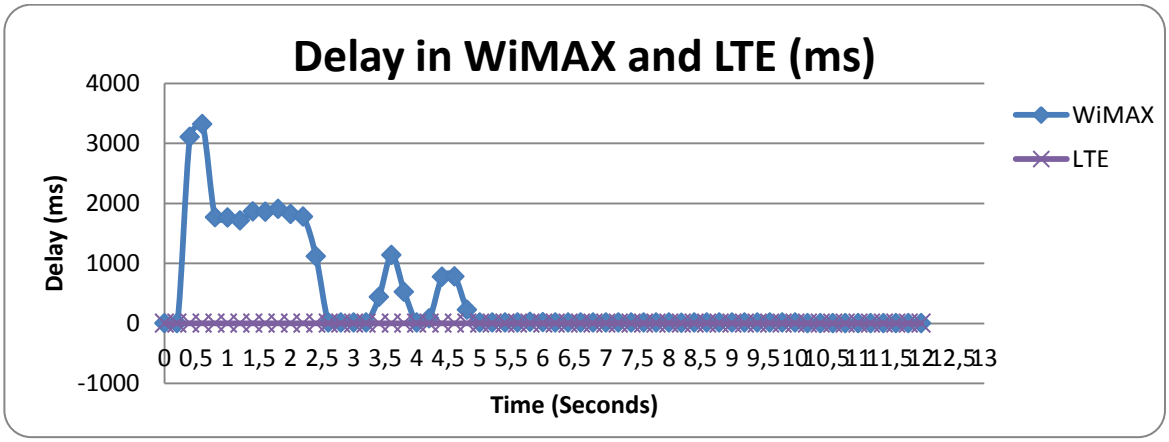

Gráfico 16 - Delay observado nos cenários WiMAX e LTE (Cenários 6 e 7)

Uma vez mais é possível constatar a melhor performance por parte do cenário LTE em relação ao WiMAX pois apresentou valores de atraso na ordem dos 1,8 ms, o que representa um atraso desprezível quando comparado com o apresentado pelo WiMAX que se mostrou instável até aos primeiros 5 segundos da simulação, chegando a atingir atrasos na ordem dos 3000 ms (3 segundos). Nos restantes segundos, os valores de atraso em WiMAX rondaram entre os 9 e 23 ms.

Outro tipo de conclusões que pode ser rapidamente retirado é o do comportamento das aplicações, quando usados diferentes tipos de modulação do sinal. Esta funcionalidade ainda somente é possível no actual módulo WiMAX pelo que o Gráfico 17 apenas aborda os vários tipos de modulação no cenário WiMAX.

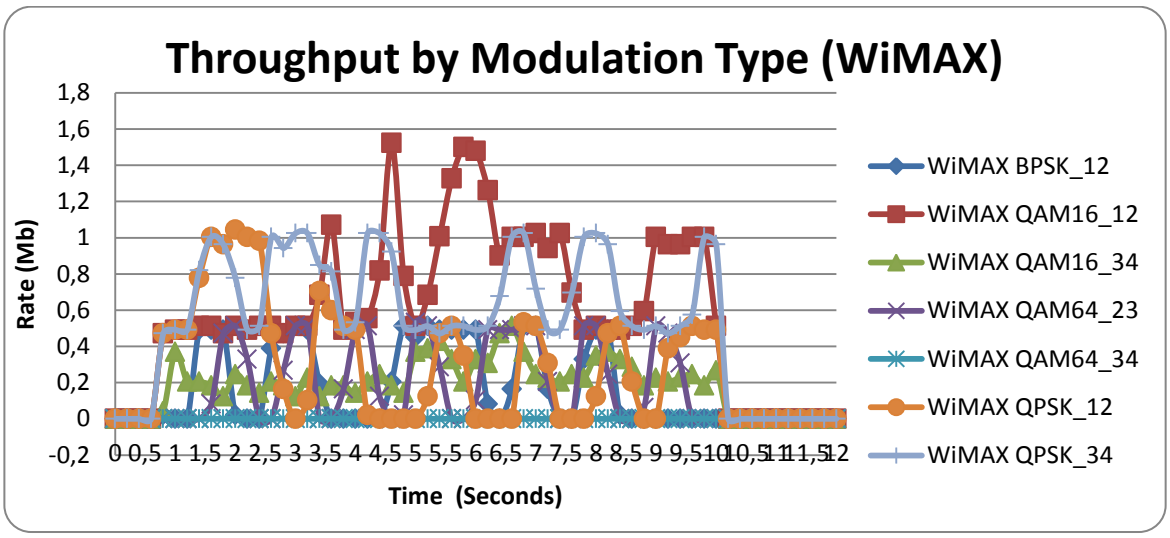

Gráfico 17 - Throughput por tipo de modulação do sinal (Cenário 6/WiMAX)

A partir do Gráfico 17 é possível concluir que a modulação QAM 16/12 (*Quadrature Amplitude Modulation)* constitui o tipo de modulação que melhores valores de *throughput* obteve, tanto que representa o tipo de modulação por omissão quando nenhum tipo de modulação é definido. Destaque também para o tipo de modulação QPSK-34 (*Quadrature Phase Shift Keying*) que apresentou a melhor performance logo após ao QAM 16/12.

7.4. CONCLUSÕES SOBRE OS CASOS DE ESTUDO

No presente capítulo foram apresentados alguns cenários de redes que atestam a fiabilidade e credibilidade do mapeamento desenvolvido no âmbito da integração do perfil NS-3 na linguagem NSDL, nomeadamente os cenários 1 a 4. Foi constatado que para aplicações UDP, os valores das várias grandezas medidas foram semelhantes em ambos os simuladores NS-2 e NS-3 porém para os fluxos TCP foram encontradas maiores discrepâncias entre os valores registados. Através do contacto com um dos fundadores do

NS-3 (Tom Henderson) ao qual foi exposta esta situação, concluiu-se que a divergência de valores alusivos aos pacotes gerados sob TCP será objecto de revisão para o actual módulo TCP sendo potencialmente modificado para futuras versões do NS-3.

Para os restantes cenários foi demonstrado como o mapeamento suporta tecnologias emergentes como é o caso do Wi-Fi, LTE e WiMAX bem como a obtenção de algumas estatísticas interessantes dado o actual estado dos módulos correspondentes a cada uma destas tecnologias.

No próximo capítulo são apresentadas as principais conclusões obtidas ao longo do desenvolvimento desta tese e que perspectivas futuras são consideradas para eventuais trabalhos de outros colegas que tenham interesse em contribuir no âmbito da *Framework* NSDL.

8. CONCLUSÕES E PERSPECTIVAS FUTURAS

Neste último capítulo são apresentadas as principais conclusões obtidas ao longo deste projecto de mestrado, contanto também com uma retrospectiva geral e perspectivas futuras para esta contribuição.

8.1. CONCLUSÕES

No desenvolvimento deste projecto, as principais conclusões assinaladas foram as seguintes:

O NS-3 constitui uma ferramenta de simulação de redes promissora, com foco especial para tecnologias emergentes como o Wi-Fi, WiMAX e LTE embora ainda apresente muitas limitações em cada uma destas tecnologias;

A *Framework* NSDL constitui uma plataforma de suporte à interligação de ferramentas de autoria de cenários de redes, por um lado as ferramentas que se focalizam na criação gráfica e/ou textual de cenários de redes e por outro lado as ferramentas de simulação de redes e obtenção de *outputs* representativos do comportamento dos cenários para futura análise, facilitando a tarefa de projecto e gestão de redes;

Com a introdução de um perfil NS-3 para NSDL, a *Framework* NSDL possui assim uma resposta face a esta plataforma para a rápida e eficaz modelação textual de cenários de redes para simulação no NS-3;

O processo de mapeamento NSDL para NS-3 permite a validação lexical/semântica de estruturas de dados NS-3 e a tradução eficiente das mesmas; Adicionalmente é disponibilizada documentação necessária ao desenvolvimento de cenários de redes NS-3 com a documentação gerada pela aplicação oXygen;

Com a criação dos cenários de redes apresentados nos casos de estudo (Capítulo 7) ficou visível a grande semelhança entre os módulos UDP de ambas as plataformas NS-2 e NS-3. Já os módulos de TCP apresentaram resultados ligeiramente distintos, nomeadamente quando não foi estipulado um valor fixo para o tamanho dos pacotes; Os resultados obtidos sobre *delay* e *jitter* foram muito semelhantes no intervalo de tempo em que as aplicações demonstraram estabilidade;

Com os resultados obtidos e a geração correcta dos *scripts* C++ para execução no ambiente NS-3, pode ser dada como válida a contribuição realizada com este projecto em prol da optimização de autoria de cenários de redes para a plataforma NS-3.

8.2. RETROSPECTIVA GERAL

Em termos gerais, os objectivos colocados inicialmente para este projecto foram aceitavelmente conseguidos na medida em que um estudo descritivo sobre o NS-3 foi desenvolvido para um melhor entendimento de que plataforma alvo estaria a ser considerada para integração na *Framework* NSDL.

O novo perfil NS-3 para NSDL inclui as principais tecnologias que definem o conjunto de tecnologias oferecido pela plataforma NS-3, embora haja a perfeita noção de que a inclusão de mais tecnologias actualmente projectadas só enriqueceria a contribuição.

A adaptação do processo de mapeamento NSDL para NS-2 para o mapeamento NSDL e NS-3 foi conseguido com sucesso mediante as tecnologias propostas e as estruturas de dados que detêm a especificação de cada elemento proposto.

8.3. PERSPECTIVAS FUTURAS

Em relação às perspectivas de continuação deste trabalho, para manter actualizado o perfil NS-3 para NSDL, o próximo passo seria a introdução de novas tecnologias que têm sido desenvolvidas para expandir as possibilidades de simulação por parte do NS-3, nomeadamente as seguintes [Nsnam, 2011]:

Bridge Device - Uma interface que emula o standard 802.1D;

EmuNet Device – Dispositivo genérico que pode funcionar com o papel de ponte (*Bridge*), P2P, etc.;

OpenFlow Switch Device – Um dispositivo que efectua o broadcast de fluxos entre vários segmentos de uma LAN;

Energy Models [Wu, 2011] - Modelos criados para simular dispositivos que funcionam com baterias, usado sobretudo por células sem fio (*wireless*);

UAN Models [Sacco, 2010] – *Underwater Acoustic Network* como próprio nome indica é uma solução submersa que visa a comunicação terrestre, aérea e aquática sobretudo para infra-estruturas costeiras.

Relativamente ao suporte à visualização, a evolução do perfil NS-3 passaria pela abordagem à visualização com NetAnim [NetAnim, 2010] e também a integração de outros modelos de mobilidade (*MobilityModel*) além do único que fora considerado para este projecto (*ConstantPositionMobilityModel*).

Além destas introduções de novos objectos, seria pertinente a integração de projectos que têm vindo a ser desenvolvidos para a melhoria de módulos actualmente existentes como é o caso do módulo LTE que tem o projecto LENA [LENA, 2011] com lançamento previsto para a versão ns-3.12.

