Rodrigo José Fernandes de Almeida Santos

Monitorização de Rede 5G

5G Networks Management

Universidade de Aveiro
2024

Rodrigo José Fernandes de Almeida Santos

Monitorização de Rede 5G

5G Networks Management

Relatório de Estágio apresentado à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Doutor Rui Luís Andrade Aguiar, Professor catedrático do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

o júri / the jury

presidente / president

Prof. Doutor Joaquim João Estrela Ribeiro Silvestre Madeira
professor auxiliar da Universidade de Aveiro

vogais / examiners committee

Prof. Doutor Rui Lopes Campos
professor auxiliar da Faculdade de Engenharia da Universidade do Porto

Prof. Doutor Rui Luís Andrade Aguiar
professor catedrático da Universidade de Aveiro

agradecimentos / acknowledgements

Gostaria de começar por agradecer à minha mãe, ao meu pai e aos meus avós por todos os valores transmitidos e por terem sido o pilar que me manteve firme e me fez chegar até ao fim deste ciclo. Ao Nuno, que embora tenha aparecido a meio deste percurso, me deu bastantes conselhos e sempre apoiou as minhas escolhas. Aos meus amigos que tornaram esta experiência memorável. Levo todas as aventuras e ensinamentos comigo. Agradeço também à Sofia Amaral que esteve sempre presente em todos os momentos, confiou em mim e nas minhas capacidades e tornou-me uma pessoa melhor. Obrigado pelo carinho e paciência ao longo destes anos. E, por fim, a todos os professores com que me cruzei, um obrigado.

Palavras Chave Rede GSM-R, Monitorização de redes, Telecomunicações

Resumo

Atualmente, as empresas de telecomunicações geram um enorme volume de dados a um ritmo muito elevado devido ao crescimento em escala, à complexidade e aos requisitos de desempenho da rede. Este crescimento tornou a qualidade de gestão da rede num desafio ainda maior porque põe em causa a sua estabilidade. O que resulta numa monitorização regular para identificar problemas e reagir o mais breve possível. Este relatório de estágio foca-se na implementação de uma solução com vista a supervisionar o desempenho da Rede de Transporte de Pacotes para um dado nó BTS-R. Com o objetivo de lidar com qualquer desvio de desempenho que possa afetar o comportamento nominal dos serviços GSM-R nos utilizadores finais, desenvolveu-se uma aplicação robusta e visualmente atrativa, baseada num serviço de gestão de rede já existente. O sistema é capaz de processar e guardar a informação estatística contida em contadores obtida após a monitorização das interfaces do nó. Ainda, apresenta a informação recolhida, numa *dashboard user-friendly*, em gráficos sendo possível integrar os valores em diferentes regiões de acordo com a sua severidade para a rede. Mais ainda, permite extrair um documento que contém os dados recolhidos em cada interface, bem como uma imagem do gráfico correspondente. Além disto, ainda são criados relatórios semanais com alarmes gerados caso um número repetido de eventos, acima de um determinado limiar de risco, ocorram. No que toca a tecnologias, o Java foi usado como linguagem de programação dos algoritmos de processamento de dados, a biblioteca Java Swing para a interface gráfica e, ainda, a *framework* Apache Maven que permitiu a integração de dependências úteis ao desenvolvimento.

Keywords GSM-R network, Network management, Telecommunications

Abstract

Today, telecom companies generate a huge volume of data at a very high rate due to growth in scale, complexity, and network performance requirements. This growth has made the quality of network management an even greater challenge because it challenges network stability. This results in regular monitoring to identify problems and react as soon as possible. This project focuses on the implementation of a solution to monitor the performance of the Packet Transport Network for a given BTS-R node. In order to handle any performance deviation that may affect the nominal behavior of GSM-R services on end-users, it is intended to build a robust and visually attractive application based on an existing network management service. The system is capable of processing and storing the statistical information contained in counters obtained after monitoring the node interfaces. It also presents the collected information, in a user-friendly dashboard, in graphs, being possible to integrate the values in different regions according to their severity for the network. Moreover, it extracts a document containing the data collected on each interface, as well as an image of the corresponding chart. In addition to this, weekly reports are also created with alarms generated if a repeated number of events, above a certain risk threshold, occur. In terms of technologies, Java was used as the programming language for the data processing algorithms, the Java Swing library for the graphical interface. Also, the Apache Maven framework that allowed the integration of dependencies useful for development.

Conteúdo

Lista de Figuras

Lista de Tabelas

Glossário

BSC-R IP	*RDN.access controller IP*
BSM	***BTS Site Manager***
BSS	*Base Station Subsystem*
BTS-R IP	*RDN.base station IP*
EDGE	*Enhanced Data for GSM Evolution*
GPRS	*General Packet Radio Service*
GSM	*Global System for Mobile Communications*
GSM-R	*Global System for Mobile Communications - Railway*
IMEI	*International Mobile Equipment Identity*
JFC	*Java Foundation Classes*
MGW-R	*Media Gateaway - Railway*
MS	*Mobile Station*
MSC-S	Centro de Comutação de Serviços Móveis
NSS	*Network and Switching Subsystem*
OMC-R	*RDN.access network manager*
OAM	*Operations, Administration and Maintenance*
ODIAG	*DIAGnostic Observation*
OFS	*Fast Statistic Observation*
OGS	*General Statistic Observation*
OIP	Observação IP
ORT	*Real Time Observation*
OSS	*Operations Subsystem*
OVFS	*Very Fast Statistic Observation*
PCU-R IP	*RDN.packet controller IP*
RHEL	*Red Hat Enterprise Linux*
RPM	*RPM Package Manager*
RTCP	*Real-Time Transport Control Protocol*
SIM	*Subscriber Identity Module*
TDMA	*Time Division Multiple Access*
YUM	*Yellowdog Updater, Modified*

CAPÍTULO 1

Introdução

Este capítulo descreve a motivação para o desenvolvimento deste trabalho, abordando a relevância da qualidade de gestão das redes e o trabalho relativo aos mecanismos de monitorização. São igualmente abordados os objetivos esperados para o resultado final, bem como a estrutura deste documento.

1.1 Enquadramento

Atualmente, as empresas de telecomunicações geram um enorme volume de dados a um ritmo muito elevado devido ao crescimento em escala, à complexidade e aos requisitos de desempenho da rede. Este crescimento tornou a qualidade de gestão da rede num desafio ainda maior e coloca limites aos métodos de análise que podem ser aplicados,[1].

Tradicionalmente, a abordagem centrava-se no conhecimento humano adquirido pela experiência de trabalho. Hoje em dia, esta técnica jamais é aplicável nas redes de grande escala que precisam de ser monitorizadas regularmente para identificar problemas e reagir de modo a assegurar a estabilidade. Nas redes celulares, as anomalias são normalmente identificadas através de alarmes, mas nem todos os alarmes são relevantes. Existe um subconjunto designado por alarmes de causa raiz e a sua análise é um processo de enorme relevância para a localização precisa e eficiente de falhas bem como, a sua recuperação. A gestão de redes envolve um conjunto de atividades e técnicas que são necessárias para planear, conceber, controlar, manter e definir uma estrutura de rede e os seus serviços associados. Estas atividades incluem a monitorização da rede e a capacidade de tomar medidas rápidas para manter os objetivos do *service-level* e controlar o fluxo de tráfego quando necessário. As atividades de gestão da rede também incluem a deteção, identificação, investigação e resolução de elementos de rede e instalações de transmissão defeituosos. O objetivo geral é fornecer uma entrega eficiente de informações entre os utilizadores finais e os vários elementos da rede, [2].

Na gestão de redes de telecomunicações, os principais problemas são o elevado desempenho e a fiabilidade da transmissão de mensagens entre a estação de controlo e os dispositivos controlados. O requisito de elevado desempenho resulta do elevado número de dispositivos na

rede e da sua sensibilidade a vários parâmetros, como as condições climatéricas, as avarias, entre outras. A questão da fiabilidade deve-se à necessidade de uma avaliação precisa da qualidade dos serviços prestados com base na evolução do estado da rede,[3].

A rede de Transporte de Pacotes de Dados é multifacetada e apresenta uma heterogeneidade de comportamentos entre os caminhos da rede. Pode ser implementada com equipamentos de rede de diferentes modelos e fabricantes, o que conduz ao aumento da latência e do *jitter* durante a interconexão dessas redes. Algumas partes da rede podem estar isoladas em zonas de segurança específicas, protegidas por *firewalls* ou sistemas de segurança de inspeção profunda. Além disso, é importante salientar que essa complexidade na rede também pode resultar em desafios adicionais. Isso inclui a necessidade de uma gestão cuidadosa dos recursos de rede, como a alocação adequada de largura de banda para satisfazer as demandas de diferentes serviços e garantir a qualidade de serviço (QoS) quando necessário.

1.2 Objetivos

Para lidar com qualquer desvio de desempenho que possa afetar o comportamento nominal dos serviços *Global System for Mobile Communications - Railway* (GSM-R) nos utilizadores finais, surgiu este pedido, do *Product Line Manager*, para encontrar uma solução em que o principal objetivo é supervisionar o desempenho da Rede de Transporte de Pacotes para um nó *RDN.base station IP* (BTS-R IP).

Neste campo, pretende-se executar um conjunto de tarefas que possam constituir um contributo relevante para o sucesso de implementação de uma aplicação robusta e visualmente atrativa. Assim, pretendem-se como objetivos gerais:

- Processar e guardar a informação estatística contida nos contadores obtidos após a monitorização das interfaces do nó;
- Apresentar a informação recolhida numa *dashboard user-friendly* com as seguintes funcionalidades:
  - Funcionar ao nível da entidade de gestão e manutenção da rede de acesso;
  - Visualizar o desempenho do nó em gráficos;
  - Extrair um documento que contenha os dados recolhidos em cada interface;
  - Criar relatórios semanais com os alarmes gerados internamente.

1.3 Estrutura do Relatório de Estágio

Este capítulo introdutório, descreve o enquadramento do tema proposto com o objetivo de compreender o contexto onde se insere a ferramenta desenvolvida. Ainda apresenta os objetivos propostos para o resultado final da ferramenta.

Segue-se o Capítulo 2 que apresenta o estado de arte. Este capítulo divide-se em duas secções: a primeira apresenta uma conceptualização da rede GSM-R e a interligação com a aplicação de supervisão com a entidade de gestão e monitorização da rede; a segunda apresenta as tecnologias e ferramentas usadas para o desenvolvimento da aplicação.

O Capítulo 3 descreve os requisitos levantados para o desenvolvimento da ferramenta de supervisão. É, igualmente, descrita a arquitetura da solução proposta e a metodologia adotada para o cumprimento dos objetivos iniciais.

O Capítulo 4 refere-se a todos os processos de desenvolvimento envolvidos na criação da ferramenta, tais como: (i) a configuração do ambiente de desenvolvimento, (ii) a exposição dos algoritmos de processamento, (iii) a criação da interface gráfica e (iv) a criação de uma *package*. Ainda é apresentado um guia de utilização para os futuros utilizadores.

O Capítulo 5 aborda o resultado final da aplicação e fornece uma resposta ao levantamento de requisitos. Ainda é feita uma análise funcional da ferramenta como resultado do *feedback* dos utilizadores.

Por fim, no Capítulo 6 abordam-se as conclusões e uma análise final baseada nos testes e validações que foram feitas.

CAPÍTULO 2

Estado de Arte

Neste capítulo, são descritos os conceitos utilizados ao longo do relatório de estágio, bem como as tecnologias e ferramentas que foram utilizadas no desenvolvimento desta aplicação. O objetivo é definir o conjunto de conceitos que fazem parte do contexto em que a ferramenta de supervisão está inserida. Para alcançar este objetivo, foi realizada uma revisão técnica dos documentos relacionados com a matéria em estudo. Estes documentos designam-se por *Network Technical Publication* e encontram-se na plataforma *Sharepoint* da Kontron, disponíveis a toda a comunidade.

2.1 Rede GSM-R

A rede GSM-R é um sistema de rádio pan-europeu que cobre as necessidades de telecomunicações móveis dos caminhos de ferro europeus. Esta rede usa a tecnologia *Global System for Mobile Communications* (GSM) padrão e alguns recursos adicionais personalizados para as operações ferroviárias.

A rede GSM é composta por algumas plataformas de tecnologia sem fios. é um sistema de radiotelefonia digital versátil e aberto. Foi concebido para reduzir não só os custos de instalação, mas também os custos de funcionamento da rede, nomeadamente, os custos de transmissão. O *General Packet Radio Service* (GPRS) é um serviço de dados por pacotes sem fios que constitui uma extensão da rede GSM. Proporciona um método eficiente de transferência de dados, otimizando a utilização dos recursos da rede. O sistema de atribuição de recursos de rádio GPRS é utilizado para fornecer vários canais de rádio a um único utilizador, de modo a atingir um elevado débito de dados. O EDGE é uma extensão da rede de acesso GSM/GPRS. Neste sentido, herda em grande parte a administração, a manutenção e a supervisão do *Base Station Subsystem* (BSS), [4].

Um sistema GSM alberga os seguintes componentes:

- *Mobile Station* (MS)
- BSS
- *Network and Switching Subsystem* (NSS)

- *Operations Subsystem* (OSS)

A arquitetura funcional de uma rede GSM é apresentado na Figura 2.1. A rede está dividida em duas partes, o núcleo GSM-R o qual liga todas as estações base ao BSC-R IP, e depois para o Centro de Comutação de Serviços Móveis (MSC-S), e a parte de controlo que é digital e baseada em IP. O comportamento da rede é implementado na parte nuclear, principalmente no MSC-S.

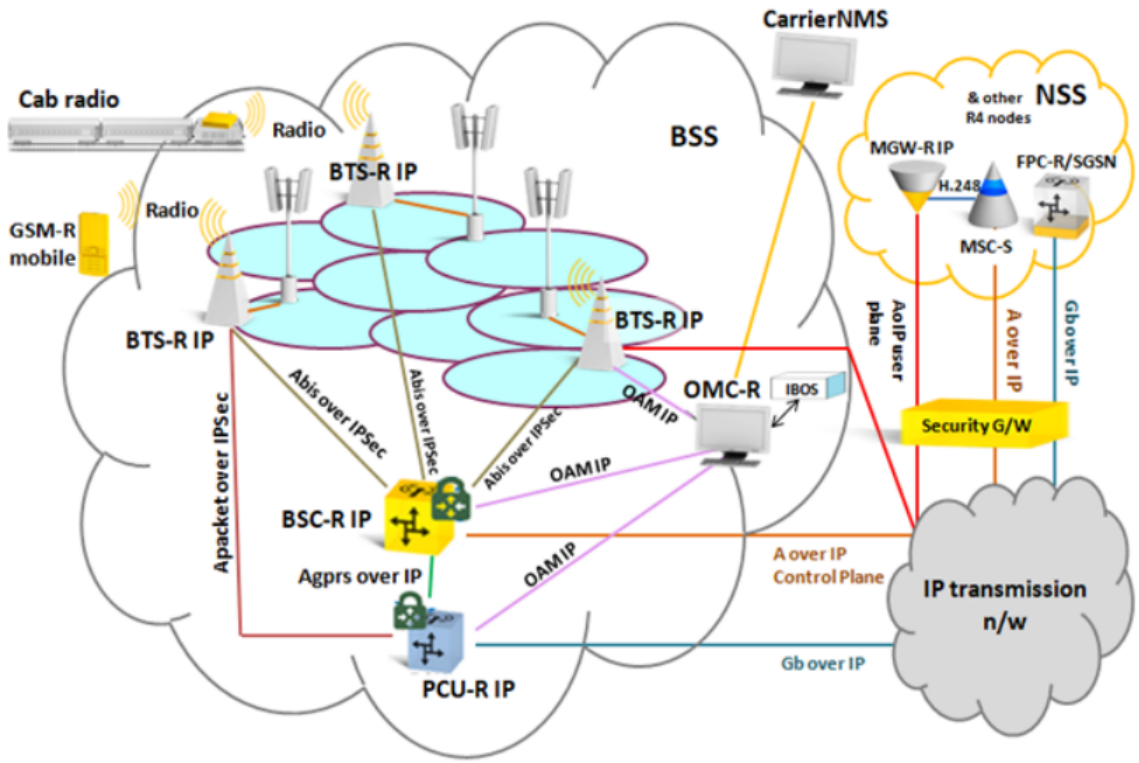

Figura 2.1: Arquitetura funcional do sistema GSM, [5]

2.1.1 Estação Móvel

A estação móvel é constituída pelo equipamento móvel (o terminal) e por um cartão inteligente denominado *Subscriber Identity Module* (SIM). O SIM proporciona uma mobilidade pessoal, para que o utilizador possa ter acesso aos serviços subscritos, independentemente de um terminal específico. Ao inserir o cartão SIM noutro terminal GSM, o utilizador pode receber chamadas nesse terminal, fazer chamadas a partir desse terminal e receber outros serviços subscritos. O equipamento móvel é identificado de forma inequívoca pelo *International Mobile Equipment Identity* (IMEI).

2.1.2 Subsistema de Rádio

O BSS, ou Subsistema de Rádio, assegura a função de distribuição da rede de comunicações. É composto pelos seguintes nós:

- BTS-R IP
- *RDN.access controller IP* (BSC-R IP)

- *RDN.packet controller IP* (PCU-R IP)
- OMC-R

A BTS-R IP inclui as *base transceiver stations* que fornecem aos utilizadores móveis a ligação rádio. Neste nó, são integradas as funções das BTS na rede de acesso IP e são controladas por um controlador de estação de base.

O BSC-R IP, que funciona num servidor HP baseado numa arquitetura x86, integra as funções do BSC na rede de acesso IP. O nó interage com a rede central através de uma interface, *A over IP* através de protocolos de comunicação nativos do IP. Este nó está ligado ao PCU-R IP através da interface *Agprs over IP*.

O PCU-R IP, que funciona num servidor HP baseado numa arquitetura x86, existe como um nó separado no BSS e é utilizado para fornecer o processamento específico de pacotes do GPRS. Nesta rede, o nó está ligado ás BTS através da interface *Apacket* e ao BSC através da interface *Agprs over IP*.

2.1.3 Subsistema de Rede e *Switching*

O NSS, ou subsistema de rede, trata de todas as funções de *routing* e *switching*. As redes de comunicação orientadas para a mobilidade exigem a localização de uma estação móvel antes de uma chamada poder ser encaminhada e configurada. O MSC-S é responsável pelo *routing* e pelo *switching*.

2.1.4 Subsistema de Operações e Suporte

O OSS, ou subsistema de operações e suporte, contém o gestor de rede de acesso, o OMC-R, que é executado numa VM dedicada num servidor com arquitetura x86 e sistema operativo HP/Linux. Com a adesão à mais recente tecnologia de virtualização, a Kontron introduziu soluções que se baseiam nesta tecnologia para suportar a virtualização de aplicações *Operations, Administration and Maintenance* (OAM) em servidores Linux com uma arquitetura x86.

2.2 OMC-R

Um dos principais componentes da rede GSM-R é o OMC-R responsável pela gestão e manutenção da camada de acesso, incluindo o controlador da estação base, as unidades codificadoras de transmissão e uma vasta gama de BTS-R IP. Exerce as funções do OAM nas diferentes áreas de suporte á gestão de dados, como por exemplo, na Gestão de Falhas, na Segurança, na Configuração, no Desempenho e na Administração.O OMC-R inclui duas entidades, um gestor e um agente. O agente, denominado MD-R, suporta a função de mediação.

Este componente da rede fornece as seguintes funções:

- Interface gráfica (MMI) que consiste numa seleção de ecrãs e comandos utilizados para aceder às funções do OMC-R. Está organizada em torno de cinco tipos de ecrãs diferentes onde cada uma representa uma função específica.
- Gestão das comunicações (COM) que fornece um conjunto de serviços usados por outras funções deste componente.

- Gestão da configuração da BSS (CM) que processa os comandos da interface nos objetos BSS e executa as ações requisitadas nas base de dados.
- Gestão de Falhas (FM) que monitoriza permanentemente as operações do sistema, e o OMC-R processa os relatórios dos eventos para levantar alarmes ou notificações.
- Gestão de Desempenho (PM) que recolhe e processa observações na BSS e no OMC-R para construir relatórios com estatísticas de operação do sistema.
- Gestão da Segurança (SM) que gere o acesso aos serviços do utilizador do OMC-R.
- Funções comuns (CF) que reagrupa todas as funções de operação e manutenção.
- Bases de Dados.
- Gestão GPRS.
- Gestão da rede IP (ou IP BTS OAM *Services* (IBOS)).

Cada uma destas funções envolve um agente e um gestor local.

2.2.1 Interface SDO

O SDO apresenta os dados de desempenho e de configuração, as notificações de gestão de falhas, as zonas de interesse da estação de trabalho, o registo da sessão e os dados do rastro das chamadas num formato ASCII predefinido, para aplicações de pós-processamento.

No modo de processamento automático, sempre que o SDO recebe um evento OMC-R sobre a disponibilidade de registos de observação ocorre uma operação de transferência. Os diferentes registos de observação estatística carregados a partir do OMC-R são codificados em ASN.1. O SDO descodifica-os, extrai os valores dos contadores de acordo com a lista de contadores definida num ficheiro de configuração específico, que associa cada um nome a um contador, e constrói os ficheiros de resultados associados.

Um registo de observação OMC-R contém dados de observação para apenas uma BSC. Uma BSC pode conter várias BTS.

Existe um processo automático de compressão que, todas as noites, imediatamente após a limpeza diária de ficheiros, comprime todos os registos e resumos de observação em bruto do dia anterior em ficheiros comprimidos do tipo tar.gz, apagando os ficheiros originais. Normalmente, apenas os dados de observação do dia atual estão disponíveis em formato descomprimido. Apenas os ficheiros são apagados por esta rotina, os diretórios não. Este processo pode sempre ser desativado utilizando a interface gráfica do SDO.

Há dois tipos de dados de monitorização a comprimir: brutos e somas diárias. Como estão armazenados em dois diretórios diferentes, existem dois ficheiros comprimidos distintos armazenados na mesma pasta. Os ficheiros comprimidos são armazenados no seguinte diretório:

- Diretório "compressed_extended/<Data>"a ser criado, em que o formato <Data> é AAAAMMDD, em que AAAA = ano, MM = mês, DD = dia.

Os nomes dos ficheiros são atribuídos se os dados dos contadores a recuperar não forem carregados durante o dia em curso, então esses dados estão contidos num ficheiro tar.gz armazenado num diretório específico:

- obs_raw.tar.gz para os registos brutos do *Fast Statistic Observation* (OFS), *General Statistic Observation* (OGS) e PCU;
- obs_sum.tar.gz para registos de somas diárias acofs e PCU;

- obs_extended_raw.tar.gz para registos brutos acovfs (os que servem como input da ferramenta de pós processamento);
- obs_extended_sum.tar.gz para registos de somas diárias acofs.

Existem dois tipos de mecanismos de exclusão, ao nível do SDO, aplicados aos ficheiros comprimidos que contêm os relatórios de observação:

- Exclusão diária, para remover ficheiros mais antigos que o período configurado. É possível configurar períodos diferentes para cada ficheiro de compressão diferente. Por defeito, o período configurado será:
  - Três dias para os ficheiros obs_raw.tar.gz e obs_sum.tar.gz que contêm registos OFS e OGS;
  - Vinte e um dias para os ficheiros obs_extended_raw.tar.gz e obs_extended_sum.tar.gz que contêm registos OVFS;
- Exclusão de defesa para evitar a saturação do disco.

2.2.2 Estrutura Comunicacional

As entidades físicas do OMC-R comunicam entre si e com o ambiente OMC-R por meio de três redes. O OMC-R e a BSS comunicam por IP através do BSC. A interface formada pela ligação entre a rede e o OMC-R, e a rede e o BSC é designada por interface de comunicação. Esta interface é responsável por permitir a existência de comunicação entre o OMC-R e a BSS através de ligação IP. O OMC pode estar conectado a um conjunto de diferentes BSSs.

O diagrama da Figura 2.2 contém os subsistemas que completam a rede e a informação é trocada entre os seguintes:

- gestor local e MD-R.
- estação de trabalho local e servidor (ativo) OMC-R.
- servidor (ativo) OMC-R e BSC.

A comunicação entre um gestor e o MD-R ocorre por meio de uma interface Q3 interna (exceto para a gestão de notificações).

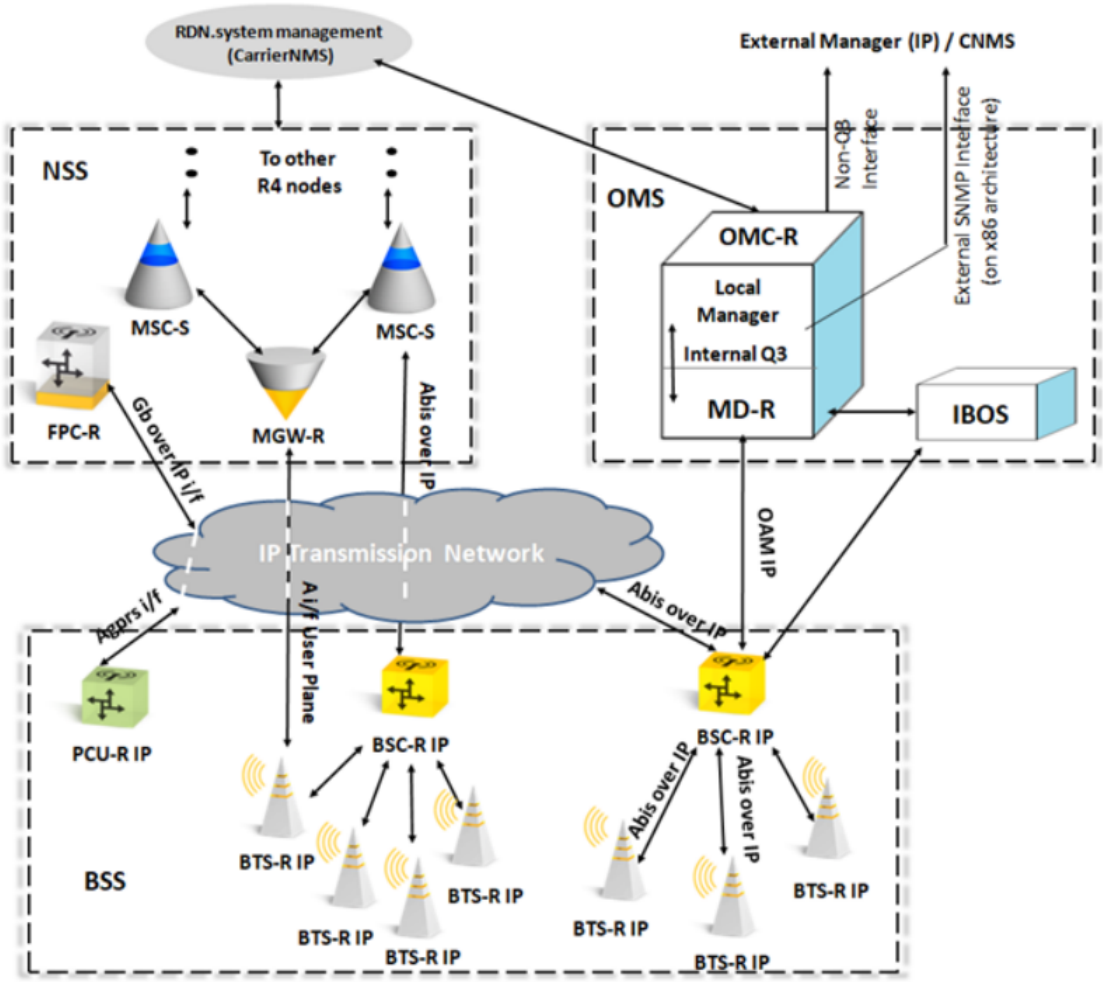

Figura 2.2: Posição do OMC-R na rede, [5].

2.3 Nó BTS-R IP

A BTS-R IP é um novo tipo de BTS dedicado aos clientes dos caminhos de ferro na banda de frequência GSM-R. A BTS-R IP, que se destina a vários ambientes operacionais, vem substituir ou complementar as BTS antigas do portefólio da Kontron para implementações em instalações novas.

2.3.1 Interfaces

A BTS-R IP suporta as seguintes interfaces baseadas na transmissão IP para a implementação da rede de acesso IP:

- IBOS;
- Interface *Abis over IP* com o BSC-R IP;
- Interface *A over IP (user plane)* com o *Media Gateaway - Railway* (MGW-R);
- Interface *Apacket* com PCU-R IP;
- Interface Debug/TIL.

Apenas as interfaces *Apacket* e *A over IP* estão a ser monitorizadas e, por isso, só esses processos de monitorização é que são abordados.

2.3.2 Monitorização Interface *A-packet*

A interface *Apacket* é introduzida na rede de acesso IP para *packet switching (PS)* em *UpLink* e *DownLink* entre o PCU-R IP e a BTS-R IP. Esta interface é também uma comunicação que utiliza transporte IPSec sobre UDP. Nesta interface, cada canal corresponde a um *Time Division Multiple Access* (TDMA). Assim, serão mantidos 12 desses canais para a comunicação do tráfego em *uplink* e *downlink*. O protocolo de transporte IPSec é utilizado nesta interface para gerir os requisitos de segurança.

Com a introdução da funcionalidade de monitorização da qualidade das interfaces IP, a qualidade do serviço de uma ligação *A-packet* entre a BTS-R IP e o PCU-R IP é monitorizada. A interface de supervisão IP *A-Packet* permite a verificação da atividade e a monitorização da qualidade do serviço da interface IP *A-Packet*, possibilitando a troca das seguintes mensagens entre a BTS e o PCU:

- *A-Packet Supervision Heartbeat* [ASH] para a supervisão da ligação.
- *A-Packet Supervision Heartbeat* [ASS] para monitorizar o estado da ligação.

O PCU envia as estatísticas da ligação através de contadores de medição, neste cao, *packet delay variation*, na mensagem ASS para cada célula da BTS, a cada minuto. É importante notar que o ASS não é enviado pelo BTS-R IP. No lado do PCU, os contadores ASS são acumulados durante um período ASS (ou seja, entre duas mensagens) e são apagados quando são reportados na própria mensagem a enviar.

Os contadores *Apacket* são geridos no OMC-R como um tipo de observação de contador muito rápido suportando a configuração dos contadores de monitorização. No entanto, o tipo de observação OVFS difere do OFS nos seguintes aspetos de configuração:

- Período de granularidade: os valores suportados são 3, 4 e 5 minutos.
- Periodicidade de armazenamento: definida por defeito para 21 dias ao nível do SDO para ficheiros comprimidos.

Para evitar problemas graves com o espaço em disco, a capacidade de armazenamento mantém-se igual. Ao nível do OMC, a capacidade de armazenamento de 7 dias para a observação OFS e a periodicidade de armazenamento OVFS é aumentada ao nível SDO em modo comprimido, [6].

2.3.3 Monitorização Interface *User Plane*

A interface introduzida na rede de acesso IP para *circuit switching (CS)* em *uplink* e *downlink* entre o MGW-R e a BTS-R IP designa-se por *User Plane*. Trata-se de uma comunicação sem conexão que utiliza o túnel IPSec sobre UDP. Os canais de tráfego são mantidos em cada nível TDMA para a troca de tráfego. Em Full rate, são mantidos 94 canais de tráfego, se todos os intervalos de tempo TDMA disponíveis estiverem configurados para CSD/*Speech*.

Com a introdução da funcionalidade de monitorização da qualidade das interfaces IP, a ligação do *User Plane* entre o BTS-R IP e o MGW-R é monitorizada segundo o protocolo *Real-Time Transport Control Protocol* (RTCP). O RTCP é um protocolo que acompanha

o RTP e é utilizado para medir a qualidade da transmissão de chamadas GSM sobre IP, incluindo:

- *Round-Trip Delay*;
- *Jitter* entre chegadas;
- Perda de pacotes.

Para cada chamada GSM, é aberta uma sessão RTCP dedicada no fluxo RTP correspondente entre a BTS e o MGW para trocar o estado do tráfego RTP. São avaliados aspetos como o número de pacotes RTP enviados, recebidos ou perdidos, o *jitter* observado, entre outros, através de vários contadores RTCP. Estes contadores RTCP são calculados ao nível da BTS e comunicados regularmente à BSC e apresentados no OMC-R.

Embora o protocolo RTCP meça a qualidade da ligação *AoIP*, não pode ser utilizado para verificar a conectividade IP entre a BTS e a *media gateway*, quando não há chamadas em curso nessa BTS. Para isso, existem mecanismos de *Packet Delay Variation* que atribuem contadores para verificar essa conectividade IP. Um mecanismo PDV separado implementa vários contadores PDV para monitorizar a qualidade da conectividade da rede IP.

Mesmo que fisicamente os pacotes da ligação *AoIP* cheguem ao nível da BTS-R IP, dependendo das perturbações de rede sofridas, podem ocorrer as seguintes situações:

- condição de demasiado tarde,
- ou demasiado cedo,
- ou ressincronização devido a uma má numeração da sequência dos pacotes.

As condições acima referidas são monitorizadas através de vários contadores internos das BTS-R IP e comunicadas ao OMC-R. A granularidade dos contadores RTCP e PDV é por BTS, considerando todas as chamadas suportadas pela BTS durante um determinado período. Se uma BTS gere várias MGW em simultâneo, os contadores são apresentados separadamente para cada MGW sob esta BTS no OMC-R.

2.4 Tecnologias de Suporte

Ao longo do desenvolvimento desta aplicação foram empregues várias tecnologias e ferramentas para a concretização do produto final.