Outra contribuição interessante que constitui como futuro trabalho é o da criação de uma ferramenta gráfica que suporte objectos contidos no *perfil* NS-3 e apresentar funcionalidades de importação e/ou exportação para *scripts* próprios do NS-3 para execução.

PUBLICAÇÕES DO AUTOR

- *Marques, E.M.D.; Sousa, J.J.F.; Sampaio; P.N.M. "NS-3 Simulation and Management of WiMAX and LTE Networks with NSDL"*. In *Proceedings of the ESM 2011 25th European Simulation and Modelling Conference, October 24-26, 2011, Guimarães, Portugal.*

- *Marques, E.M.D.; Sousa, J.J.F.; Sampaio; P.N.M.* "*Modeling and Simulation of DiffServ Scenarios with the NSDL Framework*". In *Proceedings of the CNSM 2011 - 7th International Conference on Network and Service Management, October 2011, Paris, France.*

REFERÊNCIAS

[Heidemann, 2003] Heidemann, J., "*NS Tutorial*", CSci551: Computer Networks, Friday Section, 2003

[Al-Shaer, 2011] Al-Shaer, E., "*TDC 562: Computer Network Design & Analysis (Internet Engineering) Lecture # 7-b: Network Simulation*", School of Computer Science & Telecommunications, DePaul University, Chicago, IL, 2011

[Hughes, 2009] Hughes, J., "*Network Simulation Introduction*", OpenXtra, 2009 - http://www.openxtra.co.uk/articles/network-simulation

[Marques, 2010] Marques, E.M.D.; Sampaio, P.N.M. "*A Framework for the Integration of Network Modelling and Simulation Tools*". In Proceedings of EUROSIS - The European Multidisciplinary Society for Modelling and Simulation Technology (ESM´ 2010). Hasselt, Belgium, October 25th-27th 2010

[OTcl, 2011] OTcl official website - http://otcl-tclcl.sourceforge.net/otcl/

[Cplusplus, 2011] C++ official website - http://www.cplusplus.com/

[Cisco, 2011] Cisco Academy, "*Cisco Packet Tracer*", 2011 - http://www.cisco.com/web/learning/netacad/course_catalog/PacketTracer.html

[VINT, 1995] NS-2 official website - http://isi.edu/nsnam/ns/

[Lucio, 2011] Lucio, G., Paredes-Farrera, M., Jammeh, E., Fleury, M., Reed, M., "*OPNET Modeler and Ns-2: Comparing the Accuracy Of Network Simulators for Packet-Level Analysis using a Network Testbed*", Electronic Systems Engineering Department, University of Essex, Colchester, Essex C04 3SQ, United Kingdom, 2011

[Weingartner, 2009] Weingartner, E., Lehn, H. e Wehrle, K., "*A performance comparison of recent network simulators*", Distributed Systems Group, RWTH Aachen University, Aachen, Germany, 2009

[OPNET, 2011] OPNET official website - http://www.opnet.com/

[SNT, 2001] Scalable Network Technologies, Inc., "*QualNet Simulator: Version 3.1 User's Manual*", 11022 Santa Monica Blvd., Suite 260, Los Angeles, California, 90025, USA, 2001

[Tetcos, 2011] Tetcos, "*NetSimTM Academic Ver 2.1*", No. 2089, 1267 Lake Side Avenue, Sunnyvale, California, 94085 USA, 2011

[Nsnam, 2011] NS-3 official website - http://www.nsnam.org/

[XML, 2008] W3C, "*Extensible Markup Language (XML)*", 2008 - http://www.w3.org/XML/

[XSD, 2004] W3C, "*XML Schema Definition (XSD)*", 2004 - http://www.w3.org/TR/xmlschema-0/

[DTD, 1999] W3C, "*Document Type Definition (DTD)*", 1999 - http://www.w3.org/TR/html4/sgml/dtd.html

[Chartier, 2006] Chartier, R., "*XSD vs DTD - So what's the difference?*", Rob Chartier ~ Contemplation…, .NET, C#, Work, etc., 2006 - http://weblogs.asp.net/rchartier/archive/2006/03/21/440782.aspx

[XSLT, 1999] W3C, "*eXtensible Stylesheet Language Transformation*", 1999 - http://www.w3.org/TR/xslt

[PHP, 2011] PHP official website, "*Hypertext Preprocessor*", 2011 - http://www.php.net/

[Lehr, 2003] Lehr, W., McKnight, L., "*Wireless Internet access: 3G vs. WiFi?*", Telecommunications Policy, MIT Research Program on Internet and Telecoms Convergence, Massachusetts Institute of Technology, 1 Amherst Street, E40-237, Cambridge, MA 02139, USA 4-181 Center for Science and Technology, Syracuse University, NY 13244, USA, 2003

[Prendergas, 2004] Prendergas, A., "*Why Wi-Fi?*", IT Director LV-CCLD, August 12, 2004

[Wi-Fi Alliance, 2011] Wi-Fi Alliance, official website - http://www.wi-fi.org/

[Amped, 2011] Amped Wireless, "*Wi-Fi Technology and Speeds*", Learning Center, 2011 - http://www.ampedwireless.com/learningcenter/

[Hiert, 2010] Hiert, G., Denteneer, D., Max, S., Taori, R., Cardona, J., Berlemann, L., Walke, B., "*IEEE 802.11S: THE WLAN MESH STANDARD*", IEEE Wireless Communications, February 2010

[Open80211s, 2011] Open80211s Project official website - http://www.o11s.org/

[Gabriel, 2011] Gabriel, C., "*WiMAX: The Critical Wireless Standard*",ARCchart ltd., 3 Finsbury Square, London, EC2A 1LN, UK, 2011

[Roh, 2009] Roh, W., Yanover, V., "*WiMAX Evolution: Emerging Technologies and Applications: 1. Introduction to WiMAX Technology*", John Wiley & Sons, Ltd., 2009

[Eberle, 2011] Eberle, D., "*LTE vs. WiMAX 4th generation telecommunication networks*", Computer Engineering B.Sc., Berlin Institute of Technology, Germany, 2011

[EngWeb, 2011] EngWeb, "*WiMAX Introduction*", 2011 - http://engweb.info/courses/wdt/lecture04/WIMAX_Technology_r.html

[Rohde, 2008] Rohde & Schwarz Products, "*UMTS Long Term Evolution (LTE) Technology Introduction*", September 2008

[Hamza, 2009] Hamza, A., "*Long Term Evolution (LTE) - A Tutorial*", Network Systems Laboratory, Simon Fraser University, Canada, 2009

[Bakharev, 2010] Bakharev, A., "*Simulator-Controlled Real-World Experiments in Multihop Wireless Mesh Networks*", Department of Computer Science, Electrical and Space Engineering, Luleå University of Technology, June 2010

[3GPP, 2011] 3GPP, The Mobile Broadband Standard, "*Motivation for 3GPP Release 8 - The LTE Release*", 2011 - http://www.3gpp.org/LTE

[Wikipedia, 2010] Wikipedia, "*E-UTRAN and Evolved Packet Core Architecture*", 2010 - http://en.wikipedia.org/wiki/File:EUTRAN_arch.op.svg

[Plácido, 2010] Plácido, R. "*Simulação de Redes com Multicasting e garantias de Qualidade de Serviço*", Dissertação de Mestrado em Engenharia em Telecomunicações e Redes. Universidade da Madeira, 2010

[Azevedo, 2010] Azevedo, J. "*Visual Network Descriptor*", Dissertação de Mestrado em Engenharia Informática. Universidade da Madeira, 2010

[Flex, 2011] Flex, Adobe official website - http://www.adobe.com/products/flex.html

[Araújo, 2011] Araújo, J. "*Virtualização Automática de Cenários de Rede*". Dissertação de Mestrado em Engenharia Informática. Universidade da Madeira, 2011

[Citrix, 2011] XenServer, Citrix official website - www.xensource.com

[Vyatta, 2011] Vyatta official website - http://www.vyatta.com/

[OpenSUSE, 2011] OpenSUSE v.11 official website - http://www.opensuse.org/en/

[Canoico, 2003] R. Canonico, D. Emma, G. Ventre, "*An XML Based Network Simulation Description Language*," presented at 7th International Symposium on Distributed Simulation and Real Time Applications (DS-RT 2003), Delft, Netherlands, 2003

[Piro, 2010] Piro G.; Member S.; Boggia G.; Boggia G.; and Camarda P., 2010. "*Simulating LTE Cellular Systems: an Open Source Framework"*. System, 60, no. 2, 1-16

[Ball, 2011] Ball C.; Hindelang T.; Kambourov I.; and Eder S., "*Spectral eciency assessment and radio performance comparison between LTE and WiMAX".* In PIMRC.IEEE, 1-6

[Carneiro, 2010] Carneiro, G., "*NS-3: Network Simulator 3*", UTM Lab Meeting, INESC Porto, Faculdade de Engenharia, Universidade do Porto, 20 de Abril de 2010

[Moraes, 2003] Moraes, I., "*AODV: Ad hoc On-Demand Distance Vector*", Departamento de Electrónica, Universidade Federal do Rio de Janeiro, Janeiro de 2003

[OLSR, 2008] "*OLSR (Optimized Link State Routing)*" official website - http://www.olsr.org/

[Suresh, 2011] Suresh, L., Merz, R., "*NS-3-Click: Click Modular Router Integration for NS-3*", Instituto Superior Técnico, Lisbon, Portugal, Deutsche Telekom Laboratories, Berlim, Germany, 2011

[Henderson, 2011] Henderson, T., "*Ns-3 project update",* May 2011 - http://www.nsnam.org/docs/ns-3-overview.pdf

[Ns3Install, 2011] Wikipedia, "*NS-3 Installation*", 2011 - http://www.nsnam.org/wiki/index.php/Installation

[RFC 1633] "*Integrated Services in the Internet Architecture: an Overview"* - http://www.ietf.org/rfc/rfc1633.txt

[RFC 2475] "*An Architecture for Differentiated Services"* - http://tools.ietf.org/html/rfc2475

[RFC 3031] "*Multiprotocol Label Switching Architecture*" - http://www.ietf.org/rfc/rfc3031.txt

[RFC 3561] "*Ad hoc On-Demand Distance Vector (AODV) Routing"* - http://www.ietf.org/rfc/rfc3561.txt

[Klein-Berndt, 2011] Klein-Berndt, L., "*A Quick Guide to AODV Routing*", Wireless Communications Technologies Group, National Institute of Standards and Technology, USA, 2011

[Narra, 2011] Narra, H., Cheng, Y., Çetinkaya, E., Rohrer, J., Sterbenz, J., "*Destination-Sequenced Distance Vector (DSDV) Routing Protocol Implementation in ns-3*", Department of Electrical Engineering & Computer Science, Information Technology & Telecommunications Research Center, University of Kansas, USA, 2011

[Perkins, 1994] Perkins, C., Bhagwat, P., "*Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers*", ACM SIGCOMM, 1994

[Lee, 2005] Lee, Y., Riley, G., "*Dynamic Nix-Vector Routing for Mobile Ad Hoc Networks*", Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC 2005), New Orleans, USA, Mar. 13 - 17 2005

[RFC 3626] "*Optimized Link State Routing Protocol (OLSR)*" - http://www.ietf.org/rfc/rfc3626.txt

[Lacage, 2006] Lacage, M., Henderson, T., "*Yet Another Network Simulator*", In WNS2 '06: Proceeding from the 2006 workshop on ns-2: the IP network simulator, 2006

[Andreev, 2010] Andreev, K., Boyko, P., "*IEEE 802.11s Mesh Networking NS-3 Model",* Institute for Information Transmission Problems, Moscow, Russia, 2010

[Farooq, 2009] Farooq, J., Turletti, T., "*An IEEE 802.16 WiMAX module for the NS-3 Simulator*," SIMUTools 2009 Conference, March 2009

[Sievanen, 2003] Sievanen, M., "*Application Protocol Data Unit*", T-110.497 Smart Cart Application Development, 2003 - http://www.tml.tkk.fi/Studies/T-110.497/2003/lecture4.pdf

[eTutorials, 2011] eTutorials.org, Networking, WiMAX Technology, "*Chapter 7: Convergence Sublayer (CS)",* 2011 - http://etutorials.org/Networking/

[NS3-LTE, 2011] NS-3 Project, ns-3 vns-3.11 documentation, "*Chapter Thirteen: LTE Module",* 2011 - http://www.nsnam.org/docs/release/3.11/models/html/lte.html

[RSC, 2001] Rational Software Corporation, "*Conceitos: Requisitos*", 2001 - http://www.wthreex.com/rup/process/workflow/requirem/co_req.htm

[Sommerville, 2004] Sommerville, I., "*Software Requirements*", Software Engineering, 7th Edition, Chapter 6, 2004

[Oxygen, 2011] oXygen XML Editor official website - http://www.oxygenxml.com/

[Notepad++, 2011] Notepad++ official website - http://notepad-plus-plus.org/

[Hyfinity, 2004] Hyfinity, "*MVC Overview*", 2004 - http://www.hyfinity.com/MVC_WebMaker_Overview.pdf

[Gilzar, 2002] Gilzar, N., "*Fast Track to Struts: What it Does and How*", TheServerSide.com, November 4, 2002

[Dreamweaver, 2011] Adobe Dreamweaver CS5.5, Adobe official website - http://www.adobe.com/products/dreamweaver.html

[Close, 1995] Close, D., Robbins, A., Rubin, P., Stallman, R., Oostru, P., "*The AWK Manual*", Free Software Foundation, Inc. 675 Mass Ave, Cambridge, MA 02139, USA, 1995

[Ed-Dbali, 2011] Ed-Dbali, A., "*UNIX: La programmation AWK*", Université d'Orléans, France, 2011

[Junqueira, 2007] Junqueira, T., "*Jitter formula used*", NCSA Archive, Illnois, USA, 2007

[Fortuna, 2010] Carneiro, G., Fortuna, P., Ricardo, M., "*FlowMonitor - a network monitoring framework for the Network Simulator 3 (NS-3)*", INESC Porto, Faculdade de Engenharia, Universidade do Porto, 2010

[LENA, 2011] LENA Project (LTE/EPC Network simulAtor) - http://www.nsnam.org/news/lte-development-first-public-release-of-the-lena-project/

[Wu, 2011] Wu, H., Nabar, S., Poovendran, R., "*An Energy Framework for the Network Simulator 3 (ns-3)*", Network Security Lab (NSL), Electrical Engineering Department, University of Washington, Seattle, USA, 2011

[Sacco, 2010] Sacco, A., "*UAN Framework project (GSoC)*", CodeReview, 2010 - http://codereview.appspot.com/1743057

[NetAnim, 2010] *NetAnim*, Wiki NSNam.org, Wiki, 2010 - http://www.nsnam.org/wiki/index.php/NetAnim

ANEXOS

ANEXO I – Funções PHP

Tabela 14 – Conjunto de funções PHP existentes no ficheiro "nsdl.functions.php" e descrição associada

Função	***Descrição***
`function nsdl_add_templates(& $objects, $templates, $id_parent = '')`	Esta função tem como principal objectivo verificar se os elementos que recebe (`$objects`) possuem um template na biblioteca de templates (`$templates`) e utilizar o template para preencher os dados do elemento, verificando para tal se os objectos possuem um *template* associado; Existem portanto **4** situações possíveis: 1. O elemento não possui nós filho no entanto possui um *template* na biblioteca: O *template* é usado para preencher a informação relativa ao elemento (recurso à função `nsdl_find_template` para encontrar o *template*); 2. O elemento possui nós filho mas não existe um *template* associado na biblioteca de *templates*: É feita uma chamada recursiva de forma a executar uma nova verificação aos nós filho; 3. Caso mais complexo, o elemento possui um *template* correspondente na biblioteca de *templates* e possui filhos: É feita a verificação recursiva da existência de *templates* aos nós filho. Quando as verificações terminarem, a informação do elemento é associada com as dos seus nós filho (com recurso à função `nsdl_merge_element`); 4. Caso mais simples, o elemento não possui nenhuma correspondência na biblioteca de

	templates nem algum nó filho logo não é possível utilizar um *template* para enriquecer a especificação do elemento.
`function nsdl_merge_element(& $dom1, $dom2, $id_parent='')`	Esta função é referenciada pela função `nsdl_add_templates`, em que esta envia um elemento (`$dom1`) e o conjunto dos seus nós filho (`$dom2`) e associa a informação do elemento à informação dos nós filho, eventualmente estes também com a sua própria especificação.
`function nsdl_uniform(& $nsdlObjects, & $nsdlFile)`	Esta função utiliza as capacidades das 3 funções seguintes (`nsdl_uniform_srcdst`, `nsdl_uniform_proto` e `nsdl_uniform_helpers`) para associar aplicações e protocolos pertencentes a uma mesma estrutura, de forma a facilitar a tradução.
`function nsdl_uniform_srcdst(& $nsdlObjects, & $nsdlFile)`	Com esta função, é feita a verificação num ficheiro nsdl (`$nsdlFile`) se os seus objectos (`$nsdlObjects`) correspondem a aplicações do tipo *ftp, telnet, cbr, exponential, pareto* e *trace*. Caso exista uma correspondência, é feita uma *query* para extrair a fonte (*src*) ou destino (*dst*) associado ao objecto. Função invocada pela função `nsdl_uniform`.
`function nsdl_uniform_proto(& $nsdlObjects, $nsdlFile)`	Esta função tem por finalidade a uniformização dos protocolos e suas respectivas aplicações. Inicialmente verifica se os nós dos objectos (`$nsdlObjects`) contidos no ficheiro (`$nsdlFile`) são de facto do tipo aplicação e caso assim se verifique, são criados temporariamente os protocolos para cliente e para servidor, dependendo da aplicação em questão. Posteriormente é verificado se a aplicação refere-se a uma fonte ou a um destino criando uma variável que descrimine o facto de ser cliente ou servidor. Com estas verificações realizadas, resta saber se a aplicação possui um protocolo definido em que surgem 2 situações: 1. Se não existe um protocolo associado, é

	construído um novo que denomina o novo protocolo, é-lhe atribuída uma nova ID que servirá para futura referência e estabelecidos os restantes atributos mediante a natureza do nó (i.e. servidor ou cliente); 2. Já caso exista uma id de protocolo, é verificado se existe uma versão desse protocolo: Se não existir, é adicionado um novo protocolo e consoante o tipo de nó (se servidor, se cliente) é atribuído um nome ao protocolo criado. Função invocada pela função `nsdl_uniform`.
`function nsdl_uniform_helpers(&$nsdlObjects, $nsdlFile)`	Esta função é usada para criar todas as ligações entre nós *source* e nós *destination*. Para todos os nós *destination* verificados que possuam uma *source* com uma determinada ID, é criado um *helper* para cada um de forma a facilitar o processo de tradução. Função invocada pela função `nsdl_uniform`.
`function nsdl_find_template ($templates, $tpl_value)`	Esta função é usada para encontrar um determinado *template* na biblioteca de *templates* para objectos NSDL. É assumido à partida que para certo objecto não existe *template*, posteriormente uma *query* é realizada a todos os *templates* disponíveis. Caso exista uma correspondência, é devolvido o *template* de forma a completar o objecto alvo com atributo(s) com valor(es) por omissão. Esta função é tipicamente invocada pela função `nsdl_add_templates`.
`function nsdl_validate ($xmldocument, $schema = "./schemas/nsdl_main_full.xsd", $show="true")`	Esta função tem por finalidade validar todo o documento NSDL, em termos das suas estruturas de dados e semântica entre a declaração dos objectos e seus atributos. Este processo é feito fazendo referência ao esquema XML de objectos NSDL (`"./schemas/nsdl_main_full.xsd"`), isto para o caso de a linguagem alvo ser OTcl. Para C++ (NS-3) o esquema a utilizar é o do ficheiro

	`nsdl_main_full.xsd`. Caso existam incoerências/erros de sintaxe, é invocada a função `libxml_display_error` que irá se encarregar de devolver os erros encontrados para facilitar o processo de depuração de erros por parte do utilizador.
`function nsdl_translate ($xmldocument, $stylesheet= "./transformations/ns2_nsdl_main.xsl")`	Esta função é responsável pela tradução XML -> OTcl/C++ fazendo referência ao ficheiro submetido (XML (NSDL)) e lendo os valores dos seus atributos de forma a preencher o(s) ficheiro(s) `.tcl/.cc`. Novamente o `$stylesheet` irá variar consoante a linguagem alvo: Na declaração à esquerda, a referência é feita ao esquema de transformação para NS2 (OTcl); Já para NS3 (C++) o esquema seria `"./transformations/ns3_nsdl_main.xsl"`. Esta tradução é feita assumindo que existe apenas 1 cenário.
`function nsdl_saveFile ($text, $filename, $lineStart = 2)`	Esta função é usada para salvar o resultado da tradução NSDL -> NS2/NS3, certificando-se antes que não existem declarações XML no ficheiro temporário criado com a sintaxe de OTcl/C++, eliminando-as (daí o uso da variável `$lineStart = 2` para assegurar que esta função lidará com o ficheiro alvo somente a partir da segunda linha.
`function parse_recursive($elem, $level = 0)`	Esta função tem por finalidade depurar a sintaxe do ficheiro final, criando a indentação necessária, realizando "*triming*" aos objectos e os seus elementos, efectuando uma análise semântica entre os vários elementos gerados pelas funções anteriores. Posteriormente é verificado se existem nós filho passíveis de uma reanálise semelhante à do seu nó pai. Este processo é então executado de forma recursiva até que não existam mais objectos filho a tratar (i.e. não existe `$level++`).
`function libxml_display_error($error)`	Esta função é usada para apresentar os erros (`$error`) de XML por tipo de erro: erros *fatais* (`LIBXML_ERR_FATAL`), erros *genéricos* (`LIBXML_ERR_ERROR`) ou somente *avisos* (`LIBXML_ERR_WARNING`), providenciando ao utilizador

	o código de erro param uma depuração mais eficaz e rápida. Esta função é posteriormente invocada pela função `libxml_display_errors` para providenciar a(s) *string(s)* a imprimir na página PHP após o processo de tradução.
`function libxml_display_errors()`	Esta função é encarregue de imprimir os erros na página PHP com a(s) *string(s)* geradas com a função anterior (i.e. `libxml_display_error`).
`function delxlines($txtfile, $numlines)`	Com esta função é possível eliminar linhas que são consideradas desnecessárias ao âmbito da tradução efectuada.