2.4.1 Linguagem de Programação

A escolha do Java 1.6 como linguagem de programação advém da restrição empregue pelo ambiente de execução do OMC-R onde futuramente será inserida esta aplicação. Para que ambas fossem compatíveis, a decisão de tomar a versão 6 do Java como a linguagem para o desenvolvimento da ferramenta de supervisão impulsionou e restringiu todas as possibilidades tidas, inicialmente, para os restantes componentes da aplicação. O Java 1.6 é uma versão da linguagem de programação Java lançada em 2006. Trouxe diversas melhorias e recursos, além de ser uma versão amplamente utilizada, reconhecida pela sua estabilidade e compatibilidade.

2.4.2 Sistema Operativo

O ambiente virtual *Red Hat Enterprise Linux* (RHEL) 7 proporcionou um ambiente estável e seguro para o desenvolvimento da aplicação. A escolha desta distribuição baseou-se no facto da reputação como uma plataforma empresarial confiável e escalável que garante, assim, a robustez da aplicação durante o seu ciclo de vida. O RHEL 7 é uma distribuição de Linux desenvolvida pela *Red Hat*. É especialmente projetado para uso em ambientes corporativos e oferece suporte de longo prazo. O RHEL 7 fornece um ambiente virtual confiável para executar aplicações e serviços, garantindo a integridade e a escalabilidade do sistema.

2.4.3 IDE

O *Visual Studio Code* é um ambiente de desenvolvimento integrado (IDE) leve e poderoso, desenvolvido pela *Microsoft*. Oferece suporte a uma ampla gama de linguagens de programação, incluindo Java, e possui recursos avançados de edição, depuração e controlo de versão. A interface deste IDE é bastante intuitiva e extensível, o que permite personalizar o ambiente de desenvolvimento de acordo com as necessidades individuais. Assim, tornou-se a ferramenta principal deste projeto para o desenvolvimento do código fonte. Além desta usabilidade, teve o papel de intermediário no estabelecimento de sessões SSH com a máquina de desenvolvimento para que todas as condições reunidas fossem constantemente testadas.

2.4.4 Controlo de versões

O *GitLab* é uma plataforma web para controlo de versão e colaboração de código. Oferece recursos para gerir repositórios *Git*, realizar integração contínua, monitorizar problemas e acompanhar o progresso do desenvolvimento. Com o *GitLab*, é possível controlar as versões do código, facilitar a colaboração entre a equipa de desenvolvimento e garantir a integridade do código fonte. Assim, tornou-se uma escolha fiável para que o desenvolvimento do código fonte fosse assiduamente acompanhado.

2.4.5 Bibliotecas

A interface gráfica foi desenvolvida com o framework *Java Swing* que é uma biblioteca gráfica que faz parte do *Java Foundation Classes* (JFC). Esta biblioteca oferece um conjunto vasto de componentes gráficos como, por exemplo, botões, caixas de texto, tabelas, painéis, entre outros. Permite a criação de interfaces intuitivas e visualmente atrativas para aplicações *desktop* em Java pois é conhecido pela sua flexibilidade.

O *Apache Maven* é uma ferramenta de gestão de projetos amplamente utilizada na comunidade Java. Esta ferramenta facilita o processo de compilação, empacotamento e gestão de dependências de um projeto. O principal objetivo do *Maven* é permitir que um programador compreenda o estado completo de um processo de desenvolvimento no mais curto espaço de tempo. Para atingir este objetivo, o *Maven* preocupa-se com vários aspetos, nomeadamente:

- Facilitar o processo de compilação;
- Fornecer um sistema de compilação uniforme;
- Fornecer informações de qualidade sobre o projeto;
- Incentivar melhores práticas de desenvolvimento.

A versão 3.9.1 é uma versão mais recente do Maven e inclui melhorias de desempenho, correções de bugs e recursos aprimorados em relação às versões anteriores. Com o Maven, é possível definir as dependências do projeto num ficheiro de configuração (pom.xml) pois a ferramenta é capaz de realizar o download e a configuração das bibliotecas necessárias.

As dependências que se tornaram necessárias ao desenvolvimento desta aplicação são:

- *Apache Commons-Compress*, uma biblioteca que proporciona suporte para compressão e descompressão de ficheiros em diversos formatos, incluindo ZIP, GZIP, TAR, entre outros. A utilização desta biblioteca foi relevante para operações relacionadas com o uso de ficheiros comprimidos. Fornece uma API fácil de usar para trabalhar com estes ficheiros, permitindo manipular e extrair o conteúdo para uso futuro na aplicação.
- *JFreeChart*, uma biblioteca gráfica em Java que permite criar uma variedade de gráficos, como gráficos de barras, gráficos de linhas, gráficos circulares, entre outros. Oferece recursos avançados de personalização e é amplamente utilizado para visualização de dados em aplicações Java.
- *Apache POI*, uma biblioteca Java para manipulação de documentos do Microsoft Office, como folhas de cálculo do Excel, documentos do Word e apresentações do PowerPoint. Com o Apache POI, é possível criar, modificar e ler ficheiros nesses formatos, permitindo a integração de recursos de manipulação de documentos na aplicação.

2.4.6 Ferramenta de Gestão de Pacotes

A fase final do desenvolvimento foi a criação de um pacote de instalação para facilitar o processo de testes. Para isso, o *RPM Package Manager* (RPM) foi escolhido por ser uma ferramenta utilizada para criar pacotes de instalação no formato RPM, usualmente utilizado em distribuições Linux, como o RHEL 7. Com o RPM, foi possível criar um pacote de instalação da aplicação de forma padronizada, o que facilitou a distribuição e instalação em sistemas operacionais compatíveis com este formato.

Um pacote RPM é um ficheiro que contém outros ficheiros e os seus metadados (informações sobre os ficheiros que são necessários ao sistema). Os pacotes criados consistem numa solução simplificada para a distribuição e instalação de *software*. Existem dois tipos de pacotes RPM. Ambos os tipos partilham o formato de ficheiro e as ferramentas, mas têm conteúdos diferentes e servem propósitos diferentes:

- RPM binário que contém os binários compilados a partir das fontes e dos *patches*.
- RPM de origem (SRPM) que contém o código fonte e um ficheiro SPEC, que descreve como compilar o código fonte num RPM binário. Opcionalmente, os *patches* para o código-fonte também são incluídos.

A fase de criação de uma *package* RPM consiste, primeiramente, em construir um ficheiro de especificações designado por ficheiro SPEC. Um ficheiro SPEC é como uma receita, que o *rpmbuild* utiliza para compilar um RPM. Um ficheiro SPEC fornece as informações necessárias ao sistema de compilação definindo instruções numa série de secções. Estas secções são definidas no preâmbulo e na parte corpo do ficheiro SPEC. A parte do preâmbulo contém uma série de itens de metadados que são utilizados na parte corpo e estão apresentados na

Tabela 2.1. A parte do corpo representa a parte principal das instruções e está representada na Tabela 2.2.

Tabela 2.1: Itens geralmente usados na secção Preâmbulo dos ficheiros SPEC.

Diretiva SPEC	Definição
Nome	Nome do pacote, que corresponde ao nome do ficheiro SPEC.
Versão	Número da última versão do software.
Release	Este valor deve ser incrementado a cada nova *release* do pacote.
Sumário	Breve descrição do pacote.
Licença	Licença do software a ser empacotado.
Fonte	Caminho ou URL para o ficheiro comprimido do código fonte.
Patch	Nome do primeiro *patch* aplicado ao código fonte.
Arquitetura	Arquitetura da máquina onde é compilado ou *"noarch"* caso não seja dependente de arquitetura.
Requisitos de Compilação	Lista de dependências para compilar o programa.
Requisitos	Lista de dependências para executar o software assim que instalado.

Tabela 2.2: Itens geralmente usados na secção Corpo dos ficheiros SPEC.

Diretiva SPEC	Definição
%description	Descrição completa do software a ser incorporado pelo RPM.
%prep	Comando ou série de comandos para preparar o software a ser compilado.
%build	Comando ou série de comandos para a compilação do software.
%install	Comando ou série de comandos para a instalação dos artefactos num diretório resultante.
%check	Comando ou série de comandos para "testar"o software.
%files	Lista de ficheiros a serem instalados no sistema alvo.
%changelog	Conjunto de alterações que ocorreram entre versões ou *releases*.

Além destes itens, um ficheiro SPEC também pode conter *scriplets* ou *triggers*. Estes entram em vigor em diferentes etapas durante o processo de instalação no sistema do utilizador final. São apresentados na Tabela 2.3 os *scriptlets* que geralmente aparecem nos ficheiros SPEC.

Tabela 2.3: *Scriptlets* que podem constar nos ficheiros SPEC.

Diretiva	Definição
%pre	Executado mesmo antes do pacote ser instalado no sistema alvo.
%post	Executado mesmo depois do pacote ser instalado no sistema alvo.
%preun	Executado mesmo antes do pacote ser desinstalado no sistema alvo.
%postun	Executado mesmo depois do pacote ser desinstalado no sistema alvo.

O RPM fornece um vasto conjunto de macros para tornar a manutenção de pacotes mais simples e consistente em todos os pacotes. Por exemplo, inclui uma lista de definições de caminhos padrão que são usados pelas macros do sistema de compilação, e definições para

diretórios específicos de compilação de pacotes RPM. Também fornece o conjunto padrão de sinalizadores do compilador como macros, que deve ser usado ao compilar manualmente e não depender de um sistema de compilação.

CAPÍTULO 3

Metodologia

Neste capítulo, é detalhada toda a metodologia aplicada no desenvolvimento da aplicação de supervisão. É feita uma especificação de requisitos do sistema, sendo identificados os requisitos funcionais e não funcionais. É apresentada a arquitetura geral do sistema, apresentando os métodos usados para a recolha dos dados estatísticos e para o desenvolvimento da interface gráfica. São igualmente indicados os dados de entrada da aplicação, neste caso, os ficheiros que contêm a informação estatística do desempenho das interfaces monitorizadas.

3.1 Requisitos Técnicos

O levantamento de requisitos é fundamental para compreender as necessidades dos utilizadores e definir as funcionalidades e características que serão implementadas tanto no *backend* como no *frontend* da aplicação. A análise de documentos e do contexto de inserção e uso da aplicação permitiram identificar os objetivos e as principais funcionalidades esperadas da aplicação.

Durante o levantamento de requisitos, foram identificados aspetos relacionados com o processamento de dados no *backend* bem como, na criação da interface gráfica ao nível do *frontend*. No que diz respeito ao processamento de dados, identifica-se:

- A necessidade de processar eficientemente grandes volumes de dados, sendo que não deve afetar o desempenho da aplicação e deve ser feito em tempo útil;
- Garantir a precisão dos cálculos e operações realizadas, bem como a consistência e a fiabilidade dos resultados;
- Garantir a integridade dos dados, sem perdas nem corrupção, através da validação e verificação continua dos dados de entrada;
- Suportar escalabilidade para um futuro aumento da quantidade de dados de entrada;
- Registo de todas as operações realizadas, o que permite a auditoria e a indentificação de alguma falha ou anomalia que possa ocorrer.

Já no *frontend*, levanta-se a necessidade da visualização dos dados já processados em gráficos, de forma a não afetar o desempenho geral da aplicação, a capacidade de exportar esses

gráficos em documentos Excel e uma interface robusta e intuitiva para facilitar a interação dos utilizadores com a aplicação.

3.1.1 Requisitos Funcionais

Os requisitos funcionais descrevem as funcionalidades e o comportamento esperado da aplicação, ou seja, como é que deve responder a determinados eventos e que ações deve ser capaz de realizar.

Neste caso, entende-se que a aplicação seja capaz de processar, de uma forma eficaz e eficiente, os ficheiros, recolhendo as informações necessárias à supervisão, mantendo a integridade dos valores associados. Também é esperado que a funcionalidade de configuração da supervisão, que envolve filtragem e seleção de dados, permita uma visualização personalizada dos gráficos. Em relação á apresentação visual do conteúdo gráfico, é importante que sejam intuitivos e informativos para se tirarem as conclusões corretas. E, ainda, a capacidade de exportar os gráficos em ficheiros Excel para uma análise futura sem a necessidade de estar ligado à aplicação.

Estes requisitos são essenciais para garantir o cumprimento das necessidades dos utilizadores e ir ao encontro das expectativas.

3.1.2 Requisitos Não Funcionais

Os requisitos não funcionais especificam as características que não estão relacionadas diretamente com as funcionalidades, mas que desempenham uma função importante para o desempenho e qualidade da aplicação.

Neste caso, em relação ao desempenho, é necessário garantir um processamento relativamente rápido e eficiente dos dados, bem como uma apresentação ágil dos gráficos, com alguma qualidade, nos diferentes intervalos de tempo de supervisão. Em relação à usabilidade, tem de fornecer uma interface gráfica *user-friendly* e intuitiva, que facilite a interação com os utilizadores. Ao nível da fiabilidade, é necessário assegurar a precisão dos cálculos e a credibilidade dos dados processados, de forma a evitar erros e inconsistências. Em relação à escalabilidade, deve ser garantido o possível crescimento futuro da aplicação, possibilitando novas *features* e novas configurações. Ao nível da segurança, deve estar em conformidade com as normas já definidas para a rede de acesso IP. Por último, garantir que a documentação, que servirá de suporte técnico, seja de fácil leitura e compreensão.

Estes requisitos são essenciais para garantir a segurança, a qualidade e a eficiência da aplicação.

3.2 Arquitetura

Com o objetivo de cumprir com os requisitos estabelecidos, foi desenvolvida uma *dashboard user-friendly*, completamente autónoma, que pode ser integrada com a entidade responsável pela gestão e manutenção da rede. A ferramenta funciona tanto *offline* como *online*, ou seja, pode ser executada independentemente do estado do OMC-R. A arquitetura geral do funcionamento da ferramenta de pós processamento está presente na Figura 3.1. Esta

ferramenta de pós processamento é capaz de recolher toda a informação estatística do desempenho das duas interfaces monitorizadas, que vem na forma de contadores, processar e guardar a informação em memória para, posteriormente, ser analisada pelo operador. A informação é recolhida dos ficheiros depositados pelo SDO no diretório padrão. O objetivo é fazer o *parsing* destes ficheiros, filtrando os valores dos contadores que representam o tipo de desempenho a ser avaliado.

Além deste mecanismo de processamento de dados, a ferramenta possui uma interface gráfica para a correta avaliação do desempenho do nó na rede. Integra um painel de configuração da supervisão para o operador poder personalizar os parâmetros de entrada da ferramenta, tais como, o tipo de desempenho a analisar, o período de tempo que pretende obter resultados, a data de inicio do período de duração bem como, a hora, caso pretenda alguma supervisão mais especifica e a interface a ser analizada. Após a especificação dos mesmos, um outro painel integrante desta aplicação é apresentado para revelar os gráficos, construídos com base nos dados processados. Neste painel é possível extrair um relatório de cada gráfico, com os dados e uma imagem do mesmo.

Como é normal deste tipo de aplicações de supervisão, é importante que a avaliação dos dados obtidos seja reportada em formato de alarme para o operador. Por isso, outra das funcionalidades da ferramenta é a criação de dois grandes ficheiros, cada um relativo a dois tipos de desempenho diferentes, neste caso, em relação ao *Jitter* ou *Packet Delay Variation* e ao *One Way Delay*. Estes ficheiros contêm os alarmes gerados ao longo de uma semana. Estes alarmes são o resultado da análise dos valores dos contadores e comparados com um valor de *threshold* considerado anormal para cada caso.