ANEXO II – Mapemanento entre NSDL e NS-3

Descrição do objecto	NSDL (XML)	NS3 (C++)
Principais objectos		
Criação do objecto "node"	<node id="***nodeID*** "></node>	Ptr<Node> ***nodeID*** = CreateObject<Node> ();
	<computer id=" ***computerID*** "></computer>	Ptr<Node> ***computerID*** = CreateObject<Node> ();
	<router id="***routerID***"> <staticroute> <src.address> **XXX.XXX.XXX.XXX** </src.address> <dst.address> **XXX.XXX.XXX.XXX** </dst.address> </staticroute> </router>	Ptr<Node> ***routerID*** = CreateObject<Node> (); Ipv4StaticRoutingHelper ***ipv4RoutingHelper***; Ptr<Ipv4StaticRouting> ***staticRouting*** = ***ipv4RoutingHelper***.GetStaticRoutin g (ipv4); *staticRouting*->AddHostRouteTo (Ipv4Address ("**XXX.XXX.XXX.XXX**"), Ipv4Address ("**XXX.XXX.XXX.XXX**"), <MetricValue>); MetricValue In case of multiple routes (optional)
Criação de um canal físico de	<link id="***linkID***"> <connection> <source>***SourceNode***</source>	PointToPointHelper ***linkID***Connection; NodeContainer ***linkID*** = NodeContainer (***SourceNode***,

transmissão	<destination>*DestinationNode*</destination>	*DestinationNode*);
	<connection/>	
	<type></type>	--
	<bandwidth>*DataRate*</bandwidth>	*linkID*Connection.SetDeviceAttribute ("*DataRate* ", StringValue ("<Value>Mbps"));
	<delay>*Delay*</delay>	Connection.SetChannelAttribute ("*Delay* ", StringValue ("<Value>ms"))
	<loss></loss>	--
	<queue-type></queue-type>	-- (?)
	<queue-limit></queue-limit>	--
	</link>	
Criação de um canal físico de transmissão que emula CSMA Ethernet	<link.csma id="*linkCsmaID*">	CsmaHelper *linkCsmaID*Connection;
	<connection>	NodeContainer *linkCsmaID* = NodeContainer (*SourceNode*, *DestinationNode*);
	<source>*SourceNode*</source>	
	<destination>*DestinationNode*</destination>	
	<connection/>	
	<type></type>	--
	<bandwidth>*DataRate*</bandwidth>	*linkCsmaID*Connection.SetChannelAttribute ("*DataRate* ", StringValue ("<Value>Mbps"));
	<delay>*Delay*</delay>	*linkCsmaID*Connection.SetChannelAttribute ("*Delay* ", StringValue ("<Value>ms"))
	<loss></loss>	--
	<queue-type></queue-type>	-- (?)
	<queue-limit></queue-limit>	--
	</link>	
Criação de um canal virtual de transmissão Yans (para uso em ambientes Wifi)	<link.yans id="*linkYansID*">	YansWifiChannelHelper *linkYansID*Connection;
	<delay.model>*DelayModel*</delay.model>	*linkYansID*Connection.SetPropagationDelay ("*DelayModel* ");
	<loss.model>*LossModel*</loss.model>	*linkYansID*Connection.AddPropagationLoss ("*LossModel* ");
	</link.yans>	--
Criação de um canal virtual de transmissão wimax(p	<link.wimax id="*linkWimaxID*">	Ptr<SimpleOfdmWimaxChannel> *linkWimaxID*; *linkWimaxID* = CreateObject<SimpleOfdmWimaxChannel> ();
	<loss.model>*LossModel*</loss.model>	*linkWimaxID* ->SetPropagationModel (*LossModel*);

ara uso em ambientes WiMAX)	</link.wimax>	--
Criação de um canal virtual de transmissão spectrum(para uso em ambientes LTE)	<link.spectrum id="***linkSpectrumID***">	Ptr< SingleModelSpectrumChannel> ***linkSpectrumID*** = CreateObject< SingleModelSpectrumChannel> ();
	<delay.model>***DelayModel***</delay.model>	***linkSpectrumID*** -> SetPropagationDelayModel (***DelayModel***);
	<loss.model>***LossModel***</loss.model>	***linkSpectrumID*** -> AddSpectrumPropagationLossModel (***LossModel***);
	</link.spectrum>	--
Criação de um domínio (internet)	<internet id="">	--
	<bandwidth></bandwidth>	--
	<delay></delay>	--
	<loss></loss>	--
	</internet>	--
Criação de um domínio	<domain id="">	--
	<bandwidth></bandwidth>	--
	<delay></delay>	--
	<loss></loss>	--
	</domain>	--
Criação de qos	<qos id="">	--
	<bandwidth></bandwidth>	--
	<delay></delay>	--
	<jitter></jitter>	--
	<loss></loss>	--
	</qos>	--
Criação de um router Core	<dscorenode id=””>	--
	<phbs>	--
	<entry>	--
	<phb.code></phb.code>	--
	<min></min>	--
	<max></max>	--
	<precedence></precedence>	--
	</entry>	--
	</phbs>	--
	<scheduler>	--
	<RR></RR>	--
	<PRI>	--
	<queue id=””>	--
	<rate></rate>	--
	</queue>	--
	</PRI>	--
	<WRR>	--
	<queue id=””>	--
	<weigth></weigth>	--
	</queue>	--

	</WRR>	--
	<WIRR>	--
	<queue id=””>	--
	<weigth></weigth>	--
	</queue>	--
	</WIRR>	--
	</scheduler>	--
	</dscorenode>	--
--Criação de um router Edge	<dsedgenode id="">	--
	<marker>	--
	<entry>	--
	<src.node> </src.node>	--
	<dst.node></dst.node>	--
	<phb.code></phb.code>	--
	</entry>	--
	</marker>	--
	<policer>	--
	<entry>	--
	<phb.code></phb.code>	--
	<new.phb.code></new.phb.code>	--
	<cir></cir>	--
	<TSW2CM></TSW2CM>	--
	<TSW3CM>	--
	<pir></pir>	--
	<sec.phb.code></sec.phb.code>	--
	</TSW3CM>	--
	<TokenBucket>	--
	<cbs></cbs>	--
	</TokenBucket>	--
	<srtcm>	--
	<cbs></cbs>	--
	<ebs></ebs>	--
	<sec.phb.code></sec.phb.code>	--
	</srtcm>	--
	<trtcm>	--
	<cbs></cbs>	--
	<pir></pir>	--
	<pbs></pbs>	--
	<sec.phb.code></sec.phb.code>	--
	</trtcm>	--
	</entry>	--
	</policer>	--
	</dsedgenode>	--
Identificação física e encaminhamento		
Criação de uma interface	<interface id=”***interfaceID***”>	Ipv4AddressHelper ipv4;
	<bandwidth></bandwidth>	--
	<delay></delay>	--
	<linkedto> ***NetDeviceContainer*** </linkedto>	Ipv4InterfaceContainer ***interfaceID*** = ipv4.Assign (***NetDeviceContainer***);

		With, *NetDeviceContainer* must be previously declared
	</interface>	
Criação de rotas RIP	<rip id=””>	--
	<version></version>	--
	<admdst></admdst>	--
	<updint></updint>	--
	</rip>	--
Criação de dispositivo wlan802.11 (WiFi)	< wlan802.11 id=”*wifiID*”>	WifiHelper *wifiID*;
	<standard>80211a</standard> (for an instance)	*wifiID*.SetStandard(WIFI_PHY_STANDARD_80211a**);**
	<wifi.physical>	YansWifiPhyHelper *wifiIDPhy* = YansWifiPhyHelper::Default ();
	<energy.detection>*EnergyDetectionThreshold*</energy.detection>	*wifiIDPhy*.Set ("*EnergyDetectionThreshold*", DoubleValue (<Value>));
	<ccamode>*CcaMode1Threshold*</ccamode>	*wifiIDPhy*.Set ("*CcaMode1Threshold* ", DoubleValue (<Value>));
	<txgain>*TxGain*</txgain>	*wifiIDPhy*.Set ("*TxGain*", DoubleValue (<Value>));
	<rxgain>*RxGain*</rxgain>	*wifiIDPhy*.Set ("*RxGain*", DoubleValue (<Value>));
	<txpower.levels>*TxPowerLevels*</txpower.levels>	*wifiIDPhy*.Set ("*TxPowerLevels* ", UintegerValue (<Value>));
	<txpower.end> *TxPowerEnd*</txpower.end>	*wifiIDPhy*.Set ("*TxPowerEnd* ", DoubleValue (<Value>));
	<txpower.start>*TxPowerStart*</txpower.start>	*wifiIDPhy*.Set ("*TxPowerStart* ", DoubleValue (<Value>));
	<rxnoise.figure>*RxNoiseFigure*</rxnoise.figure>	*wifiIDPhy*.Set ("*RxNoiseFigure* ", DoubleValue (<Value>));
	<state>*State<*/state>	*wifiIDPhy*.Set ("*State* ", StringValue (“<Value>”));
	<channel.switch.delay>*ChannelSwitchDelay*</channel.switch.delay>	*wifiIDPhy*.Set ("*ChannelSwitchDelay* ", TimeValue (MicroSeconds (<Value>)));
	<channel.number>*ChannelNumber*</channel.number>	*wifiIDPhy*.Set ("*ChannelNumber*", UintegerValue (<Value>));
	<yans.link.id>*YansChannelID*</yans.link.id>	*wifiIDPhy*.SetChannel(*YansChannelID*); with, *YansChannelID* must be previously declared
	</wifi.physical>	--
	<nonqoswifi>	NqosWifiMacHelper *wifiIDMac* = NqosWifiMacHelper::Default ();
	<type>adhoc</type> (for an instance)	*wifiIDMac*.SetType ("ns3::AdhocWifiMac",
	<cts.timeout> *CtsTimeout* </cts.timeout>	"*CtsTimeout*", TimeValue(MicroSeconds(<Value>)),
	<ack.timeout>*AckTimeout*	"*AckTimeout*", TimeValue(