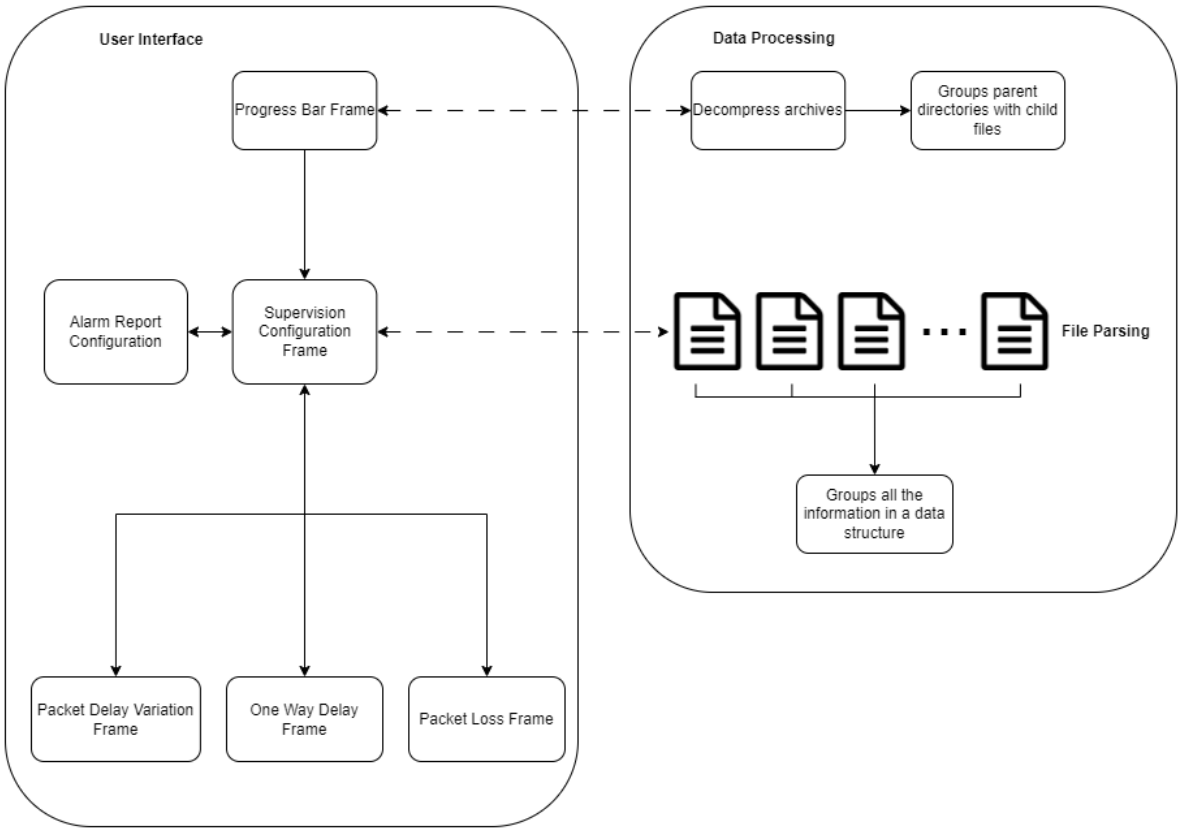

Figura 3.1: Arquitetura geral da aplicação.

3.3 Princípios de gestão de monitorização

Para avaliar o desempenho da Rede de Transporte de Pacotes, é necessário clarificar como é realizada a monitorização das interfaces. Esta secção tem como objetivo a compreensão dos contadores de observação no contexto da entidade de gestão de monitorização do OMC-R. Além de, clarificar como se relacionam os contadores de observação com os diferentes cenários de fluxo de chamadas GSM.

3.3.1 Tipos e Configurações Padrão de cada observação

As observações podem ser classificadas por:

- OGS: este tipo de observação permite a realização de estatísticas diárias;
- OFS: esta observação permite efetuar estatísticas periódicas. Pode ser configurada para recolher observações a cada 15, 30, 45 ou 60 minutos.
- *Real Time Observation* (ORT): este tipo de observação permite a deteção em tempo real de incidentes através da rede e a observação de variações rápidas da carga de tráfego.
- *DIAGnostic Observation* (ODIAG): esta observação permite a investigação de um mau funcionamento, ou a verificação e o controlo da rede (após reorganização). Esta observação reporta os contadores que estão associados a células específicas da rede. Os contadores BSC podem ser associados a esta observação.

- OVFS: esta observação permite a realização de estatísticas periódicas durante um período de observação mais curto. Pode ser configurada para recolher observações a cada 3, 4 ou 5 minutos.
- Observação IP (OIP).

3.3.2 Gestão de Observação

As observações BSS são ativadas através da criação de objectos **mdScanner** cujo atributo **mdGranularityPeriod** define os períodos em que os contadores de observação são lidos e transferidos para o agente OMC-R.

Quando uma observação BSS é ativada, todos os contadores relacionados começam a contar. Quando a observação é desativada, os contadores deixam de contar, ou seja, deixam de ser recolhidos.

A avaliação do desempenho da rede com base nos resultados da observação das entidades rádio abrange quatro domínios:

- Volume de tráfego;
- Qualidade do serviço;
- Disponibilidade de recursos;
- Controlo da configuração da rede.

3.3.3 Tipo OVFS

Este tipo de observação é bastante parecido com o tipo que já existe OFS, e é suportado para a configuração dos contadores de monitorização. Todo o tratamento exigido para estas observações deve ser replicado para o tipo OVFS, com duas diferenças principais:

- Período de **granularidade**: os valores suportados são 3, 4 e 5 minutos (3 por defeito);
- Periodicidade de armazenamento: definida por defeito para 21 dias a nível do SDO para ficheiros comprimidos.

Para evitar problemas graves com o espaçamento do disco, a solução proposta consiste em manter a mesma capacidade de armazenamento de 7 dias ao nível do OMC-R que o OFS e aumentar a periodicidade de armazenamento do OVFS ao nível do servidor de dados do OMC-R (SDO) em modo comprimido. Isto significa que os relatórios têm as mesmas facilidades que os OFS mas, caso seja necessária uma análise/evidências mais antigas, os dados estão apenas disponíveis em ficheiros comprimidos no servidor de dados OMC-R (SDO), separados por dias e por BSC em formato legível ASCII.

Após a descodificação dos registos de observação pelo SDO carregados a partir do OMC-R, é fornecido o acesso aos dados registados em formato ASCII legível. Assim, os contadores RTCP e APacket no **btsSiteManager** são adicionados aos ficheiros representados pelo *relative path* na Tabela 3.1.

Tabela 3.1: *Relative Path* dos ficheiros armazenados.

Por ficheiro de registo	/SDO/data/obs/raw/<Date>/<Network>/<BSC>/aOVFS_BSM_<UTC><HHMN>
Ficheiro diário	/SDO/data/obs/sum/<Network>/<BSC>/aOVFS_BSM.<Date>

Devido ao tamanho dos ficheiros armazenados, são utilizados algoritmos de compressão para otimizar a utilização do espaço em disco do SDO. O SDO armazena um dia de dados de observação num formato não comprimido. Todas as noites, é iniciado um procedimento de compressão e todos os dados de observação são comprimidos, excepto os do dia de observações atual. Os ficheiros comprimidos são, por defeito, armazenados no diretório '/SDO/data/obs/compressed/<Date>/':

- obs_raw.tar.gz para os registos brutos OFS, OGS e PCU;
- obs_sum.tar.gz para os registos das somas diárias do OFS e da PCU.

Uma vez que a observação OVFS requer um período de armazenamento mais longo, os respetivos ficheiros comprimidos são armazenados no diretório '/SDO/data/obs/compressed_extended/<Date>/':

- novo obs_raw_ovfs.tar.gz para registos brutos OVFS;
- novo obs_sum_ovfs.tar.gz para registos de somas diárias do OVFS.

Em ambos os casos, <Date> refere-se ao carimbo de data/hora dos registos recolhidos.

Para aceder e processar os dados contidos em ficheiros comprimidos, é necessário descomprimir os ficheiros de observação do SDO. Não se pode realizar esta ação no próprio disco do SDO pois pode causar perda de dados devido à falta de armazenamento. O tipo de mecanismo de eliminação ao nível do SDO que se pode aplicar a este tipo de observação é a limpeza diária. Serve para remover ficheiros mais antigos que o período configurado. Por defeito, é 21 dias para os ficheiros obs_raw_ovfs.tar.gz e obs_sum_ovfs.tar.gz.

Os contadores que representam os três tipos de desempenho das interfaces monitorizadas encontram-se na Tabela 3.2.

De acordo com a informação apresentada na Tabela 2.3, os contadores que representam o *Packet Loss* são do tipo sintético, ou seja, não têm unidades científicas. Os do tipo *Artp* representam o rácio de pacotes perdidos entre a *btsSiteManager* e a MGWX, em que X varia entre 0 e 3, em *UpLink* e *DownLink*, cada um representado por UL e DL, respetivamente. Já os do tipo *Apacket* representam o rácio de pacotes perdidos entre a DB0 e o PCU, em DL e UL.

Os contadores que representam o *Packet Delay Variation* são do tipo VAL, ou seja, têm unidades cientificas, neste caso, microsegundos. Os do tipo *Artp* representam o máximo dos máximos *jitters*, durante o período de observação, entre a *btsSiteManager* e a MGWX, em que X varia entre 0 e 3, em *UpLink* e *DownLink*, cada um representado por UL e DL, respetivamente. Já os do tipo *Apacket* representam o máximo dos máximos *jitters*, durante o período de observação, calculados e enviados pelo PCU para a BTS, em UL e DL.

Os contadores que representam o *One Way Delay* são do tipo VAL, com métricas de microsegundos. Os do tipo *Artp* representam máximo dos máximos do *Round Trip Delay* calculado na BTS, pelo protocolo RTCP, entre a *btsSiteManager* e a MGWX, sendo que X varia entre 0 e 3. Já os do tipo *Apacket* são calculados e acumulados na BTS e, posteriormente, enviados para a BSC. Representam, igualmente, o máximo dos máximos do *Round Trip Delay*. Aqui não existe distinção possível entre DL e UL, por isso, considera-se um valor global.

Tabela 3.2: Lista de contadores a considerar, [7].

Desempenho	**Tipo**	**Contador**
Packet Loss	*Artp*	ratioOfRtpPacketsLostUlMgw0
		ratioOfRtpPacketsLostUlMgw1
		ratioOfRtpPacketsLostUlMgw2
		ratioOfRtpPacketsLostUlMgw3
		ratioOfRtpPacketsLostDlMgw0
		ratioOfRtpPacketsLostDlMgw1
		ratioOfRtpPacketsLostDlMgw2
		ratioOfRtpPacketsLostDlMgw3
	Apacket	ratioOfPacketsLostUlApacketDB0
		ratioOfPacketsLostDlApacketDB0
Packet Delay Variation	*Artp*	2298/0: averageJitterUlMgw0 value averageJitterUlMgw0Max
		2298/1: averageJitterUlMgw1 value averageJitterUlMgw1Max
		2298/2: averageJitterUlMgw2 value averageJitterUlMgw2Max
		2298/3: averageJitterUlMgw3 value averageJitterUlMgw3Max
		2299/0: averageJitterDlMgw0 value averageJitterDlMgw0Max
		2299/1: averageJitterDlMgw1 value averageJitterDlMgw1Max
		2299/2: averageJitterDlMgw2 value averageJitterDlMgw2Max
		2299/3: averageJitterDlMgw3 value averageJitterDlMgw3Max
	Apacket	2303/0: jitterUlApacketIf value jitterUlApacketIfMax
		2307/0: jitterDlApacketIf value jitterDlApacketIfMax
One Way Delay	*Artp*	2300/0: rtdMgw0 value rtdMgw0Moy
		2300/1: rtdMgw1 value rtdMgw1Moy
		2300/2: rtdMgw2 value rtdMgw2Moy
		2300/3: rtdMgw3 value rtdMgw3Moy
	Apacket	2308/0: rtdBtsApacketIf value rtdBtsApacketIfMoy

CAPÍTULO 4

Desenvolvimento

Neste capítulo, é abordada toda a fase de desenvolvimento da aplicação de supervisão. São retratados todos os processos de instalação e programação, tais como, a instalação do OMC-R na máquina de desenvolvimento, a recolha de dados simulados, a implementação dos algoritmos de processamento dos dados dos ficheiros, a criação da interface gráfica e o algoritmo de gestão de alarmes. Ainda é tratada a fase final de desenvolvimento que consiste na criação de uma package de instalação.

4.1 Configuração do ambiente de desenvolvimento

Inicialmente, foi atribuída uma máquina de desenvolvimento para a concretização de todo o projeto. O sistema operativo desta máquina é o RHEL 7, que já foi apresentado no Capítulo 2.

4.1.1 Instalação do OMC-R

Com o objetivo de recolher valores de medição para a realização de testes unitários, é necessário instalar e configurar um OMC-R. Para isso, e com os acessos devidos, seguem-se as instruções para a instalação deste componente de monitorização.

Como é a primeira vez a instalar o OMC-R na máquina virtual, é necessário instalar o ambiente GSM OAM. Para isso, seguem-se as instruções:

- Clonar o ambiente e preparar tudo;
- Atualizar os ficheiros *shell startup* com a variável *PATH* e *LD_LIBRARY_PATH*;
- Adicionar a licensa TCL;
- Adicionar uma chave SSH para o acesso remoto sem ser preciso palavra passe;
- Garantir que o grupo "mera"é um dos grupos secundários.

Após estes fatores estarem cumpridos, e a instalação do ambiente for bem sucedida, é o momento de instalar o OMC-R. Para isso, seguem-se as devidas instruções:

- Clonar o repositório omc: Executar o comando "grc"e selecionar o projeto omc;
- Selecionar o repositório ao executar o comando "grs"(selecionar o repositório clonado);

- Escolher a *release branch* especifica: "git checkout -t remote/<branch>"
- Executar o comando *build*: "omc_build";
- Executar o comando *install*: "omc_install";
- Executar o comando *start*: "omc_start":
- Iniciar a interface gráfica: "mmi".

4.1.2 Instalação do *Maven*

Após a familiarização com o ambiente RHEL 7, foi efetuada a escolha da *framework* para a criação do projeto. A escolha do *Apache Maven* foi feita com base na facilidade do processo de compilação e da integração de bibliotecas que auxiliaram a criação do produto final.

A instalação desta *framework* passou por:

- Executar o *download* do ficheiro binário tar.gz do site "https://maven.apache.org/download.cgi"
- Extrair o conteúdo do ficheiro através do comando "tar xzvf apache-maven-3.9.1-bin.tar.gz"
- Adicionar o diretório bin à variável de ambiente *PATH* "setenv PATH ${PATH}:/opt/apache-maven-3.9.1/bin:."
- Verificar a instalação com o comando "mvn -v"numa nova janela

A versão mais recente do *Maven* carece da versão 8 do JDK. No entanto, a máquina de desenvolvimento utiliza a versão 6 do JDK para suportar o OMC-R e todas as ferramentas envolvidas. A solução encontrada para que a ferramenta seja desenvolvida com a versão 6, de forma a ser compatível com o OMC-R, foi definir a variável *JAVA_HOME* com a versão 8. No início de cada sessão, para executar o comando *build*, é necessário configurar esta variável através do comando "setenv JAVA_HOME /usr/lib/jvm/jre-1.8.0-openjdk-1.8.0.181-7.b13.el7.x86_64".

Para a primeira instrução *build and run* do projeto, é necessário executar os seguintes comandos dentro do diretório onde o ficheiro **pom.xml** se encontra:

- "mvn clean install"
- "mvn install dependency:copy-dependencies"
- "mvn package"
- "java -cp target/SupervisionTool-0.1.0.jar:target/dependency/ com.example.App"

Este conjunto de instruções é igualmente válido para quando se adiciona alguma dependência nova ao projeto. Para adicionar uma dependência, é importante ter em conta a versão compatível com a versão do compilador. Neste caso, o compilador está a usar a versão 6 do java, logo, todas as versões têm de ser compatíveis com esta versão.

4.2 Processamento de dados

Numa fase inicial, e para os testes unitários, os valores de medição são simulados e é usado um mecanismo de observção para os recolher. O mecanismo de observação do NEOS é uma

parte funcional que simula a recolha de dados de medições do BSC e o seu envio (dados) para o OMC-R.

O principal problema da realização da gestão da observação no simulador é a impossibilidade de recolher dados de medição reais a partir de *hardware* real em funcionamento. Mas, em vez disso, este mecanismo artificial é altamente configurável e permite a simulação de muitos casos em que o BSC real poderia provavelmente ter uma falha.

Para configurar o simulador NEOS, executaram-se as seguintes instruções:

- No *host* do NEOS é necessário configurar "/home_local/neos/ profile_NEOS_BCN<version>/ConfigurationFile"com os parâmetros (*OMC_IP_ADDRESS*, *EQP_ID*, *SOFTWARE*) de acordo com o OMC-R e a BSC a estudar;
- No *GUI* do OMC-R configurar o objeto *bscMdInterface* com o endereço IP do simulador NEOS;
- Iniciar o NEOS com o comando "/opt/nortel/NEOS_BCN<version>/bin/neos.x /home_local/neos/profile_NEOS_BCN<version>"
- No OMC-R: i) Desbloquear o objeto BSC; ii) *Build* do BDA no BSC.

De forma a que os valores de medição simulados sejam recebidos pelo OMC-R e, posteriormente, descodificados pela interface SDO, é necessário configurar o OMC-R. Para esse efeito, seguem-se as instruções que precisam de ser efetuadas:

- Confirmar se o serviço do OMC-R está ativo com o comando "omc_get";
- Se estiver ativo, iniciar a interface gráfica com o comando "mmi";
- Na barra de menus, selecionar *Security* e fazer *Login*;
- Selecionar a BSC e escolher *Build BDA*;
- Adicionar a *btsSiteManager* com o campo *mdScannerType* definido para *very fast statistic*;
- Criar um objeto *child* para a BSC;

Desta forma, o OMC-R vai começar a recolher os valores das medições, a cada 3 minutos, da entidade de simulação NEOS. O SDO encarrega-se de depositar os ficheiros no diretório padrão dedicado aos registos diários. Posteriormente, são comprimidos e armazenados, neste caso, durante vinte e um dias no diretório padrão para este efeito.