	</ack.timeout>	MicroSeconds(<Value>)),
	<basic.block.ack.timeout> ***BasicBlockAckTimeout*** </basic.block.ack.timeout>	" ***BasicBlockAckTimeout*** ", TimeValue(MicroSeconds(<Value>)),
	<compressed.block.ack.timeout> ***CompressedBlockAckTimeout*** </compressed.block.ack.timeout>	"***CompressedBlockAckTimeout*** ", TimeValue(MicroSeconds(<Value>)),
	<sifs> ***Sifs*** </sifs>	" ***Sifs*** ", TimeValue(MicroSeconds(<Value>)),
	<eifsnodifs> ***EifsNoDifs*** </eifsnodifs>	" ***EifsNoDifs*** ", TimeValue(MicroSeconds(<Value>)),
	<slot> ***Slot*** </slot>	" ***Slot*** ", TimeValue(MicroSeconds(<Value>)),
	<pifs> ***Pifs*** </pifs>	" ***Pifs*** ", TimeValue(MicroSeconds(<Value>)),
	<maxdelay> ***MaxPropagationDelay*** </maxdelay>	" ***MaxPropagationDelay*** ", TimeValue(Seconds(<Value>)),
	<ssid> ***Ssid*** </ssid>	" ***Ssid***", SsidValue(Ssid("Value")),
	<beacon.interval> ***BeaconInterval*** </beacon.interval> *For "type" Ap*	" ***BeaconInterval***", TimeValue(MicroSeconds(<Value>)),
	<beacon.generation> ***BeaconGeneration*** </beacon.generation> *For "type" Ap*	" ***BeaconGeneration*** ", BooleanValue(<Value>),
	<probe.request.timeout> ***ProbeRequestTimeout*** </probe.request.timeout> *For "type" Sta*	" ***ProbeRequestTimeout*** ", TimeValue(Seconds(<Value>)),
	<assoc.request.timeout> ***AssocRequestTimeout*** </assoc.request.timeout> *For "type" Sta*	" ***AssocRequestTimeout*** ", TimeValue(Seconds(<Value>)),
	<max.missed.beacons> ***MaxMissedBeacons*** </max.missed.beacons> *For "type" Sta*	" ***MaxMissedBeacons*** ", UintegerValue (<Value>),
	<active.probing> ***ActiveProbing*** </active.probing> *For "type" Sta*	" ***ActiveProbing***", BooleanValue(<Value>),
	</nonqoswifi>	--
	<qoswifi>	QosWifiMacHelper ***wifiIDMac*** = QosWifiMacHelper::Default ();
	(same configuration as "*nonqoswifi*" only with ***QosEnabled*** attribute set to true)	--
	</qoswifi>	--
	<remote.station.manager>	***wifiID***.SetRemoteStationManager ("ns3::AarfWifiManager", (for an instance) With, ***wifiID (WifiHelper)*** must be previously declared
	<low.latency> ***IsLowLatency*** </low.latency>	" ***IsLowLatency*** ", BooleanValue(<Value>),

	<maxssrc> ***MaxSsrc*** </maxssrc>	" ***MaxSsrc*** ", UintegerValue (<Value>),
	<maxslrc> ***MaxSlrc*** </maxslrc>	" ***MaxSlrc*** ", UintegerValue (<Value>),
	<rtscts.threshold> ***RtsCtsThreshold*** </rtscts.threshold>	" ***RtsCtsThreshold*** ", UintegerValue (<Value>),
	<fragmentation.threshold> ***FragmentationThreshold*** </fragmentation.threshold>	" ***FragmentationThreshold*** ", UintegerValue (<Value>),
	<non.unicast.mode> ***NonUnicastMode*** </non.unicast.mode>	" ***NonUnicastMode*** ", WifiModeValue (<Value>),
	<!—Management Algorithms ->	--
	<aarf>	(Declared on *RemoteStationManager* instantiation – see above)
	<successk> ***SuccessK*** </successk>	" ***SuccessK***", DoubleValue (<Value>),
	<timerk> ***TimerK*** </timerk>	" ***TimerK*** ", DoubleValue (<Value>),
	<maxsuccess.threshold> ***MaxSuccessThreshold*** </maxsuccess.threshold>	" ***MaxSuccessThreshold*** ", UintegerValue (<Value>),
	<mintimer.threshold> ***MinTimerThreshold*** </mintimer.threshold>	" ***MinTimerThreshold*** ", UintegerValue (<Value>),
	<minsuccess.threshold> ***MinSuccessThreshold*** </minsuccess.threshold>	" ***MinSuccessThreshold*** ", UintegerValue (<Value>),
	</aarf>	--
	<aarfcd>	(Declared on *RemoteStationManager* instantiation – see above)
	<minrtswnd> ***MinRtsWnd*** </minrtswnd>	" ***MinRtsWnd***", UintegerValue (<Value>),
	<maxrtswnd> ***MaxRtsWnd*** </maxrtswnd>	" ***MaxRtsWnd*** ", UintegerValue (<Value>),
	<turn.off.rts.after.rate.decreases> ***TurnOffRtsAfterRateDecrease*** </turn.off.rts.after.rate.decreases>	" ***TurnOffRtsAfterRateDecrease*** ", BooleanValue(<Value>),
	<turn.on.rts.after.rate.increases> ***TurnOnRtsAfterRateIncrease*** </turn.on.rts.after.rate.increases>	" ***TurnOnRtsAfterRateIncrease*** ", BooleanValue(<Value>),
	<successk> ***SuccessK*** </successk>	" ***SuccessK***", DoubleValue (<Value>),
	<timerk> ***TimerK*** </timerk>	" ***TimerK*** ", DoubleValue (<Value>),

	<maxsuccess.threshold> ***MaxSuccessThreshold*** </maxsuccess.threshold>	" ***MaxSuccessThreshold*** ", UintegerValue (<Value>),
	<mintimer.threshold> ***MinTimerThreshold*** </mintimer.threshold>	" ***MinTimerThreshold*** ", UintegerValue (<Value>),
	<minsuccess.threshold> ***MinSuccessThreshold*** </minsuccess.threshold>	" ***MinSuccessThreshold*** ", UintegerValue (<Value>),
	</aarfcd>	--
	<amrr>	(Declared on *RemoteStationManager* instantiation – see above)
	<update.period> ***UpdatePeriod*** </update.period>	" ***UpdatePeriod***", TimeValue(Seconds(<Value>)),
	<failure.ratio> ***FailureRatio*** </failure.ratio>	" ***FailureRatio*** ", DoubleValue (<Value>),
	<success.ratio> ***SuccessRatio*** </success.ratio>	" ***SuccessRatio*** ", DoubleValue (<Value>),
	<maxsuccess.threshold> ***MaxSuccessThreshold*** </maxsuccess.threshold>	" ***MaxSuccessThreshold*** ", UintegerValue (<Value>),
	<minsuccess.threshold> ***MinSuccessThreshold*** </minsuccess.threshold>	" ***MinSuccessThreshold*** ", UintegerValue (<Value>),
	</amrr>	--
	<arf>	(Declared on *RemoteStationManager* instantiation – see above)
	<timer.threshold> ***TimerThreshold*** </timer.threshold>	" ***TimerThreshold***", UintegerValue (<Value>),
	<success.threshold> ***SuccessThreshold*** </success.threshold>	" ***SuccessThreshold*** ", UintegerValue (<Value>),
	</arf>	--
	<cara>	(Declared on *RemoteStationManager* instantiation – see above)
	<probe.threshold> ***ProbeThreshold*** </probe.threshold>	" ***ProbeThreshold***", UintegerValue (<Value>),
	<failure.threshold> ***FailureThreshold*** </failure.threshold>	" ***FailureThreshold*** ", UintegerValue (<Value>),
	<success.threshold></success.threshold>	" ***SuccessThreshold*** ", UintegerValue (<Value>),
	<timeout> ***Timeout*** </timeout>	" ***Timeout*** ", UintegerValue (<Value>),
	</cara>	--
	<constantrate>	(Declared on *RemoteStationManager*

		instantiation – see above)
	<datamode> ***DataMode*** </datamode>	" ***DataMode***", StringValue (“<Value>”),
	<controlmode> ***ControlMode*** </controlmode>	" ***ControlMode***", StringValue (“<Value>”),
	</constantrate>	--
	<ideal>	(Declared on *RemoteStationManager* instantiation – see above)
	<ber.threshold> ***BerThreshold*** </ber.threshold>	" ***BerThreshold***", DoubleValue (<Value>),
	</ideal>	--
	<minstrel>	(Declared on *RemoteStationManager* instantiation – see above)
	<update.statistics> ***UpdateStatistics*** </update.statistics>	" ***UpdateStatistics***", TimeValue(Seconds(<Value>)),
	<lookaroundrate> ***LookAroundRate*** </lookaroundrate>	" ***LookAroundRate*** ", DoubleValue (<Value>),
	<ewma> ***EWMA*** </ewma>	" ***EWMA*** ", DoubleValue (<Value>),
	<segment.size> ***SegmentSize*** </segment.size>	" ***SegmentSize*** ", DoubleValue (<Value>),
	<sample.column> ***SampleColumn*** </sample.column>	" ***SampleColumn*** ", DoubleValue (<Value>),
	<packet.length> ***PacketLength*** </packet.length>	" ***PacketLength*** ", DoubleValue (<Value>),
	</minstrel>	--
	<onoe>	(Declared on *RemoteStationManager* instantiation – see above)
	<update.period> ***UpdatePeriod*** </update.period>	" ***UpdatePeriod***", TimeValue(Seconds(<Value>)),
	<raise.threshold> ***RaiseThreshold*** </raise.threshold>	" ***RaiseThreshold*** ", UintegerValue (<Value>),
	<addcredit.threshold> ***AddCreditThreshold*** </addcredit.threshold>	" ***AddCreditThreshold*** ", UintegerValue (<Value>),
	</onoe>	--
	<rraa>	(Declared on *RemoteStationManager* instantiation – see above)
	<basic> ***Basic*** </basic>	" ***Basic***", BooleanValue(<Value>),
	<timeout> ***Timeout*** </timeout>	" ***Timeout*** ", TimeValue(Seconds(<Value>)),
	<ewnd.for.54mbps>	" ***ewndFor54mbps*** ", UintegerValue

	ewndFor54mbps </ewnd.for.54mbps>	(<Value>),
	<ewnd.for.48mbps> ***ewndFor54mbps*** </ewnd.for.48mbps>	" ***ewndFor48mbps*** ", UintegerValue (<Value>),
	<ewnd.for.36mbps> ***ewndFor36mbps*** </ewnd.for.36mbps>	" ***ewndFor36mbps*** ", UintegerValue (<Value>),
	<ewnd.for.24mbps> ***ewndFor24mbps*** </ewnd.for.24mbps>	" ***ewndFor24mbps*** ", UintegerValue (<Value>),
	<ewnd.for.18mbps> ***ewndFor18mbps*** </ewnd.for.18mbps>	" ***ewndFor18mbps*** ", UintegerValue (<Value>),
	<ewnd.for.12mbps> ***ewndFor12mbps*** </ewnd.for.12mbps>	" ***ewndFor12mbps*** ", UintegerValue (<Value>),
	<ewnd.for.9mbps> ***ewndFor9mbps*** </ewnd.for.9mbps>	" ***ewndFor9mbps*** ", UintegerValue (<Value>),
	<ewnd.for.6mbps> ***ewndFor6mbps*** </ewnd.for.6mbps>	" ***ewndFor6mbps*** ", UintegerValue (<Value>),
	<pori.for.48mbps> ***poriFor48mbps*** </pori.for.48mbps>	" ***poriFor48mbps***", DoubleValue (<Value>),
	<pori.for.36mbps> ***poriFor36mbps*** </pori.for.36mbps>	" ***poriFor36mbps*** ", DoubleValue (<Value>),
	<pori.for.24mbps> ***poriFor24mbps*** </pori.for.24mbps>	" ***poriFor24mbps*** ", DoubleValue (<Value>),
	<pori.for.18mbps> ***poriFor18mbps*** </pori.for.18mbps>	" ***poriFor18mbps*** ", DoubleValue (<Value>),
	<pori.for.12mbps> ***poriFor12mbps*** </pori.for.12mbps>	" ***poriFor12mbps*** ", DoubleValue (<Value>),
	<pori.for.9mbps> ***poriFor9mbps*** </pori.for.9mbps>	" ***poriFor9mbps*** ", DoubleValue (<Value>),
	<pori.for.6mbps> ***poriFor6mbps*** </pori.for.6mbps>	" ***poriFor6mbps*** ", DoubleValue (<Value>),
	<pmtl.for.54mbps> ***pmtlFor54mbps*** </pmtl.for.54mbps>	" ***pmtlFor54mbps*** ", DoubleValue (<Value>),
	<pmtl.for.48mbps> ***pmtlFor48mbps*** </pmtl.for.48mbps>	" ***pmtlFor48mbps*** ", DoubleValue (<Value>),
	<pmtl.for.36mbps> ***pmtlFor36mbps*** </pmtl.for.36mbps>	" ***pmtlFor36mbps*** ", DoubleValue (<Value>),
	<pmtl.for.24mbps> ***pmtlFor24mbps***	" ***pmtlFor24mbps*** ", DoubleValue (<Value>),

	</pmtl.for.24mbps>	
	<pmtl.for.18mbps> ***pmtlFor18mbps*** </pmtl.for.18mbps>	" ***pmtlFor18mbps*** ", DoubleValue (<Value>),
	<pmtl.for.12mbps> ***pmtlFor12mbps*** </pmtl.for.12mbps>	" ***pmtlFor12mbps*** ", DoubleValue (<Value>),
	<pmtl.for.9mbps> ***pmtlFor9mbps*** </pmtl.for.9mbps>	" ***pmtlFor9mbps*** ", DoubleValue (<Value>),
	</rraa>	--
	</remote.station.manager>	);
	<linkedto>***wifiID2***</linkedto>	NetDeviceContainer ***devices*** = ***wifiID***.Install (***wifiphyID***, ***nonqoswifiID*** | ***QoswifiID***, ***node***); With, ***Node*** is the holder of "***wifiID2***" interface instance
	</ wlan802.11>	--
Criação de dispositivo lan802.3	< lan802.3 id="">	--
	<bandwidth></bandwidth>	--
	<delay></delay>	--
	<linkedto></linkedto>	--
	</ lan802.3>	--
Criação de dispositivo wlan802.16 (WiMAX)	<wlan802.16 id="***WimaxID***"> (for an instance)	WimaxHelper::SchedulerType ***WimaxID*** scheduler = WimaxHelper::SCHED_TYPE_***SIMPLE***; NetDeviceContainer d***<node>***d***<linkednode>*** = wimax.Install (NodeContainer (***<node>,<linkednode>***) , WimaxHelper::DEVICE_TYPE_*SUBSCRIBER_STATION* , WimaxHelper::SIMPLE_PHY_TYPE_OFDM , ***WimaxLink,*** ***WimaxID*** scheduler); Ptr<*SubscriberStationNetDevice*> ***<node>***SS; *For "type" SubscriberStation* ***<node>***SS = d***<node>***d***<linkednode>***.Get(0) ->GetObject<*SubscriberStationNetDevice*> (); *For "type" SubscriberStation* ***OR*** Ptr<*BaseStationNetDevice*> ***<node>***BS; *For "type"*
	<type>*BaseStation or SubscriberStation* </type>	
	<wimax.link.id>***WimaxLink***</wimax.link.id>	
	<scheduler.type>***simple***</scheduler.type>	

		BaseStation ***<node>***BS = d***<node>***d***<linkednode>***.Get(0) ->GetObject<*BaseStationNetDevice*> (); *For "type" BaseStation* With, Wimax (***WimaxHelper***) must be previously declared ***node*** is the holder of wlan802.16 object; ***linkednode*** is the linked holder of wlan802.16 object (see below on <linkedto> element).
	<modulation.type>***qam16_12***</modulation.type> *For "type" SubscriberStation*	***<node>***SS ->SetModulationType (WimaxPhy::MODULATION_TYPE_***QAM16_12***);
	<service.flow>	ServiceFlow ***Down | Up*** ServiceFlow***WimaxID DestinationLowerPort*** = wimax.CreateServiceFlow (ServiceFlow::SF_DIRECTION_***DOWN***, ServiceFlow::SF_TYPE_***BE***, Down***WimaxID DestinationLowerPort***); ***<node>***SS ->AddServiceFlow (***Down | Up*** ServiceFlow***WimaxID DestinationLowerPort***); With, Wimax (***WimaxHelper***) must be previously declared Down***WimaxID DestinationLowerPort*** is an instance of following <ipcs.classifier>
	<direction>***up | down***</direction>	
	<scheduler.type>***be***</scheduler.type>	
	<ipcs.classifier>	IpcsClassifierRecord Down***WimaxID DestinationLowerPort*** (Ipv4Address ("***SourceIPv4***"), Ipv4Mask ("***SourceMask***"), i***WimaxID***.GetAddress(0), Ipv4Mask("***DestinationMask***"), ***SourceLowerPort***, ***SourceUpperPort***, ***DestinationLowerPort***, ***DestinationUpperPort***, ***tcp | udp | 6 | 17***,
	<src.address> ***SourceIPv4*** </src.address>	
	<src.mask> ***SourceMask*** </src.mask>	
	<dst.address> ***DestinationIPv4*** </dst.address>	
	<dst.mask> ***DestinationMask*** </dst.mask>	
	<src.port.lower>	

	SourceLowerPort </src.port.lower> <src.port.upper> ***SourceUpperPort*** </src.port.upper> <dst.port.lower> ***DestinationLowerPort*** </dst.port.lower> <dst.port.upper> ***DestinationUpperPort*** </dst.port.upper> <protocol.id> ***tcp | udp | 6 | 17*** </protocol.id> <priority> ***PriorityValue*** </priority> </ipcs.classifier> </service.flow>	***PriorityValue***);
	<linkedto> ***OtherWlan802.16ID***</linkedto>	Defined on constructor (see above) With ***OtherWlan802.16ID*** as the instance of WiMAX included in ***linkednode*** specification
	</wlan802.16>	--
Criação de dispositivo wlan802.11s (Mesh)	<wlan802.11s id="***MeshID***">	MeshHelper ***MeshID*** = MeshHelper::Default ();
	<standard>80211a</standard> (for an instance)	***MeshID***.SetStandard(WIFI_PHY_STANDARD_80211a**);**
	<stack.type>flame</stack.type> (for an instance)	***MeshID***.SetStackInstaller ("ns3::FlameStack");
	<mesh.physical>	YansWifiPhyHelper ***meshPhy*** = YansWifiPhyHelper::Default ();
	<energy.detection>***EnergyDetectionThreshold***</energy.detection>	***meshPhy***.Set ("***EnergyDetectionThreshold***", DoubleValue (<Value>));
	<ccamode>***CcaMode1Threshold***</ccamode>	***meshPhy***.Set ("***CcaMode1Threshold*** ", DoubleValue (<Value>));
	<txgain>***TxGain***</txgain>	***meshPhy***.Set ("***TxGain***", DoubleValue (<Value>));
	<rxgain>***RxGain***</rxgain>	***meshPhy***.Set ("***RxGain***", DoubleValue (<Value>));
	<txpower.levels>***TxPowerLevels***</txpower.levels>	***meshPhy***.Set ("***TxPowerLevels*** ", UintegerValue (<Value>));
	<txpower.end> ***TxPowerEnd***</txpower.end>	***meshPhy***.Set ("***TxPowerEnd*** ", DoubleValue (<Value>));
	<txpower.start>***TxPowerStart***</txpower.start>	***meshPhy***.Set ("***TxPowerStart*** ", DoubleValue (<Value>));
	<rxnoise.figure>***RxNoiseFigure***</rxnoise.figure>	***meshPhy***.Set ("***RxNoiseFigure*** ", DoubleValue (<Value>));
	<state>***State***</state>	***meshPhy***.Set ("***State*** ", StringValue (“<Value>”));
	<channel.switch.delay>***ChannelSwitchDelay***</channel.switch.delay>	***meshPhy***.Set ("***ChannelSwitchDelay*** ", TimeValue (MicroSeconds (<Value>)));

	<channel.number>*ChannelNumber*</channel.number>	*meshPhy*.Set ("*ChannelNumber*", UintegerValue (<Value>));
	<yans.link.id>*YansChannelID*</yans.link.id>	*meshPhy*.SetChannel(*YansChannelID*); with, *YansChannelID* must be previously declared
	</mesh.physical>	--
	<spread>True</spread> (for an instance)	*MeshID*.SetSpreadInterfaceChannels (MeshHelper::SPREAD_CHANNELS);
	<start.time>TimeValue</start.time>	*MeshID*.SetMacType ("*RandomStart*", TimeValue (Seconds(<TimeValue >)));
	<remote.station.manager></remote.station.manager>	*MeshID*.SetRemoteStationManager ("ns3::AarfWifiManager", (for an instance) With, *MeshID (MeshHelper)* must be previously declared (Same specification that elements wlan802.11 holds – see above)
	<linkedto>*anotherMeshID*</linkedto>	NetDeviceContainer *devices* = *MeshID*.Install (*meshPhy*, NodeContainer (node (holding *MeshID*), node (holding *anotherMeshID*));
	</wlan802.11s>	--
Criação de dispositivo LTE	<lte id="*lteID*">	LteHelper *lteID*;
	<type>ue</type> (for an instance)	Ptr<UeNetDevice> = *nodeID*UE; With *nodeID* as the holder of element *lteID*
	<lte.physical>	Ptr<UeLtePhy> *lteID*PhyUE = CreateObject<UeLtePhy> ();
	<spectrum.channel>	Ptr<UeLteSpectrumPhy> *lteID*dlUE = CreateObject<UeLteSpectrumPhy> (); Ptr<UeLteSpectrumPhy> *lteID*ulUE = CreateObject<UeLteSpectrumPhy> (); *lteID*PhyUE->SetDownlinkSpectrumPhy (*lteID*dlUE); *lteID*PhyUE ->SetUplinkSpectrumPhy (*lteID*ulUE);
	<dl.spectrum.link.id>spectrumLinkID</dl.spectrum.	*lteID*dlUE ->SetChannel (spectrumLinkID);

	link.id>	**spectrumLinkID** ->AddRx (***lteID***dlUE); With **spectrumLinkID** as an instance of SpectrumChannel
	<ul.spectrum.link.id>**spectrumLinkID2**</ul.spectrum.link.id>	***lteID***ulUE ->SetChannel (**spectrumLinkID2**); With **spectrumLinkID2** as an instance of SpectrumChannel
	</spectrum.channel>	--
	</lte.physical>	--
	<enb.id>***eNBID***</enb.id> *for UE nodes only*	***lteID***.AddDownlinkChannelRealization (***ueID***Mobility, ***eNBID***Mobility, ***nodeID***UE->GetPhy ()); With ***ueID***Mobility and ***eNBID***Mobility as instances of mobility model and ***nodeID*** as the holder of element ***lteID***
	<radio.bearer>	Ptr<RadioBearerInstance> bearer***lteID******DestinationLowerPort*** = CreateObject<RadioBearerInstance> (); **Attention:** Here is clear the importance to define different ports for different bearers to avoid possible re-declarations
	<direction>dl</direction> (for an instance)	bearer***lteID******DestinationLowerPort*** ->SetBearerDirection (RadioBearerInstance::DIRECTION_TYPE_DL);
	<bearer.type>drb</bearer.type> (for an instance)	bearer***lteID******DestinationLowerPort*** ->SetBearerType (RadioBearerInstance::BEARER_TYPE_DRB);