4.2.1 Descompressão dos ficheiros

O algoritmo para a descompressão dos ficheiros tar.gz está representado no Código 1 e faz uso de uma das dependências abordadas no Capítulo 2, designada por *Apache Commons-Compress*. O objetivo desta classe é descomprimir os ficheiros de forma a possibilitar o acesso futuro aos ficheiros que contêm os valores das medições de cada observação. Para isso, a lista de ficheiros do diretório padrão é iterada e só são descomprimidos os ficheiros de medições brutas. A ordem de descompressão começa com o ficheiro .zip, que serve como *input* da classe *GZIPInputStream*, seguindo para o ficheiro .tar, *input* da classe *TarArchiveInputStream*. A este objeto são aplicadas algumas condições de verificação e a partir da classe *IOUtils* é feita a cópia do conteúdo deste objeto para o destino.

```
File[] files = file.listFiles();
for (File f : files) {
    if (f.getName().contains("obs_raw_ovfs")) {
        try {
            TarArchiveInputStream tarInput = new TarArchiveInputStream(new GZIPInputStream
                                                (new FileInputStream(f)));
            TarArchiveEntry currentEntry = tarInput.getNextTarEntry();
            while (currentEntry != null) {
                File outFile = new File(file.getAbsolutePath() + File.separator
                                    + currentEntry.getName());
                if (currentEntry.isDirectory()) {
                    if (!outFile.exists()) {
                        outFile.mkdirs();
                    }
                } else {
                    File parentDir = outFile.getParentFile();
                    if (!parentDir.exists()) {
                        parentDir.mkdirs();
                    }
                    OutputStream fos = new FileOutputStream(outFile);
                    IOUtils.copy(tarInput, fos);
                    fos.close();

                }
                currentEntry = tarInput.getNextTarEntry();
            }
            tarInput.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Código 1: Implementação da descompressão dos ficheiros.

4.2.2 *Parsing* dos ficheiros

O objetivo desta classe é devolver uma estrutura de dados em que os valores de medição estão associados ao tipo de desempenho correto, associados à *btsSiteManager* correta e à hora e dia em que foram recolhidos. A primeira tarefa é a leitura de cada ficheiro OVFS presente no diretório padrão. A leitura é feita pela classe *FileReader* e a informação é guardada temporariamente num *buffer* criado pela classe *BufferedReader*. A leitura é feita linha a linha pois a informação contida nos ficheiros segue a estrutura presente na Figura 4.1. Ou seja, após a primeira nomenclatura *[COUNTER_LIST]*, o conteúdo da próxima linha é a lista de contadores dos quais foram retirados valores de medição e, após a nomenclatura *[COUNTER_VALUE]*, a próxima linha contém a informação da *btsSiteManager* monitorizada e os valores recolhidos. A primeira linha dos contadores é única mas a linha dos valores de medição pode repetir-se múltiplas vezes.

Para associar os valores de medição aos contadores, a linha que contém a lista dos contadores presentes no ficheiro é iterada bem como, a linha ou linhas que contêm os valores de medição. Esta informação é guardada em duas *ArrayLists* diferentes. Como à partida, o número de *btsSiteManager* presentes em cada ficheiro é desconhecido, existe outro objeto

```
[CLASS_BSM]
[COUNTER_LIST]
2294 0 CUM SITE,2294 1 CUM SITE,2294 2 CUM SITE,2294 3 CUM SITE,2295 0 CUM SITE,2295 1 CUM SITE,2295 2 CUM SITE,
[COUNTER_VALUE]
BSM 45/0,3,3,3,3,3,3,3,3,1000003,1000003,1000003,1000003,1000003,1000003,1000003,1,2,0.50,3,1,2,0.50,3,1
BSM 45/1,3,3,3,3,3,3,3,3,1000003,1000003,1000003,1000003,1000003,1000003,1000003,1,2,0.50,3,1,2,0.50,3,1
```

Figura 4.1: Formato geral de qualquer ficheiro OVFS.

ArrayList para guardar todas as ocorrências. Posteriormente, este objeto vai ser iterado para associar a cada uma das *btsSiteManagers*, um mapa com os valores de medição e os contadores. Este processo é apresentado no Código 2.

Após este processo, é associado a cada hora, extraida do nome do ficheiro, um conjunto de mapas das *btsSiteManagers* contidas nesse ficheiro. Isto garante que, a cada ficheiro lido, é devolvido um mapa com toda a informação organizada dessa hora. No fim, garante-se que esta estrutura de dados seja adicionada a um mapa final associado ao dia da recolha das amostras. O dia é retirado do diretório pai dos ficheiros OVFS. Este processo é apresentado no Código 3.

As duas funções auxiliares a este método têm como objetivo:

- Procurar pelos ficheiros OVFS no diretório padrão e retornar uma lista desses ficheiros;
- Filtrar o mapa dos valores das medições associados aos contadores para retirar aqueles que não interessam para a análise estatística.

```
buff = new BufferedReader(new FileReader(file));
String line = "";
while ((line = buff.readLine()) != null) {
    if (line.equals("[COUNTER_LIST]")){
        line = buff.readLine();
        String[] valores = line.split(",");
        for (int i = 0; i < valores.length; i++ ){
            countersList.add(valores[i]);
        }
    }
    if (line.equals("[COUNTER_VALUE]")) {
        line = buff.readLine();
        String[] valores = line.split(",");
        for (int i = 0; i < valores.length; i++ ) {
            if (valores[i].contains("BSM")) {
                bsmList.add(valores[i]);
            } else {
                countersValues.add(valores[i]);
            }
        }
        while ((line = buff.readLine()) != null) {
            valores = line.split(",");
            for (int i = 0; i < valores.length; i++ ){
                if (valores[i].contains("BSM")) {
                    bsmList.add(valores[i]);
                } else {
                    countersValues.add(valores[i]);
                }
            }
        }
    }
}
```

Código 2: Implementação da associação dos valores de medição aos contadores.

```
int bsmCount = bsmList.size();
int counterCount = countersList.size();
for (int i = 0; i < bsmCount; i++) {
    counterValueMap = new LinkedHashMap<String, Double>();

    for (int j = 0; j < counterCount; j++) {
        int index = i * counterCount + j;
        String counter = countersList.get(j);
        double value = Double.parseDouble(countersValues.get(index));
        if (value != -1) {
            counterValueMap.put(counter, value);
        }
    }
    counterValueMap = filterCounterValueMap(counterValueMap);
    String bsm = bsmList.get(i);
    mapBSMCounterValue.put(bsm, counterValueMap);
}
if (hourMap.containsKey(timeSample)) {
    List<Map<String, Map<String, Double>>> listBsm = hourMap.get(timeSample);
    listBsm.add(mapBSMCounterValue);
} else {
    List<Map<String, Map<String, Double>>> listBsm = new
                ArrayList<Map<String, Map<String, Double>>>();
    listBsm.add(mapBSMCounterValue);
    hourMap.put(timeSample, listBsm);
}
```

Código 3: Implementação da associação do mapa de contadores ás ***BTS Site Manager***s (BSMs) e a lista de BSMs a cada hora.

4.3 Interface gráfica

A interface gráfica foi desenvolvida com suporte da biblioteca *Java Swing*. Com o objetivo de responder aos requisitos levantados, desenvolveu-se uma interface intuitiva e simples, de modo, a facilitar o seu uso. Ainda foi integrada a dependência *JFreeChart* para a construção dos conjuntos de dados, personalização dos gráficos e da própria interface.

4.3.1 *Windows*

Nesta secção, abordam-se os diferentes ecrãs integrantes da aplicação, incluindo, o objetivo da criação de cada um deles bem como, os algoritmos que tratam os dados recolhidos para apresentar ao utilizador.

Splash Window

O propósito da criação desta janela é apenas informar o utilizador dos processos que ocorrem em *background*. É constituída por uma barra de progresso que representa o estado dos processos que estão a ocorrer. Estes processos constituem a descompressão dos ficheiros nos diretórios de armazenamento do SDO e seguem, por isso, o algoritmo já referido.

Ecrã de Configuração da Supervisão

O ecrã da configuração da supervisão representa o painel principal da aplicação porque todos os outros paineis integrantes da aplicação são executados a partir deste. Nomeadamente, os paineis que contêm a informação gráfica, o painel de configuração alarmística e um painel informativo que tem como função verificar a versão atual do sistema.

O objetivo para este painel é a configuração dos parâmetros de entrada para a supervisão desejada. Fazem parte deste painel as opções:

- Tipo de Desempenho;
- *BTS Site Manager*;
- Duração da supervisão;
- Data de inicio;
- Hora de inicio.

A escolha do tipo de desempenho é feita através de um *JComboBox*, isto é, um menu *pop up* que apresenta as três escolhas possíveis (*One Way Delay*, *Packet Delay Variation*, *Packet Loss*). A escolha da BSM é feita através de um *JTextField*, ou seja, aceita uma entrada de texto pelo utilizador. A escolha da duração da supervisão é feita também por um menu *pop up* que apresenta as quatro opções disponíveis ("15 minutos", "1 hora", "24 horas"e "1 semana"). Para a data de inicio, também é apresentado um menu *pop up* que apresenta os dias disponíveis para efetuar as observações. Este é um processo em que, antes de o ecrã ficar visivel, é feita uma pesquisa pelos diretórios de forma a retirar as datas disponíveis. O objetivo é garantir que o utilizador nunca coloque dias em que não existam valores de medição. Assim, os dias já são apresentados quando o ecrã aparece. Por fim, no *JTextField* que representa a hora de inicio é feita uma pesquisa, igualmente antes do ecrã ficar visivel, para extrair a primeira amostra de tempo disponivel consoante a data escolhida.

Ecrã de Visualização dos Gráficos

Existe uma classe que recolhe os valores dos parâmetros de entrada do ecrã de configuração da supervisão e cria um objeto com esses atributos. Essa classe denomina-se por *SupervisionChoices.java*. Este objeto é passado como argumento para o próximo ecrã, o que possibilita o acesso a estes atributos para retornar a informação desejada. De acordo com o tipo de desempenho escolhido, um de três ecrãs aparece representados pelas classes *OneWayDelayFrame.java*, *PacketDelayVariationFrame.java* e *PacketLossFrame.java*. Sendo que a informação que integra cada conjunto de dados para gerar cada gráfico é processada por tipo de desempenho, é mais fácil separar o código por três classes diferentes. O processo de *file parsing* é, então, posto em prática e é criada a estrutura de dados com os valores de medição já tratados.

A função que cria o conjunto de dados para cada gráfico baseia-se na estrutura de dados que resulta do *file parsing*. Existem quatro algoritmos diferentes para cada duração de supervisão.

O Código 4 representa o algoritmo construído para a duração de "15 minutos"e "1 hora". Nesta função, são passados como argumentos o objeto com os atributos dos parâmetros de entrada, a estrutura de dados que contém os valores das medições, o tipo de desempenho, o conjunto de dados para construir os gráficos e a duração, neste caso, toma o valor de 15 ou 60 para o período de 15 minutos e 1 hora, respetivamente. Primeiramente, consoante a duração passada como argumento, é calculada a amostra temporal limite somando este valor à amostra temporal inserida pelo operador. É construído um objeto do tipo Intervalo com estes dois valores, de modo a criar um limite temporal. Segue-se a iteração sobre cada entrada desse mapa que representa os valores de medição associados a cada contador e é verificado se cada uma delas pertence ao intervalo. Em caso afirmativo, é retirado o mapa que corresponde à amostra temporal inserida naquele intervalo. O mapa de valores de medição vai ser iterado através da filtragem pela BSM e pelo tipo de desempenho pretendidos. Então, cada um dos valores recolhido é dividido por mil, para converter de microsegundos para milisegundos e é adicionado ao conjunto de dados através da instrução *addValue()*, juntamente com a hora de recolha.

Para a duração de "24 horas", o algoritmo é apresentado no Código 5 onde é feita uma iteração sobre cada intervalo de horas que integra a lista *listaIntervalos*. Esta lista tem os intervalos temporais das horas de um dia inteiro, representados por tuplos, na forma "(HH:mm, HH:mm)", em que HH representa as horas e mm os minutos. A classe interna *Intervalo* constrói os intervalos temporais para integrar nesta lista, por exemplo, um dos tuplos presentes na lista é "(0000,100)". Segue-se a iteração sobre a *listaIntervalos* e, para cada intervalo, verifica-se se cada chave do mapa correspondente ao dia inserido pelo utilizador pertence a esse intervalo de horas. Caso pertença, a soma dos valores das medições é feita para todas as amostras que se inserem nesse período temporal. Por fim, é feita a média dos valores pelo número de amostras e o valor resultante vai representar cada intervalo de horas. No fim, os valores das médias são adicionados a um *ArrayList* e, de seguida, quando os períodos temporais terminarem, os valores dessa lista são adicionados ao conjunto de dados que vai gerar o gráfico.

No Código 6, está representado o algoritmo caso a escolha do utilizador seja o período de

```
public void computeValues(SupervisionChoices superChoices, Map<String, Map<String,
    List<Map<String, Map<String, Double>>>>> mapFinal, String performance,
    DefaultCategoryDataset dataset, int duration) throws ParseException {
    if (mapFinal.get(superChoices.getStartDate()).containsKey(superChoices.getStartTime())) {
        String timeSample = superChoices.getStartTime();
        DateFormat sdf = new SimpleDateFormat("HHmm");
        Date initialDate = sdf.parse(timeSample);
        Calendar cal = Calendar.getInstance();
        cal.setTime(initialDate);
        cal.add(Calendar.MINUTE, duration);
        DecimalFormat mFormat= new DecimalFormat("00");
        String newTimeSample = mFormat.format(cal.get(Calendar.HOUR_OF_DAY)) +
                               mFormat.format(cal.get(Calendar.MINUTE));
        Intervalo intervalo = new Intervalo(timeSample, newTimeSample);
        for (String key : mapFinal.get(superChoices.getStartDate()).keySet()) {
            if (intervalo.contains(key)){
                List<Map<String, Map<String, Double>>> bsmListForHour =
                                        mapFinal.get(superChoices.getStartDate()).get(key);
                for (Map<String, Map<String, Double>> counterValueMap : bsmListForHour) {
                    if (counterValueMap.containsKey(superChoices.getBSM())) {
                        if (counterValueMap.get(superChoices.getBSM()).containsKey(performance)) {
                            Date date = sdf.parse(key);
                            DateFormat newSdf = new SimpleDateFormat("HH:mm");
                            Double value = counterValueMap.get(superChoices.getBSM()).get(performance);
                            value /= 1000;
                            if (!Double.isNaN(value)) {
                                dataset.addValue((Number) value, "Values", newSdf.format(date));
                            }
                        }
                    }
                }
            }
        }
    }
}
```

Código 4: Implementação do algoritmo para a construção dos gráficos com duração de supervisão de "15 minutos"e "1 hora".

duração de "1 semana". É aplicado o algoritmo da duração de "24 horas"mas sem intervalos de horas, ou seja, a média que é calculada é em relação ás amostras recolhidas em cada hora. Para isso, o mapa de valores correspondente ao dia inserido pelo utilizador, vai ser iterado. Assim, a média é efetuada com todos os valores recolhidos ao longo desse dia. No fim, o dia que consta nos atributos do objeto, é convertido para o tipo *Date*, define-se uma estrutura *Calendar* e é adicionado "1"dia a este valor. O valor resultante vai ser a próxima entrada no mapa final para retirar mais medições. Este algoritmo repete-se sete vezes e o valor gerado corresponde á média de valores recolhidos durante um dia. São alocados numa *ArrayList* que, posteriormente, deposita os valores no conjunto de dados para gerar os gráficos.

Após os conjuntos de dados estarem criados, são passados como argumento à função que cria os gráficos. Esta função tem como objetivo retornar o gráfico construído a partir da configuração de certos parâmetros. Primeiramente, o tipo de gráfico a criar é gráfico de linha. Os parâmetros de configuração do gráfico incluem, o título, a legenda dos eixos x e y, o conjunto de dados, a orientação e três condições *boolean*. A partir do gráfico gerado, começa-se

```
/*Instanciação da listaIntervalos, cumValues e hours*/
for (Intervalo intervalo : listaIntervalos)
{
    Double value = 0.0;
    Double mean = 0.0;
    int count = 0;
    for (String key : mapFinal.get(superChoices.getStartDate()).keySet())
    {
        if (intervalo.contains(key))
        {
            List<Map<String, Map<String, Double>>> bsmListForHour =
                      mapFinal.get(superChoices.getStartDate()).get(key);
            for (Map<String, Map<String, Double>> counterValueMap : bsmListForHour)
            {
                if (counterValueMap.containsKey(superChoices.getBSM()))
                {
                    if (counterValueMap.get(superChoices.getBSM()).containsKey(performance))
                    {
                        value = counterValueMap.get(superChoices.getBSM()).get(performance);
                        value += value/1000;
                        count++;
                    }
                }
            }
        }
    }
    mean = value/count;
    cumValues.add(mean);
}
for (int i = 0; i < hours.size(); i++) {
    dataset.addValue(cumValues.get(i), "Mean Values", hours.get(i));
}
```

Código 5: Implementação do algoritmo para a construção dos gráficos com duração de supervisão de "24 horas".

por construir a interface gráfica. Cria-se um *ChartPanel* que se insere noutro *JPanel*. Ainda é adicionado a cada painel de cada gráfico, um botão para exportar os valores e a imagem correspondente num documento Excel. Os vários painéis ficam alinhados em grelha segundo um *layout*, denominado por *GridLayout*.

Assim, para cada tipo de performance, aparece um ecrã com os gráficos construídos segundo os algoritmos descritos. Caso não exista tráfego numa das interfaces monitorizadas, o gráfico não é gerado e o ecrã devolve uma mensagem de texto a informar o utilizador.

```
int keyNumber = 7;
int counter = 0;
String startingDate = superChoices.getStartDate();
List<Double> cumValues = new ArrayList<Double>();
List<String> daysWeek = new ArrayList<String>();
daysWeek.add(startingDate);
while (counter < keyNumber) {
    SimpleDateFormat sdf = new SimpleDateFormat("yyyyMMdd");
    Date date = sdf.parse(startingDate);
    Calendar calendar = Calendar.getInstance();
    calendar.setTime(date);
    calendar.add(Calendar.DAY_OF_MONTH, 1);
    date = calendar.getTime();
    startingDate = sdf.format(date);
    daysWeek.add(startingDate);
    counter++;
}

for (int j = 0 ; j < daysWeek.size(); j++) {
    Double value = 0.0;
    Double mean = 0.0;
    int count = 0;
    Map<String, Map<String, List<Map<String, Map<String, Double>>>>> map =
                    FileParsing.getMapFinal(daysWeek.get(j));
    if (map.containsKey(daysWeek.get(j))) {
        for (String key : map.get(daysWeek.get(j)).keySet()) {
            List<Map<String, Map<String, Double>>> bsmListForHour =
                    map.get(daysWeek.get(j)).get(key);
            for (Map<String, Map<String, Double>> counterValueMap : bsmListForHour) {
                if (counterValueMap.containsKey(superChoices.getBSM())) {
                    if (counterValueMap.get(superChoices.getBSM()).containsKey(performance)) {
                        value = counterValueMap.get(superChoices.getBSM()).get(performance);
                        value += value/1000;
                        count++;
                    }
                }
            }
        }
        mean = value/count;
        cumValues.add(mean);
    }
}

for (int i=0; i<cumValues.size(); i++) {
    if (!Double.isNaN(cumValues.get(i))) {
        dataset.addValue((Number) cumValues.get(i), "Mean Values", daysWeek.get(i));
    }
}
```

Código 6: Implementação do algoritmo para a construção dos gráficos com duração de supervisão de "1 semana".

Ecrã de Configuração Alarmística

Outra das funcionalidades esperadas, é a criação de dois grandes ficheiros com informação alarmística por períodos de monitorização com duração de uma semana. Para aceder ao painel de configuração alarmistica, basta clicar na barra de menu na opção *Alarms* e é apresentado ao utilizador um novo ecrã com duas opções:

- O número de *samples* a analisar, ou seja, quantos valores de medição é necessário avaliar caso esses valores se encontrem acima de um *threshold* pré-definido;
- Uma *CheckBox* para ativar ou desativar este processo.

Este é um processo automático que é lançado em *background*, ou seja, é ativado assim que a aplicação é instalada. É lançada uma instrução para ser incluída no conjunto de tarefas da *crontab*. Esta ferramenta permite aos utilizadores programar a execução automática de comandos, *scripts* ou programas em intervalos de tempo regulares, seja diariamente, semanalmente ou em dias e horários especificos. As tarefas agendadas são executadas em segundo plano, sem a necessidade de intervenção manual. A estrutura da *crontab* consiste em campos separados por espaços que representam o horário e a frequência de execução da tarefa. Os valores são personalizáveis e incluem os minutos, as horas, os dias do mês, os meses e os dias da semana. Assim, esta ferramenta permite que o mecanismo de gestão alarmistica se torne confiável e correto.

A gestão dos alarmes consiste em processar todos os valores de medição das interfaces monitorizadas existentes na rede nos últimos 7 dias, com base em dois critérios: *Packet Delay Variation* e *Packet Loss*. Após a recolha destes valores, é necessário identificar se é atingido um limiar de risco para o serviço. Este cálculo não conduzirá a qualquer armazenamento no disco. Haverá um alarme interno por BTS e por critério. O fluxograma do algoritmo proposto para a gestão alarmistica, encontra-se presente na Figura 4.2.

O primeiro processo é o *parsing* dos ficheiros do dia anterior. O resultado é uma estrutura de dados com os valores de medição recolhidos. O próximo processo é construir uma lista com todas as BSMs que foram monitorizadas e extrai-las para uma lista. Isto permite que os restantes processos sejam efetuados ao longo da iteração sobre os elementos desta lista. A seguir, consoante as decisões, é criado um diretório para cada BSM e dois ficheiros, um para cada critério. Verifica se o dia ainda se encontra no limite de uma semana, e arranca o processo de gerar os alarmes. Para cada critério, os valores são comparados com um limiar de risco e, caso o valor de medição ultrapasse este limiar, é lançado um alarme. O alarme fica registado no ficheiro na forma:

- *BTS 12345 Bad performances on interface type/way – Risk for GSM-R service*, onde é identificada a BSM em que ocorreu, o tipo de desempenho e se aconteceu em *downlink* ou *uplink*.

Enquanto as amostras de medição corresponderem a um dia dentro do limite de duração de uma semana, o conteúdo é reescrito nos mesmos ficheiros. Logo após o término dos sete dias, é alterado o nome dos ficheiros que contêm os alarmes, de forma, a identificar o período correspondente à supervisão. E, são criados novos ficheiros de escrita dos próximos alarmes.

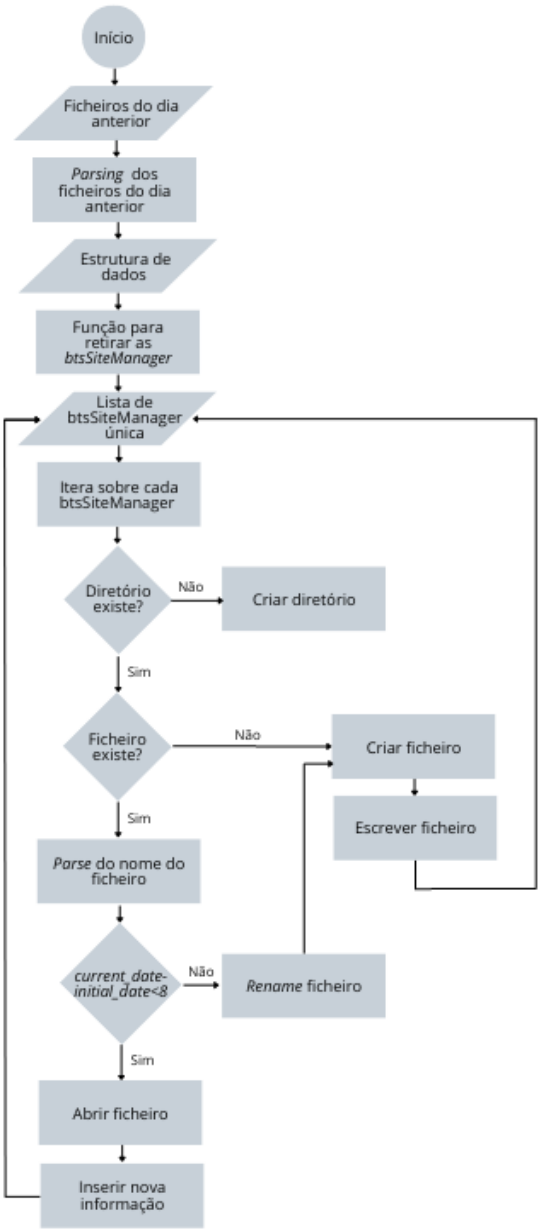

Figura 4.2: Fluxograma do algoritmo alarmistico.

4.4 Criação de uma *package*

A criação de uma *package* de instalação é feita pela ferramenta de gestão RPM. Esta ferramenta é importante quando se trata de partilhar *software* entre máquinas com distribuições Linux baseadas no sistema *Red Hat*, como, o RHEL.

Para a criação do pacote RPM, é necessário construir um *script* de controlo designado por ficheiro SPEC. Neste caso, o ficheiro SPEC é do tipo binário constituído apenas pelos binários compilados a partir do código fonte e pelos *patches*. Os itens usados no preâmbulo são o nome do pacote, a versão, a *release*, um sumário que descreve o pacote, a licença, a

arquitetura da máquina, os requisitos de instalação e o *buildroot*. Encontra-se representado no Código 7 o preâmbulo usado para a construção do ficheiro SPEC.

```
Name:   supervision-tool
Version:    00.01
Release:    1
Summary:    Supervision Tool to supervise quality of user-plane links at OMC-R
License:    Proprietary
BuildArch:  x86_64
BuildRoot:  %{name}-%{version}-%{release}
Requires:   java >= 1.8
```

Código 7: Estrutura do Preâmbulo do ficheiro SPEC.

As secções constituintes do corpo do ficheiro SPEC são *%description*, *%build*, *%install* e *%files*. Na secção *%description*, é usada a macro que aponta ao sumário presente no preâmbulo do ficheiro. Na secção *%build*, retratada no Código 8, é passada a instrução que compila o projeto e constrói um ficheiro ".jar"que contém, por sua vez, os ficheiros de classe Java com o *bytecode* compilado. Contém ainda, os recursos necessários ao funcionamento da aplicação, neste caso, ficheiros de imagens. Desta forma, os recursos e as classes ficam organizadas de forma eficiente e modular, o que facilita a partilha deste *software*. É definida uma variável com o caminho do diretório final onde vai ser instalada a package.

```
%build

echo "------------------------------------------------------------"
echo "Creating application´s .jar."
echo "------------------------------------------------------------"

%define ToolRoot /opt/kontron/SupervisionTool

mvn -f %{project_dir}/pom.xml package -X -Dinstall.dir=%{ToolRoot}
```

Código 8: Estrutura da secção *%build* do ficheiro SPEC.

Para que o processo de compilação seja bem sucedido, a instrução "mvn package"necessita de um ficheiro XML que descreva todas as informações para construir e gerir um projeto Maven. Este ficheiro intitula-se por "pom.xml"e contém as configurações e metadados associados à ferramenta. Os aspetos principais de um ficheiro POM (*Project Object Model*) são:

- Coordenadas do Projeto.
- Dependências, ou seja, as bibliotecas externas que o projeto precisa para compilar e funcionar corretamente.
- *Plugins* do Maven, ou seja, extensões que permitem a execução de várias tarefas durante as diferentes fases do projeto como a compilação, o empacotamento, entre outras.
- Configurações do *build*, ou seja, informações como o projeto deve ser construído a partir de configurações de compilação, recursos e filtros.
- Configurações de Repositório.
- Outras configurações, nomeadamente, personalização de propriedades, diretórios de saída, entre outras.

Neste caso, todas estes aspetos foram tidos em consideração nesta fase de compilação do projeto. As propriedades do porjeto Maven foram personalizadas com o propósito de alterar o compilador para a versão 1.6 do Java e o diretório de instalação ser o diretório *target*. As dependências apresentadas no Capítulo 2, foram inseridas no projeto e encontram-se representadas no Código 9. As versões foram tidas em conta para que estivessem de acordo com a versão 6 do Java. Algumas apresentam certas vulnerabilidades mas não afetam o *overall* do funcionamento da ferramenta.

```
<dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.11</version>
      <scope>test</scope>
    </dependency>
    <dependency>
      <groupId>org.apache.commons</groupId>
      <artifactId>commons-compress</artifactId>
      <version>1.12</version>
    </dependency>
    <dependency>
      <groupId>jfree</groupId>
      <artifactId>jfreechart</artifactId>
      <version>1.0.13</version>
    </dependency>
    <dependency>
      <groupId>org.apache.poi</groupId>
      <artifactId>poi</artifactId>
      <version>3.17</version>
    </dependency>
    <dependency>
      <groupId>org.apache.poi</groupId>
      <artifactId>poi-ooxml</artifactId>
      <version>3.17</version>
    </dependency>
</dependencies>
```

Código 9: Bloco das dependências no ficheiro POM.

Foram também adicionados novos plugins com o objetivo de executar algumas tarefas durante o processo de compilação. Nomeadamente, copiar os recursos de imagens que a aplicação necessita para o correto funcionamento de forma a evitar o uso de *paths* absolutas. Copiar também o *script* "start.sh"para o diretório *target* aplicando um filtro. Este filtro consiste em substituir uma variável durante o processo de compilação. As dependências também foram copiadas para um novo diretório dentro do *target*.

Na secção *%install*, é desenhada a árvore de diretórios e ficheiros, presente na Figura 4.3, que será criada após a instalação da package da ferramenta. Nesta secção estão presentes as instruções de instalação do software para a ferramenta *rpmbuild* no diretório *BUILDROOT*. Este diretório é um diretório base *chroot* vazio, que se assemelha ao diretório raiz do utilizador final. Aqui criam-se os diretórios que irão conter os ficheiros instalados. Os diretórios são criados através das instruções "mkdir"e os ficheiros são copiados através das instruções "install"e "cp".

```
%install

echo "-------------------------------------------------------"
echo "Creating directories tree and copying files"
echo "-------------------------------------------------------"

mkdir -p %{buildroot}%{ToolRoot}/etc/profile.d
cp %{project_dir}/src/main/etc/supervisionTool.sh %{buildroot}%{ToolRoot}/etc/profile.d
cp %{project_dir}/src/main/etc/supervisionTool.csh %{buildroot}%{ToolRoot}/etc/profile.d

mkdir -p %{buildroot}/etc/profile.d

install -D -m 644 -t %{buildroot}/etc/profile.d %{project_dir}/src/main/etc/supervisionTool.sh
install -D -m 644 -t %{buildroot}/etc/profile.d %{project_dir}/src/main/etc/supervisionTool.csh

mkdir -p %{buildroot}/etc/cron.d
install -D -m 644 -t %{buildroot}/etc/cron.d %{project_dir}/src/main/etc/Ktr-SpvTool

mkdir -p %{buildroot}%{ToolRoot}/etc/cron.d
cp %{project_dir}/src/main/etc/Ktr-SpvTool  %{buildroot}%{ToolRoot}/etc/cron.d

mkdir -p %{buildroot}%{ToolRoot}/lib
cp %{project_dir}/target/lib/*.jar %{buildroot}%{ToolRoot}/lib

mkdir -p %{buildroot}%{ToolRoot}/bin
cp %{project_dir}/target/bin/SupervisionTool-0.1.0.jar %{buildroot}%{ToolRoot}/bin
cp %{project_dir}/target/bin/start.sh %{buildroot}%{ToolRoot}/bin
cp %{project_dir}/reportAlarm.sh %{buildroot}%{ToolRoot}/bin

mkdir -p %{buildroot}%{ToolRoot}/config
cp %{project_dir}/config/ReportConfig.cfg %{buildroot}%{ToolRoot}/config
```

Código 10: Estrutura da secção *%install* do ficheiro SPEC.

O propósito da secção *%files*, retratada no Código 11, é complementar a secção anterior especificando a lista de ficheiros fornecidos pelo RPM e a localização completa do caminho no sistema do utilizador final. São definidas também as permissões de escrita ou leitura destes ficheiros consoante o utilizador final. Neste caso, a ferramenta permanecerá funcional para o utilizador *root*.