	<ipcs.classifier> <src.address> ***SourceIPv4*** </src.address> <src.mask> ***SourceMask*** </src.mask> <dst.address> ***DestinationIPv4*** </dst.address> <dst.mask> ***DestinationMask*** </dst.mask> <src.port.lower> ***SourceLowerPort*** </src.port.lower> <src.port.upper> ***SourceUpperPort*** </src.port.upper> <dst.port.lower> ***DestinationLowerPort*** </dst.port.lower> <dst.port.upper> ***DestinationUpperPort*** </dst.port.upper> <protocol.id> ***tcp | udp | 6 | 17*** </protocol.id> <priority> ***PriorityValue*** </priority> </ipcs.classifier>	IpcsClassifierRecord Down***lteIDDestinationLowerPort*** (Ipv4Address ("***SourceIPv4*** "), Ipv4Mask ("***SourceMask*** "), i***lteID***.GetAddress(0), Ipv4Mask("***DestinationMask*** "), ***SourceLowerPort***, ***SourceUpperPort***, ***DestinationLowerPort***, ***DestinationUpperPort***, ***tcp | udp | 6 | 17***, ***PriorityValue***);
	<bearer.qos><!-- QoS Support -->	Ptr<BearerQosParameters> ***lteIDDirection***QoS = CreateObject<BearerQosParameters> (
	<type>ngbr</type> (for an instance)	/*independent of constructor*/ ***lteIDDirection***QoS ->SetBearerQosType (BearerQosParameters::BEARER_TYPE_NGBR);
	<qci>***QCIValue***</qci>	***QCIValue*** , (defined on constructor)
	<apec>***APEC***</apec>	***APEC***, (defined on constructor)
	<apev> ***APECV*** </apev>	***APECV***, (defined on constructor)
	<gbr>***GBRValue***</gbr>	***GBRValue***, (defined on constructor)
	<mbr>***MBRValue*** </mbr>	***MBRValue***, (defined on constructor)
	<max.delay> ***MaxDelay***</max.delay>	/*independent of constructor*/ ***lteIDDirection***QoS ->SetMaxDelay (***MaxDelay***);
	</bearer.qos>	); (end of constructor)
	</radio.bearer>	bearer***lteIDDestinationLowerPort*** ->SetIpcsClassifierRecord (Down***lteIDDestinationLowerPort***);
	<linkedto>***anotherlteID*** </linkedto>	NetDeviceContainer dnodednode =

		lteID.Install (, NodeContainer (node (holding ***lteID***), node (holding ***anotherlteID***));
	</lte>	--
Protocolos		
Instanciação de protocolo TCP	<tcp id="***tcpID***">	OnOffHelper onoff***tcpID*** ("ns3::TcpSocketFactory",
	<interface.id> ***interfaceID*** </interface.id>	InetSocketAddress (***interfaceID***.GetAddress (***SourceNode***), <TCPport>));
	<isolayer></isolayer>	--
	<tcpiplayer></tcpiplayer>	--
	<version></version>	--
	</tcp>	--
Instanciação de protocolo UDP	<udp id="***udpID***">	OnOffHelper onoff***udpID*** ("ns3::UdpSocketFactory",
	<interface.id> ***interfaceID*** </interface.id>	InetSocketAddress (***interfaceID***.GetAddress (***SourceNode***), <UDPport>));
	<isolayer></isolayer>	--
	<tcpiplayer></tcpiplayer>	--
	<version></version>	--
	</udp>	--
Instanciação de protocolo RTP	<rtp id="">	--
	<interface.id> ***interfaceID*** </interface.id>	--
	<isolayer></isolayer>	--
	<tcpiplayer></tcpiplayer>	--
	<version></version>	--
	</rtp>	--
Criação de endereçamento IPv4	<ipv4 id="***ipv4ID***">	Ipv4AddressHelper ***ipv4ID***;
	<net.address> **XXX.XXX.XXX.0**</net.address>	***ipv4ID***.SetBase ("**XXX.XXX.XXX.0**", "255.255.255.0");
	<mask>255.255.255.0</mask>	(correspondence above)
	<address></address>	--
	<isolayer></isolayer>	--
	<bandwidth></bandwidth>	--
	<delay></delay>	--
		Ipv4InterfaceContainer ***iinterfaceIDiinterfaceID2*** = ***ipv4ID***.Assign (***NetDeviceContainer***); ***NetDeviceContainer*** must be previously declared ***interfaceID2*** is the interface linked to ***interfaceID***
	<interface.id>***interfaceID***</interface.id>	
	<tcpiplayer></tcpiplayer>	--
	</ipv4>	--
Criação de	<ipv6 id="***ipv6ID***">	Ipv6AddressHelper ***ipv6ID***;
	<net.address> **2001::1**</net.address>	***ipv6ID***.SetBase ("**2001::1**", 64);

endereçamento IPv6	<mask>255.255.255.0</mask>	--
	<address></address>	--
	<isolayer></isolayer>	--
	<bandwidth></bandwidth>	--
	<delay></delay>	--
		Ipv6InterfaceContainer ***iinterfaceIDiinterfaceID2*** = ***ipv6ID***.Assign (***NetDeviceContainer***); ***NetDeviceContainer*** must be previously declared ***interfaceID2*** is the interface linked to ***interfaceID***
	<interface.id>***interfaceID***</interface.id>	
	<tcpiplayer></tcpiplayer>	--
	</ipv6>	--
Aplicações		
Criação de uma aplicação genérica	<application id="***appID***">	ApplicationContainer apps***appID*** = onoff<***tcpID*** | ***udpID*** >.Install (***SourceNode***);
	<role></role>	
	<protocol.id> ***tcpID*** | ***udpID*** </protocol.id>	
	<src.app> ***SourceNode*** </src.app>	
	<dst.app> ***DestinationNode*** </dst.app>	PacketSinkHelper ***sink*** ("ns3::**<Tcp | Udp>**SocketFactory", InetSocketAddress (Ipv4Address::GetAny (), **<TCP| UDP>**port)); ***appID*** = ***sink***.Install (***DestinationNode***);
	<rate> ***DataRate*** </rate>	onoff<***tcpID | udpID***>.SetAttribute ("***DataRate*** ", StringValue ("<Value>kb/s"));
	<packetsize> ***PacketSize*** </packetsize>	onoff<***tcpID | udpID***>.SetAttribute ("***PacketSize*** ", UintegerValue (<Value>));
	</application>	--
Criação de uma aplicação FTP	<ftp id="***ftpID***">	(instance OnOffHelper generic application above)
	<role></role>	
	<protocol.id> ***tcpID*** </protocol.id>	
	<src.app> ***SourceNode*** </src.app>	
	<dst.app> ***DestinationNode*** </dst.app>	--
	<rate> ***DataRate*** </rate>	onoff***ftpID***.SetAttribute ("***DataRate*** ", StringValue ("<Value>kb/s"));
	<packetsize> ***PacketSize*** </packetsize>	onoff***ftpID***.SetAttribute ("***PacketSize*** ", UintegerValue (<Value>));
	<maxpackets>***MaxPackets***</maxpackets>	onoff***ftpID***.SetAttribute ("***MaxBytes***", UintegerValue (<Value>)); (***MaxPackets can be translated to MaxBytes***)
	</ftp>	--

Criação de uma aplicação TELNET	<telnet id="***telnetID***">	(instance OnOffHelper generic application)
	<role></role>	--
	<protocol.id> ***tcpID*** </protocol.id>	--
	<src.app> ***SourceNode*** </src.app>	--
	<dst.app> ***DestinationNode*** </dst.app>	--
	<rate> ***DataRate*** </rate>	onoff***telnetID***.SetAttribute ("***DataRate*** ", StringValue ("<Value>kb/s"));
	<packetsize> ***PacketSize*** </packetsize>	onoff***telnetID***.SetAttribute ("***PacketSize*** ", UintegerValue (<Value>));
	<interval></interval>	--
	</telnet>	--
Criação de uma aplicação CBR	<cbr id="***cbrID***">	(instance OnOffHelper generic application) onoff***cbrID***.SetAttribute ("OnTime", RandomVariableValue (ConstantVariable (1))); onoff***cbrID***.SetAttribute ("OffTime", RandomVariableValue (ConstantVariable (0)));
	<role></role>	--
	<protocol.id> ***udpID*** </protocol.id>	--
	<src.app> ***SourceNode*** </src.app>	--
	<dst.app> ***DestinationNode*** </dst.app>	--
	<rate> ***DataRate*** </rate>	onoff***cbrID***.SetAttribute ("***DataRate*** ", StringValue ("<Value>kb/s"));
	<packetsize> ***PacketSize*** </packetsize>	onoff***cbrID***.SetAttribute ("***PacketSize*** ", UintegerValue (<Value>));
	<maxpackets></maxpackets>	onoffapp.SetAttribute ("MaxBytes", UintegerValue (<Value>));
	<random></random>	--
	</cbr>	--
Criação de uma aplicação Exponencial	<exponential id="***expID***">	(instance OnOffHelper generic application) onoff***expID***.SetAttribute ("OnTime", RandomVariableValue (ExponentialVariable (double ***m***, double ***b***))); onoff***expID***.SetAttribute ("OffTime", RandomVariableValue (ExponentialVariable ())); Where ***m*** is a specific mean and ***b*** is an upper limit (both optional)
	<role></role>	--
	<protocol.id> ***tcpID*** </protocol.id>	--
	<src.app> ***SourceNode*** </src.app>	--
	<dst.app> ***DestinationNode*** </dst.app>	--

	<rate> ***DataRate*** </rate>	onoff***expID***.SetAttribute ("***DataRate*** ", StringValue ("<Value>kb/s"));
	<packetsize>***PacketSize***</packetsize>	onoff***expID***.SetAttribute ("***PacketSize*** ", UintegerValue (<Value>));
	<bursttime></bursttime>	--
	<iddletime></iddletime>	--
	</exponential>	--
Criação de uma aplicação Pareto	<pareto id="***paretoID***">	(instance OnOffHelper generic application) onoff***paretoID***.SetAttribute ("OnTime", RandomVariableValue (ParetoVariable (double ***m***, double ***s***))); onoff***paretoID***.SetAttribute ("OffTime", RandomVariableValue (ParetoVariable ())); Where ***m*** is a specific mean and ***s*** is a shape parameter (both optional) **Note:** ***s*** must be greater than 1
	<role></role>	--
	<protocol.id> ***tcpID*** | ***udpID*** </protocol.id>	--
	<src.app> ***SourceNode*** </src.app>	--
	<dst.app> ***DestinationNode*** </dst.app>	--
	<rate> ***DataRate*** </rate>	onoff***paretoID***.SetAttribute ("***DataRate*** ", StringValue ("<Value>kb/s"));
	<packetsize> ***PacketSize*** </packetsize>	onoff***paretoID***.SetAttribute ("***PacketSize*** ", UintegerValue (<Value>));
	<bursttime></bursttime>	--
	<iddletime></iddletime>	--
	<shape>***s***</shape>	(included on constructor)
	</pareto>	--
Visualização		
Criação do container responsável por possuir a especificação das vistas	<visualizations>	--
	<visualization id=""></visualization>	--
	</visualizations>	--
Criação de uma vista	<ns3visualization id="">	--
	<description>	--
	<general>	--
	<bg.image></bg.image>	--