```
%files
%dir %{ToolRoot}

%{ToolRoot}/etc/*
%{ToolRoot}/config/ReportConfig.cfg

%defattr(755, root, root, 755)
%{ToolRoot}/bin/SupervisionTool-0.1.0.jar
%{ToolRoot}/bin/start.sh
%{ToolRoot}/bin/reportAlarm.sh
%{ToolRoot}/lib/*

%attr(644, root, root) /etc/profile.d/supervisionTool.sh
%attr(644, root, root) /etc/profile.d/supervisionTool.csh
%attr(644, root, root) /etc/cron.d/Ktr-SpvTool
```

Código 11: Estrutura da secção *%files* do ficheiro SPEC.

O ficheiro SPEC é executado através de um *script* para facilitar este processo e através, mais uma vez, do uso de macros para evitar *paths* absolutas. Usando a ferramenta *rpmbuild* constrói-se um pacote RPM com uma extensão .rpm. Este será o pacote de instalação da aplicação noutra máquina tanto numa vertente de desenvolvimento como na vertente mais nominal, ligada ao cliente.

4.5 Guia de utilização

Nesta secção é apresentado o guia de utilização para o correto funcionamento da ferramenta.

4.5.1 Instalação

A aplicação foi desenvolvida para dar suporte à entidade de gestão e manutenção da rede GSM-R. Isto significa que, na próxima versão lançada deste componente, a package construída irá junto com as outras ferramentas de forma a ser instalada na máquina destino. O ambiente de execução, em principio, estará configurado para que a instalação ocorra sem problemas secundários.

Primeiramente, é necessário executar o comando "yum install <Pacote .rpm>"como utilizador *root* para instalar o pacote correspondente á aplicação. O *Yellowdog Updater, Modified* (YUM) verifica se as dependências do pacote existem, caso contrário, esta instrução é capaz de as instalar. O serviço de gestão de pacotes YUM tem como objetivo instalar a package RPM noutra máquina. O YUM é uma ferramenta que simplifica a instalação, atualização e desinstalação de software em sistemas Linux.

Para que tudo corra bem, a estrutura esperada dos diretórios e respetivos ficheiros na máquina destino é a que está apresentada na Figura 4.3. No momento, o uitlizador *root* fica responsável pelo diretório da ferramenta com as devidas permissões.

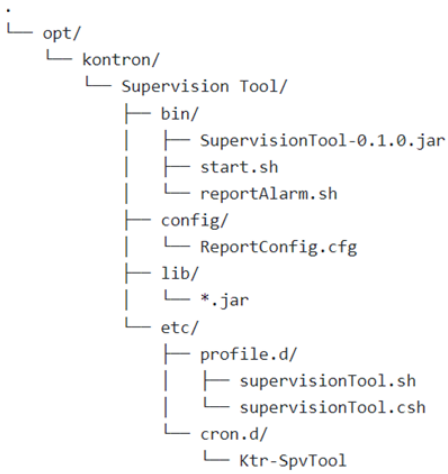

Figura 4.3: Estrutura dos diretórios e ficheiros na máquina destino.

Nesta árvore, existem quatro diretórios principais. No diretório "/bin", encontra-se o ficheiro ".jar"que contém as classes e os recursos utilizados pela aplicação. É a partir daí que a aplicação é iniciada. Os *scripts* que também se encontram no diretório "/bin"são executados pela

ferramenta. O utilizador comum não tem permissões para os alterar, tem apenas permissões de leitura. O *script* "start.sh"é o *script* de inicialização da ferramenta. Copia os ficheiros recolhidos e processados pela interface SDO para o diretório "/tmp/compressed_extended". Descomprime os ficheiros e inicia a aplicação. No final, após o operador fechar a janela principal, limpa o diretório temporário onde os ficheiros estão armazenados.

O script "ReportAlarm.sh"é executado por uma instrução presente no *crontab* da máquina. A sua principal função é reportar situações que ultrapassem um limiar de risco através de alarmes. O seu período de observação é de uma semana de atividade e são analisados dois tipos de desempenho da rede: Variação de Atraso de Pacotes e Perda de Pacotes.

No diretório "/config", existe o ficheiro "ReportConfig.cfg"que configura a gestão de alarmes. É possível ativar ou desativar esta funcionalidade e escolher o número de amostras para ativar um dos dois alarmes.

No diretório "/lib", encontram-se todas as dependências necessárias para executar a aplicação.

O diretório "/etc"contém dois sub-diretórios que incluem os ficheiros para definir o caminho para a ferramenta a colocar no *PATH* e a instrução a enviar para o *crontab*.

4.5.2 Arranque

De seguida, e após a entrada no diretório "/bin", executa-se o *script* "start.sh"com o comando "./start.sh". As tarefas de execução deste *script* são:

- A criação do diretório /compressed_extended no /tmp;
- A sincronização dos ficheiros OVFS depositados no SDO para este novo diretório;
- A descompressão dos ficheiros com os valores de medição brutos;
- A inicialização da aplicação;
- A remoção dos ficheiros e do diretório criado após o término da aplicação.

Após os processos de criação do diretório e a sincronização e descompressão dos ficheiros, a interface gráfica é apresentada. É apresentado ao utilizador um painel *splash*, que contém a barra de progresso. Assim que esta ficar preeenchida, é apresentado o painel de configuração da supervisão.

A partir deste painel, podem ser realizadas as seguintes funções:

- Verificação da versão da ferramenta, acessível por "*Check Version*"presente no menu "*About*";
- Configuração e visualização de uma supervisão, através do correto preenchimento dos campos presentes no ecrã da configuração da supervisão;
- Configuração do relatório semanal de alarmes, acessível por "*Report Configuration*"presente no menu "*Alarms*".

4.5.3 Menu Alarmes

Configuração do Relatório

O relatório semanal de alarmes é configurado através de uma opção que solicita ao utilizador que inicie esta tarefa. Na barra de menus, apenas disponível no painel de configuração da

supervisão, existe uma opção denominada "*Report Configuration*"dentro da opção principal "*Alarms*". O painel que é apresentado ao utilizador contém dois parâmetros de entrada: uma *CheckBox* com o objetivo de ativar ou desativar este processo e um campo para introduzir o número de amostras personalizável para que o algoritmo analise. Depois de editar estas opções, basta clicar no botão intitulado por "*Apply Changes*"para confirmar as alterações e regressar à página de configuração. Ou, se as opções já estiverem definidas, basta clicar no botão intitulado por "*Cancel*"para regressar à janela de configuração.

4.5.4 Menu Sobre

Verificação da Versão da Ferramenta

Para verificar a versão da ferramenta, a opção que leva o utilizador a realizar esta tarefa encontra-se na barra de menus disponível apenas no painel de configuração da supervisão. Neste caso, a opção intitulado por "*Check Version*"encontra-se dentro da opção principal "*About*". O painel apresentado contém informações sobre a aplicação e, para regressar ao painel de configuração, existe um botão com a indicação "OK".

4.5.5 Supervisão

Para a configuração e visualização de uma supervisão, o utilizador que se encontra no painel de configuração tem de preencher os campos de configuração com os dados corretos, no formato correto, para poder avançar com o processo.

No campo "*Performances*", escolher entre as três opções disponíveis, tais como, "*Packet Delay Variation*", "*One Way Delay*"e "*Packet Loss*". O campo "*BTS Site Manager*"é corretamente preenchido por "*BSM XXX/YYY*". O XXX é o número de BSC e o YYY o número de BTS. Esta informação pode ser encontrada nos ficheiros OVFS, na secção *[COUNTER_VALUES]*, no início de cada linha de valores estatísticos. Em "*Supervision Duration*", existe um outro menu pendente com 4 opções possíveis: 15 minutos, 1 hora, 24 horas e 1 semana. O campo "*Starting Date*"contém todas as datas disponíveis no diretório temporário. A ferramenta só funciona com as datas fornecidas pelo OMC-R. O campo "*Starting time*"deve ser preenchido no formato "HH:mm", em que HH representa as horas e mm os minutos.

Após o preenchimento dos campos de configuração, o passo seguinte é clicar no botão intitulado "*Supervise*"para iniciar a recolha dos valores de medição e apresentá-los no painel seguinte. Existe um processo que precede este para verificar se os campos preenchidos contêm valores fiáveis e num formato aceite pela ferramenta. Após a conclusão do processamento em segundo plano, é apresentado o painel que contém os gráficos construídos a partir do conjunto de valores de medição. É possível analisar os gráficos e tirar conclusões sobre o comportamento de cada interface. Existe ainda um botão para exportar a informação contida em cada gráfico. Os gráficos permitem ainda a interação do utilizador com a informação, ou seja, é possível fazer *zoom in* e *zoom out* nos gráficos.

Caso o utilizador pretenda efetuar uma nova supervisão, existe um botão com a designação "*New Supervision*"no canto superior direito do painel para regressar à página de configuração dos parâmetros de supervisão. E basta seguir os passos anteriores para uma nova supervisão.

4.5.6 Sair da aplicação

Em qualquer altura, se o utilizador premir o botão de saída da aplicação, é apresentada uma mensagem de confirmação se a intenção é realmente sair da aplicação ou regressar à janela de configuração de supervisão.

4.5.7 Desinstalação da aplicação

Para remover o pacote da máquina, é necessário executar o comando "*yum remove <Package .rpm>*"como *root*. Todos os ficheiros devem ser removidos do diretório de destino automaticamente após a execução deste comando.

CAPÍTULO 5

Validação

Neste capítulo, são apresentados os detalhes da validação, testes e os resultados obtidos durante a implementação da aplicação. A validação desempenha um papel fundamental para garantir a qualidade, confiabilidade e adequação da aplicação aos requisitos estabelecidos. O objetivo foi alcançado ao projetar testes para avaliar a precisão dos cálculos, a eficácia das funções executadas, a usabilidade, o desempenho e a integridade dos dados. Os resultados obtidos dessas avaliações vão ser explicados, incluindo detalhes sobre a eficiência das funcionalidades e o *feedback* dos utilizadores. Através da análise detalhada desses testes e resultados, é possível obter informações fidedignas sobre o desempenho e a eficácia da aplicação, o que pode encaminhar as decisões sobre o ajuste e melhorias a ter em conta para obter melhores resultados.

5.1 Resposta aos requisitos funcionais

5.1.1 Resultados do Processamento de Dados

Nesta secção, são analisados os resultados obtidos dos algoritmos de pós processamento dos valores de medição. No Capítulo 3, foram levantados alguns requisitos funcionais em relação, neste caso, ao processamento de dados. A partir da análise técnica realizada aos ficheiros OVFS, observou-se que a informação contida seguia uma ordem e um formato especifico. A cada nova entrada, ou linha, do ficheiro, é apresentada nova informação, seja uma lista com os contadores ou uma lista com os valores de medição de cada *btsSiteManager*. Desta forma, o algoritmo construído está preparado para retirar os dados neste tipo de formato. Um dos entraves ao desempenho dos algoritmos é o elevado número de ficheiros e, por conseguinte, um grande volume de dados a processar. Neste caso, a solução para reduzir o tempo de processamento foi a utilização consciente de ciclos e estruturas de dados eficientes no código fonte, de modo, a reduzir a complexidade computacional. Também foram seguidas boas práticas de programação, como por exemplo, a prática consistente de fechar todos os blocos *if* com as suas respetivas cláusulas *else*, garantindo um fluxo lógico. E na gestão de *buffers* e ficheiros onde todos os *buffers* foram libertados, de modo, a assegurar que a memória

alocada não é mais usada. Sendo que, todos os ficheiros foram fechados utilizando os métodos apropriados.

Já em relação à garantia de precisão e integridade dos cálculos efetuados, as fórmulas a serem usadas são compostas por cálculos simples permitindo, assim, que não se torne um processo demorado. Para acompanhar a integridade dos dados, foi feito um controlo para identificar possíveis erros através de mecanismos de *logging*, ou registo de eventos. Este processo consiste em registar eventos durante a execução do programa, como, mensagens de erros, exceções e valores de variáveis. O registo de eventos foi direcionado para a consola ou para ficheiros de *log* com o objetivo de compreender o estado do sistema em determinado momento e identificar possíveis erros. Ao longo do desenvolvimento do código fonte, foi sempre tido em conta o suporte a escalabilidade futura. Para isso, o algoritmo mantém-se imparcial caso as variáveis definidas sofram alterações.

5.1.2 Resultado Visual

Nesta seção, apresenta-se o resultado visual da interface gráfica criada para analisar o desempenho das interfaces monitorizadas. A aplicação oferece uma interface gráfica intuitiva e *user-friendly* ao utilizador, de fácil utilização e interação. Os gráficos gerados são informativos e atendem ao objetivo original, fornecendo as informações desejadas.

Além disso, a função de exportação dos valores de medição e da imagem do gráfico foi implementada com sucesso. A ferramenta gera um ficheiro Excel com duas folhas de cálculo: uma contendo os dados inseridos numa tabela e outra com a imagem do gráfico.

Splash Window

Na Figura 5.1, é apresentada a janela de *Splash* com que é iniciada a aplicação. O propósito ja foi explicado no capítulo anterior.

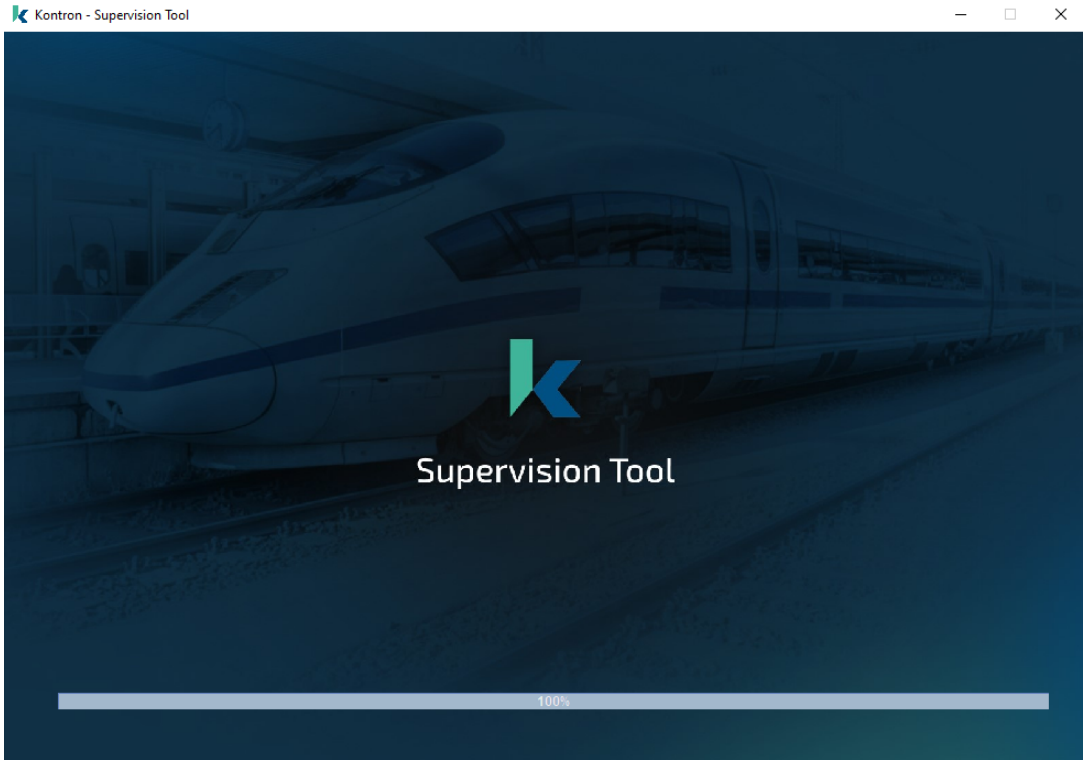

Figura 5.1: *Splash Window.*

Ecrã de Configuração da Supervisão

Na Figura 5.2, é apresentado o ecrã de configuração da supervisão. Cabe ao utilizador personalizar os parâmetros de entrada de acordo com a monitorização que deseja efetuar.

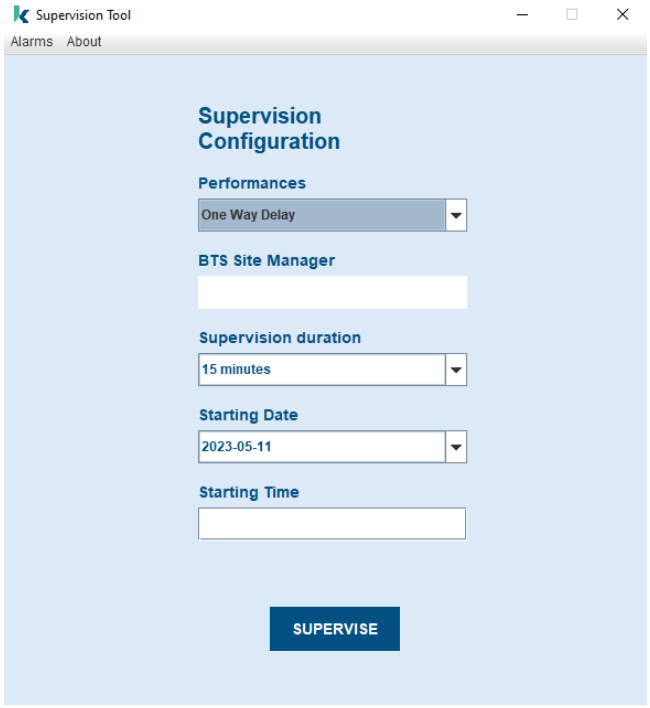

Figura 5.2: Ecrã de Configuração da Supervisão.

Ecrã da Configuração Alarmística

Na Figura 5.3, é apresentado o ecrã da configuração alarmistica. Cabe ao utilizador definir o número de *samples* a ter em conta para a análise e geração de alarmes, bem como a ativação ou desativação deste processo.

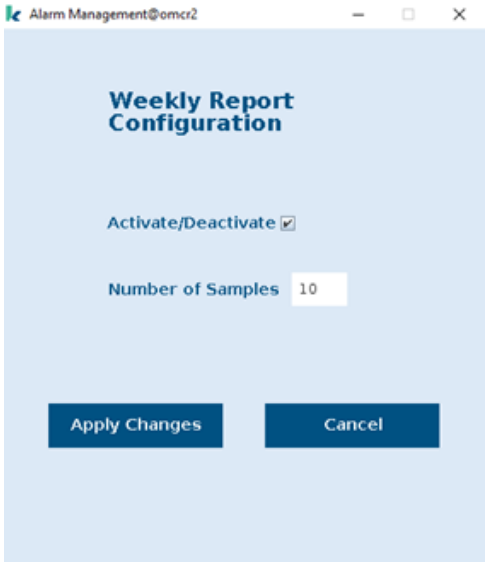

Figura 5.3: Ecrã da Configuração Alarmistica.

Ecrã da Verificação da Versão

Na Figura 5.4, é apresentado o ecrã da verificação da versão. Foi mais um requisito pedido e que consiste em manter o cliente atualizado das versões que vão sendo lançadas desta aplicação.

Figura 5.4: Ecrã da Verificação da Versão.

Ecrã da Análise do Desempenho

Na Figura 5.5, é apresentado um dos ecrãs da análise do critério de desempenho selecionado no painel da configuração da supervisão. Os dados são apresentados em gráficos, com os limites de fundo por cores consoante a severidade que representam para o sistema. A escala está

dividida em 3 cores: verde para "Situação Normal", amarelo para "Tolerância Esporádica"e vermelho para "Risco para o serviço". Além da informação visual, ainda é possível extrair relatórios em formato Excel com a imagem do gráfico selecionado e os dados monitorizados numa tabela.

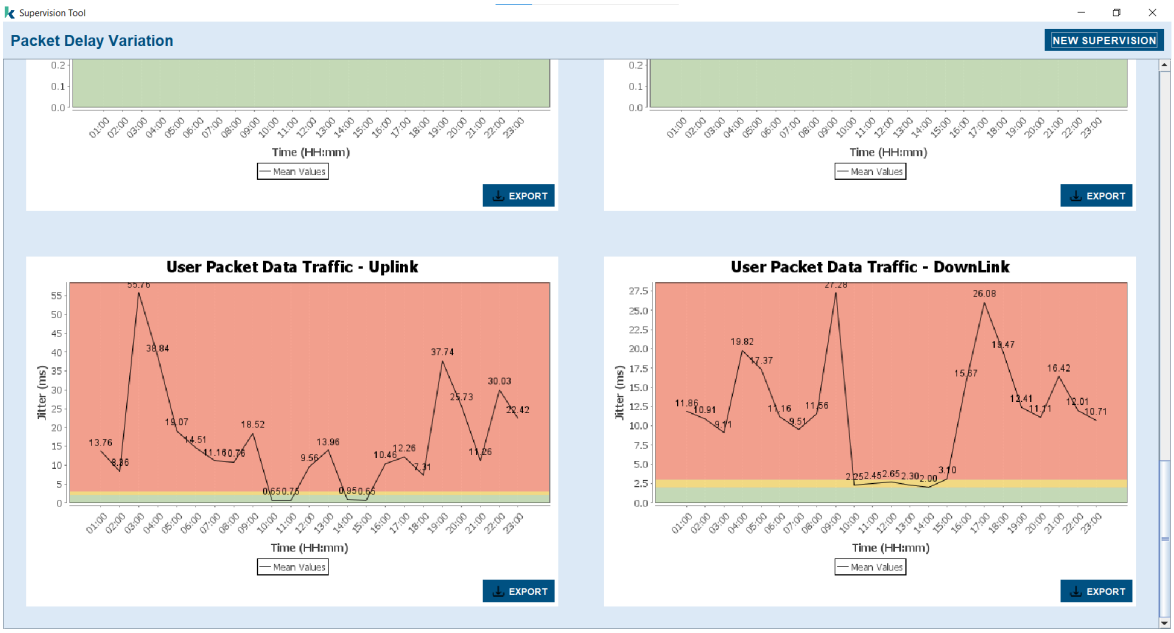

Figura 5.5: Ecrã da Análise do Desempenho.

5.2 Testes

Com o propósito de avaliar a funcionalidade e assegurar o cumprimento dos requisitos, a ferramenta desenvolvida foi submetida a casos de testes com condições iniciais. Foi necessário a definição de uma sequência de execução e os resultados esperados, identificar recursos e indicadores de desempenho e segurança e analisar a introdução de funcionalidades. Além da entrega de correções para que os problemas encontrados tenham sido resolvidos e não tenham surgido novos. Desta forma, a eficiência e a coerência dos testes aplicados é garantida.

O ambiente dos testes caracteriza-se pelos seguintes fatores:

- Servidor do omc: OMCR3
- Localização da ferramenta: "./opt/kontron/SupervisionTool"
- Localização do *script* de arranque: "./opt/kontron/SupervisionTool/bin/start.sh"
- Localização dos dados para monitorização: "/SDO/data/obs/compressed_extended"

A tabela que contém a especificação dos testes está presente no Apêndice 6.2. De salientar que a tabela contém os testes efetuados á primeira versão da ferramenta. O que significa que os testes reprovados já foram devidamente corrigidos, o que permitiu manter a coerência dos testes.

De uma forma geral, a sequência de execução inicia-se pela instalação do pacote rpm, segue-se o arranque da aplicação em dois cenários previsiveis: um com dados de observação e outro sem dados de observação. De seguida, efetuou-se a verificação das funcionalidades

presentes: o acesso aos menus de *dropdown* na barra superior, os campos de configuração presentes no painel principal e o botão que inicia a supervisão dos dados estatísticos. No que diz respeito à instalação do pacote, não houve quaisquer problemas com a estrutura previamente definida para a árvore dos diretórios e ficheiros que compoêm a ferramenta. Assim, a instalação decorreu como previsto, com um resultado positivo para um cenário com dados de observação. No entanto, existe a restrição ao utilizador *root* que permanece como titular da ferramenta. Esta decisão foi tomada na construção do ficheiro SPEC e oferece maior flexibilidade ao cliente, beneficiando das suas funcionalidades tanto no modo *online* como *offline*.

No que diz respeito à barra de menus, o painel relacionado com a configuração de alarmes é agora exibido e o ficheiro de configuração é atualizado de forma adequada quando se pressiona o botão para aplicar essas alterações no ficheiro. A única observação feita foi em relação ao tempo de resposta, que estava um pouco longo, mas isso foi devidamente corrigido para tornar o processo mais suave. A alteração consistiu em adicionar um círculo de carregamento para indicar que há um processo em curso e enquanto este não estiver concluído, a aplicação não pode ser utilizada. Esta melhoria foi aplicada a todas as ações que tinham tempos de resposta mais longos.

No painel relacionado com a informação da versão da ferramenta, foram feitas algumas alterações estéticas com base nas observações feitas. Nomeadamente, as cores utilizadas foram modificadas para melhorar a aparência.

No painel de configuração da supervisão, foram realizados testes para todos os diferentes parâmetros de entrada em todos os casos possíveis. Nos campos onde é possível introduzir dados, foram inseridos dados de teste com formatos aceitáveis e não aceitáveis pela ferramenta para verificar a integridade dos dados. A ferramenta reagiu corretamente aos dados inseridos, mostrando mensagens de erro quando o valor não estava de acordo com o formato pré-definido.

Após a introdução de dados válidos para supervisionar uma determinada interface, procedeu-se à ação de pressionar o botão para iniciar o processamento desses dados. O resultado deste teste foi inconclusivo porque os dados estatísticos que estavam a ser recolhidos pelo OMC-R não estavam corretos, resultando num painel vazio. Por um lado, considera-se que a aplicação analisou corretamente os dados de entrada inseridos visto que, como não existem dados nos ficheiros recolhidos pela interface SDO do OMC-R, o painel apresenta as mensagens com o conteúdo "No traffic". Por outro lado, não se concluiu o resultado esperado caso existissem dados corretos. No entanto, durante o desenvolvimento houve fases de teste, com a equipa presente, e conseguiu-se provar a vericidade dos dados e dos resultados gráficos.

CAPÍTULO 6

Conclusão

6.1 Considerações Finais

O diagnóstico de falhas está a revelar-se um desafio intenso devido à crescente complexidade das redes celulares emergentes. A heterogeneidade dos comportamentos das ligações da rede é identificada como um dos problemas para as falhas de desempenho do serviço GSM-R. A monitorização das interfaces dos componentes constituintes da rede é uma das soluções para prever e recuperar de eventos como estes. Este trabalho teve como objetivo a criação de uma aplicação capaz de supervisionar as interfaces de um nó BTS-R IP da rede GSM-R. É capaz de expôr os problemas de desempenho sofridos pela Rede de Transporte de Pacotes durante um período de supervisão.

Este projeto começou com a análise de documentação técnica do ambiente onde se iria inserir a aplicação, bem como das tecnologias que iriam ser necessárias à fase de desenvolvimento, tais como: (i) a análise da rede GSM-R, (ii) a instalação de um serviço do gestor de manutenção e da consequente aprendizagem de funcionamento, (iii) a instalação de um simulador de dados estatísticos para os testes futuros, (iv) a análise dos ficheiros e dos dados estatísticos recolhidos e (v) a configuração do ambiente de desenvolvimento. A escolha cuidadosa das tecnologias permitiu a criação de uma aplicação robusta, com uma interface gráfica atraente e recursos avançados de manipulação de dados. A combinação destas ferramentas e bibliotecas demonstra a preocupação em adotar soluções eficientes e confiáveis para o desenvolvimento do projeto.

A recolha dos valores das medições foi feita com a criação de algoritmos para ler e processar um enorme volume de dados. Este processo foi essencial para o progresso do projeto visto que, os resultados obtidos derivam do processamento dos dados, mantendo sempre a integridade e a fiabilidade dos mesmos. A criação da interface gráfica levou à concretização da visualização dos dados, um requisito fundamental deste projeto. A análise da informação recolhida nas interfaces monitorizadas permite a perceção do comportamento destas interfaces, em diferentes períodos de supervisão. Já a funcionalidade de extração dos relatórios permite que a informação recolhida, de maior relevância, se mantenha armazenada por tempo indeterminado.

Consoante os desvios em relação aos limites de severidade, a produção de alarmes permite ao operador uma análise mais abrangente e atenta do desempenho da Rede de Transporte de Pacotes.

Em suma, a construção desta ferramenta permitiu a monitorização de qualquer desvio do comportamento nominal de uma interface de um determinado nó. Desta forma, o operador adota as devidas precauções para antecipar eventos ou falhas na rede.

6.2 Trabalho Futuro

De acordo com os resultados obtidos, existem melhorias que podem ser realizadas ao nível do processamento de dados e da interface gráfica. Nos algoritmos de processamento, é possivel torná-los mais eficentes e precisos, o que resulta em ganhos de desempenho da aplicação, pode torná-la visualmente mais interativa. Isto pode ser feito com a otimização dos algoritmos recorrendo ao uso de técnicas de paralelização ou mecanismos de memória que acessem mais rápidamente os dados.

A interface gráfica pode também ser alvo de alterações e melhorias, de forma a torná-la mais intuitiva, atrativa e fácil de utilizar. Uma abordagem eficaz envolve a realização de testes de usabilidade mais específicos, que avaliem a navegabilidade, com o propósito de otimizar os fluxos dentro da aplicação. Pode-se considerar a adição de novas funcionalidades às interfaces de gráficos de desempenho, como, por exemplo, tornar a escolha dos intervalos de tempo mais dinâmica através da introdução de um campo que permita a alteração dos intervalos de supervisão. Também é possível possibilitar a exportação dos ficheiros em novos formatos, de forma a facilitar a integração com outras ferramentas em diferentes contextos.

Referências

[1] M. S. Riaz, H. N. Qureshi, U. Masood, A. Rizwan, A. Abu-Dayya e A. Imran, «Deep Learning-based Framework for Multi-Fault Diagnosis in Self-Healing Cellular Networks,» *IEEE Wireless Communications and Networking Conference, WCNC*, vol. 2022-April, pp. 746–751, 2022, ISSN: 15253511. DOI: 10.1109/WCNC51071.2022.9771947.

[2] K. Zhang, M. Kalander, M. Zhou, X. Zhang e J. Ye, «An Influence-Based Approach for Root Cause Alarm Discovery in Telecom Networks,» *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12632 LNCS, pp. 124–136, 2021, ISSN: 16113349. DOI: 10.1007/978-3-030-76352-7_16.

[3] S. Bartolini, P. Foglia, C. P. -, INFORMATICS, V. VII e undefined 2003, «Extending SNMP Management to GSM Radio Devices,» *garga.iet.unipi.itS Bartolini, P Foglia, CA Prete7TH WORLD MULTICONFERENCE ON SYSTEMICS, CYBERNETICS AND INFORMATICS, 2003 • garga.iet.unipi.it*, URL: http://garga.iet.unipi.it/paper/2003/SNMP%20BFP-SCI03-READY.pdf.

[4] *GSM BSS Overview*. URL: https://mysnt.sharepoint.com/sites/KTR-cust-rlw-ntp/Access/BSS/Pick%20and%20Choose%207%20(PC7)/GSM%20BSS%20Overview.pdf?CT=1705062272696&OR=ItemsView.

[5] *GSM RDN.Access Network Manager (OMC-R) Fundamentals*. URL: https://mysnt.sharepoint.com/sites/KTR-cust-rlw-ntp/OAM/OMCR/Pick%20and%20Choose%207%20(PC7)/GSM%20OMC-R%20Fundamentals.pdf?CT=1705062122370&OR=ItemsView.

[6] *GSM BTS-R Fundamentals*. URL: https://mysnt.sharepoint.com/sites/KTR-cust-rlw-ntp/Access/BTS/BTS-R/Pick%20and%20Choose%207%20(PC7)/GSM%20BTS-R%20Fundamentals.pdf?CT=1705062181735&OR=ItemsView.

[7] *GSM BSS Performance Management— Observation Counters Dictionary*. URL: https://mysnt.sharepoint.com/sites/KTR-cust-rlw-ntp/Access/BSS/Pick%20and%20Choose%207%20(PC7)/GSM%20BSS%20Performance%20Management%20-%20Observation%20Counters%20Dictionary.pdf?CT=1705062229512&OR=ItemsView.

Apêndice A

Testes Realizados

Supervision tool - version 00.03

Teste #	Descrição / Condição	Resultado Expectável	Resultado
1	Instalação - caso nominal	Ferramenta instalada com toda a estrutura pertencente ao grupo omc	REPROVADO
2	Arranque da aplicação	Ferramenta em execução	APROVADO
3	Arranque da aplicação num cenário com dados de observação	Ferramenta em execução após descomprimir todos os ficheiros de dados	APROVADO
4	Arranque da aplicação num cenário sem dados de observação	Ferramenta em execução mostra um aviso de que não existe dados para supervisionar	REPROVADO
5	Verificar o menu "*Alarms* "	O menu lista a opção "*Report Configuration"*	APROVADO
6	Verificar o menu "*Report Configuration* "	Abrir a janela "*weekly report configuration* "onde é possível "*Activate/Deactivate* ", especificar o "*Number of Samples* " e "*Apply Changes* ".	APROVADO
7	"Report Configuration" window: configure Activate/Deactivate	O ficheiro *ReportConfig.cfg* é atualizado corretamente.	APROVADO
8	"Report Configuration" window: configure the number of samples	O ficheiro ReportConfig.cfg é atualizado corretamente.	APROVADO
9	Perform changes in the Report Configuration and in the end Apply Changes	É aplicada uma nova configuração	REPROVADO
10	Verificar o menu "*About* "	O menu lista a opção "*Check version* "	APROVADO
11	Verificar o menu "*Check Version* "	Abrir a janela "*check version* " onde é apresentada o número da versão da ferramenta	APROVADO
12	Configuração da Supervisão - Verificar o campo de configuração "*Performances* "	Não é possivel escrever neste campo de configuração porque é para ser preenchido por uma das três opções disponíveis: "*One way delay* ", "*Packet delay variation* " e "*Packet loss* ".	APROVADO
13	Configuração da Supervisão - Verificar o campo de configuração "*BTS Site Manager* " com formato aceitável	Aceita os dados introduzidos	APROVADO
14	Configuração da Supervisão - Verificar o campo de configuração "*BTS Site Manager* " com formato não aceitável	Não aceita os dados introduzidos	APROVADO
15	Configuração da Supervisão - Verificar o campo de configuração "*Supervision Duration* "	Não é possível escrever neste campo de configuração porque é para ser preenchido por uma das quatro opções disponiveis: "15 *minutes* ", "1 *hour* ", "24 *hours* ", "1 *week* ".	APROVADO
16	Configuração da Supervisão - Verificar o campo de configuração "*Starting date* "	É apresentada a lista das datas disponiveis	APROVADO
17	Configuração da Supervisão - Verificar o campo de configuração "*Starting time* " com formato aceitável	Aceita a hora introduzida	APROVADO
18	Configuração da Supervisão - Verificar o campo de configuração "*Starting time* " com formato não aceitável	Não aceita a hora introduzida	APROVADO
19	Configuração da Supervisão - Verificar o botão "*Supervise* "	Executa a supervisão dos dados no periodo de tempo escolhido	INCONCLUSIVO