	<init.zoom></init.zoom>	--
	</general>	--
	<object id="***objectID***">	--
	<x.position>**X**</x.position> <y.position>**Y**</y.position> <z.position>**Z**</z.position>	positionAlloc->Add (Vector (**x**, **y**, **z**)); mobility.Install (***objectID***); positionAlloc (***ListPositionAllocator***) must be previously declared mobility (***MobilityHelper***) must be previously declared
	<image></image>	--
	<color></color>	--
	</object>	--
	</description>	--
	</ns3visualization>	--
Selecção de um conjunto de objectos	<set id="">	--
	<listobjects>	--
	<objectid></objectid>	--
	</listobjects>	--
	</set>	--
Simulação		
Criação de container de todas as simulações	<simulations>	--
	<ns3simulation id=""></ns3simulation>	--
	</simulations>	--
Criação de uma simulação	<ns3simulation id="***SimulationID***">	--
	<description>	--
	<extra>	Will hold extra parameters according to target platform language
	<routing>	--
	<aodv id="***AodvID***" />	(Covered in "Protocolos de Encaminhamento")
	<dsdv id="***DsdvID***" />	(Covered in "Protocolos de Encaminhamento")
	<olsr id="***OlsrID***" />	(Covered in "Protocolos de Encaminhamento")
	<nix.vector id="***NixVectorID***" />	(Covered in "Protocolos de Encaminhamento")
	</routing>	--
	<global.variables>	--
	<realtime><Value></realtime>	If <Value> is set to "true" GlobalValue::Bind ("SimulatorImplementationType", StringValue ("ns3::RealtimeSimulatorImpl"));
	<enable.checksu	If <Value> is set to "true"

	m><Value></enable.checksum>	GlobalValue::Bind ("ChecksumEnabled", BooleanValue (true));
	</global.variables>	--
<extra>		--
<general>		--
<duration>		--
<time> *StopTime* </time>		Simulator::Stop(Seconds(<Value>));
<condition></condition>		--
</duration>		--
<seed></seed>		--
<runs></runs>		--
<simulator></simulator>		--
</general>		--
<events>		--
<event>		--
<objectid></objectid>		--
<time></time>		--
<parameter>		--
<name></name>		--
<value></value>		
</parameter>		--
</event>		--
</events>		--
<outputs>		--
<output>		--
<outputid></outputid>		--
<source> ***linkID***</source>		AsciiTraceHelper ascii; ***linkID***Connection.EnableAsciiAll (ascii.CreateFileStream (“<filename>**.tr**"));
<type ***nstrace*** </type>		
<format>***nstrace***</format>		
<path></path>		
<filename>*FileName*</filename>		***linkID*** must be previously declared
</output>		--
<output>		--
<outputid></outputid>		--
<source> ***linkID*** </source>		***linkID***Connection. EnablePcapAll (“<filename>");
<type ***pcap***</type>		
<format>***pcap***</format>		***linkID*** must be previously declared
<path></path>		

	<filename>*FileName*</filename>	
	</output>	--
	</outputs>	--
	</description>	--
	</ns3simulation>	--
Protocolos de Encaminhamento		
Instanciação de protocolo de encaminhamento AODV global ou por nó	<aodv id="***AodvID***">	AodvHelper ***AodvID***;
	<interval> ***Interval*** </interval>	***AodvID***.Set("***HelloInterval*** ", TimeValue (Seconds(<Value>)));
	<retries> ***Retries*** </retries>	***AodvID***.Set("***RreqRetries***", UintegerValue (<Value>));
	<rate.limit> ***RateLimit*** </rate.limit>	***AodvID***.Set("***RreqRateLimit***", UintegerValue (<Value>));
	<node.traversal.time>***NodeTraversalTime*** </node.traversal.time>	***AodvID***.Set("***NodeTraversalTime***", TimeValue (MilliSeconds(<Value>)));
	<next.hop.wait>***NextHopWait*** </next.hop.wait>	***AodvID***.Set("***NextHopWait***", TimeValue (MilliSeconds(<Value>)));
	<route.timeout>***ActiveRouteTimeout*** </route.timeout>	***AodvID***.Set("***ActiveRouteTimeout***", TimeValue (Seconds(<Value>)));
	<myroute.timeout>***MyRouteTimeout*** </myroute.timeout>	***AodvID***.Set("***MyRouteTimeout***", TimeValue (Seconds(<Value>)));
	<blacklist.timeout>***BlackListTimeout*** </blacklist.timeout>	***AodvID***.Set("***BlackListTimeout***", TimeValue (Seconds(<Value>)));
	<delete.period>***DeletePeriod*** </delete.period>	***AodvID***.Set("***DeletePeriod***", TimeValue (Seconds(<Value>)));
	<timeout.buffer>***TimeoutBuffer*** </timeout.buffer>	***AodvID***.Set("***TimeoutBuffer***", UintegerValue (<Value>));
	<net.diameter>***NetDiameter*** </net.diameter>	***AodvID***.Set("***NetDiameter***", UintegerValue (<Value>));
	<net.traversal.time>***NetTraversalTime*** </net.traversal.time>	***AodvID***.Set("***NetTraversalTime***", TimeValue (Seconds(<Value>)));
	<path.discovery.time>***PathDiscoveryTime*** </path.discovery.time>	***AodvID***.Set("***PathDiscoveryTime***", TimeValue (Seconds(<Value>)));
	<queue.length>***MaxQueueLen*** </queue.length>	***AodvID***.Set("***MaxQueueLen***", UintegerValue (<Value>));
	<queue.time>***MaxQueueTime*** </queue.time>	***AodvID***.Set("***MaxQueueTime***", TimeValue (Seconds(<Value>)));
	<loss>***AllowedHelloLoss*** </loss>	***AodvID***.Set("***AllowedHelloLoss***", UintegerValue (<Value>));
	<gratuitous.reply>***GratuitousReply*** </gratuitous.reply>	***AodvID***.Set("***GratuitousReply***", BooleanValue (<Value>));
	<destination.only>***DestinationOnly*** </destination.only>	***AodvID***.Set("***DestinationOnly***", BooleanValue (<Value>));
	<enable.hello>***EnableHello*** </enable.hello>	***AodvID***.Set("***EnableHello***", BooleanValue (<Value>));
	<enable.broadcast>***EnableBroadcast*** </enable.broadcast>	***AodvID***.Set("***EnableBroadcast***", BooleanValue (<Value>));
	</aodv>	***AodvID***.Create(<***node***>); *For single node routing* ***internet***.SetRoutingHelper (***AodvID***); *For global routing*

		With, <*node*> is the element that holds *AodvID* instance; *internet* is an instance of *InternetStackHelper.*
Instanciação de protocolo de encaminhamento DSDV global ou por nó	<dsdv id="*DsdvID*">	DsdvHelper *DsdvID*;
	<update.interval>*PeriodicUpdateInterval*</update.interval>	*DsdvID*.Set("*PeriodicUpdateInterval*", TimeValue (Seconds(<Value>)));
	<settling.time>*SettlingTime*</settling.time>	*DsdvID*.Set("*SettlingTime*", TimeValue (Seconds(<Value>)));
	<queue.length>*MaxQueueLen*</queue.length>	*DsdvID*.Set("*MaxQueueLen*", UintegerValue (<Value>));
	<queue.packets.peer.dst>*MaxQueuedPacketsPerDst*</queue.packets.peer.dst>	*DsdvID*.Set("*MaxQueuedPacketsPerDst*", UintegerValue (<Value>));
	<queue.time>*MaxQueueTime*</queue.time>	*DsdvID*.Set("*MaxQueueTime*", TimeValue (Seconds(<Value>)));
	<buffering>*EnableBuffering*</buffering>	*DsdvID*.Set("*EnableBuffering*", BooleanValue (<Value>));
	<wst>*EnableWST*</wst>	*DsdvID*.Set("*EnableWST*", BooleanValue (<Value>));
	<hold.times>*Holdtimes*</hold.times>	*DsdvID*.Set("*Holdtimes*", UintegerValue (<Value>));
	<weighted.factor>*WeightedFactor*</weighted.factor>	*DsdvID*.Set("*WeightedFactor*", DoubleValue (<Value>));
	<route.aggregation>*EnableRouteAggregation*</route.aggregation>	*DsdvID*.Set("*EnableRouteAggregation*", BooleanValue (<Value>));
	<route.aggregation.time>*RouteAggregationTime*</route.aggregation.time>	*DsdvID*.Set("*RouteAggregationTime*", TimeValue (Seconds(<Value>)));
	</dsdv>	*DsdvID*.Create(<*node*>); *For single node routing* *internet*.SetRoutingHelper (*DsdvID*); *For global routing* With, <*node*> is the element that holds *DsdvID* instance; *internet* is an instance of *InternetStackHelper.*
Instanciação de protocolo de encaminhamento OLSR global ou por nó	<olsr id="*OlsrID*">	DsdvHelper *OlsrID*;
	<interval>*HelloInterval*</interval>	*OlsrID*.Set("*HelloInterval*", TimeValue (Seconds(<Value>)));
	<tc.interval>*TcInterval*</tc.interval>	*OlsrID*.Set("*TcInterval*", TimeValue (Seconds(<Value>)));
	<mid.interval>*MidInterval*</mid.interval>	*OlsrID*.Set("*MidInterval*", TimeValue (Seconds(<Value>)));
	<hna.interval>*HnaInterval*</hna.interval>	*OlsrID*.Set("*HnaInterval*", TimeValue (Seconds(<Value>)));
	<willingness>*Willingness*	*OlsrID*.Set("*Willingness*",

	</willingness>	EnumValue (<Value>));
	</olsr>	***OlsrID***.Create(<***node***>); *For single node routing* ***internet***.SetRoutingHelper (***OlsrID***); *For global routing* With, <***node***> is the element that holds ***OlsrID*** instance; ***internet*** is an instance of ***InternetStackHelper.***
Instanciação de protocolo de encaminhamento NixVector global ou por nó	<nix.vector id="***NixVectorID***">	Ipv4NixVectorHelper ***NixVectorID***;
	</nix.vector>	***NixVectorID***.Create(<***node***>); *For single node routing* ***internet***.SetRoutingHelper (***NixVectorID***); *For global routing* With, <***node***> is the element that holds ***NixVectorID*** instance; ***internet*** is an instance of ***InternetStackHelper.***


Centro de competência de
Ciências Exactas
e da Engenharia


UNIVERSIDADE da MADEIRA