João Pedro Lopes de Oliveira

Licenciado em Ciências da Engenharia Eletrotécnica e de Computadores

Deteção de anomalias em sequências de pedidos a servidores de redes de distribuição de vídeo

Dissertação para obtenção do Grau de Mestre em

Engenharia Eletrotécnica e de Computadores

Orientador: Pedro Miguel Figueiredo Amaral, Professor Auxiliar, Faculdade de Ciências e Tecnologia, Universidade NOVA de Lisboa

Júri

Presidente:	Fernando José Vieira do Coito
Arguente:	Paulo da Fonseca Pinto
Vogal:	Pedro Miguel Figueiredo Amaral

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Novembro, 2020

Deteção de anomalias em sequências de pedidos

Este documento foi gerado utilizando o processador (pdf)LaTeX, com base no template "novathesis" [1] desenvolvido no Dep. Informática da FCT-NOVA [2].
[1] https://github.com/joaomlourenco/novathesis [2] http://www.di.fct.unl.pt

Agradecimentos

Quero expressar o meu sentido agradecimento a todos aqueles que, de alguma forma, contribuíram para o meu sucesso, não só na realização desta dissertação mas também ao longo de todo o percurso académico.

Em primeiro lugar quero agradecer ao Professor Pedro Amaral, pela orientação, disponibilidade e prontidão demonstrada sempre que foi solicitada ao longo da realização desta dissertação.

À Faculdade de Ciências e Tecnologia da Universidade NOVA de Lisboa agradeço por me ter proporcionado uma ótima aprendizagem juntamente com as excelentes experiências ao longo dos anos.

Resumo

As crescentes exigências dos clientes e a grande competitividade no setor obrigam a que as operadoras estejam constantemente a necessitar de atualizar e melhorar o seu sistema de distribuição de vídeo para garantir a continuidade dos seus subscritores ou para aumentar o interesse de outros clientes.

Como tal, nesta dissertação pretende-se desenvolver um algoritmo que possa ser utilizado em tempo real que ajude a operadora a identificar as respostas de um *backend* correspondentes ao pedido efetuado pelo *frontend*. O algoritmo baseou-se em algumas regras propostas através do estudo da arquitetura. Foi necessário utilizar várias tabelas de diferentes bases de dados para conseguir fazer a correlação dos pedidos.

Alguns pedidos demoram mais tempo que o esperado e por isso considera-se que houve um atraso significativo ou um erro no pedido ou no seu processamento. Para evitar isto, pretende-se ainda fazer um estudo estatístico que descubra falhas no sistema, nomeadamente nas *set-top-box* (STB), nas funções associadas aos pedidos dos utilizadores ou no servidor que processou o pedido, para que a operadora consiga, através de otimizações no hardware ou no software, reduzir a frequência com que acontecem.

A versão da STB "NDS_HDZ_4_10_0" apresenta um valor médio de duração dos pedidos com atraso demasiado elevado. Verificou-se também que pode existir um problema particular com a função "getEPGRating" ao ser executada pela versão "NDS_PVR_4_9_0". Estes tipos de problemas podem ser provenientes de falhas na otimização da função ou de problemas com o *software* ou *hardware* dos dispositivos.

Palavras-chave: Cliente, Operadora, pedido, resposta, distribuição de vídeo, dados, algoritmo, otimização, *set-top-box*, falhas, atrasos, erros, servidor, *frontend*, *backend*.

Abstract

The growing demand from clients and the big competitive edge on the sector are forcing the companies to be frequently updating and improving their video distribution system to ensure their clients permanency and to raise other clients interest.

As so, in this thesis it is intended to develop an algorithm that may be used in real time to help the company to identify backend responses corresponding to requests made by the frontend. The algorithm is based on a few proposed rules discovered through the study of the arquitecture. It was necessary to use multiple tables from different databases to be able to to correlate the requests.

Some requests take more time than expected and for that it is considered that has been a significant delay or an error in that request or in its processing. To avoid this, it is also intended to do a statistical study to find flaws on the system. This study will be done specifically on the STBs, on the function related to the requests and on the server that processed the request. This is done so the company can update their hardware and software to reduce the frequency that this flaws happens.

The STB version "NDS_HDZ_4_10_0" shows an average duration of the requests that has too high delay. It was also found that may exist a problem in the particular function "getEPGRating" when executed by the version "NDS_PVR_4_9_0". These kind of problems may come from flaws in the function optimization or from problems on software or hardware on devices.

Keywords: Client, company, request, responses, video distribution, data, algorithm, optimization, set-top-box, flaws, delays, errors, server, frontend, backend

Índice

Lista de Figuras

Lista de Tabelas

Siglas

AAAA	Authentication, Authorization, Accounting, and Auditing.
API	*Aplication Programming Interface*.
BD	Base/es de Dados.
BE	*Back-end*.
FE	*Front-end*.
HMM	*Hidden Markov Model*.
HTTP	*Hypertext Transfer Protocol*.
I/O	*Input/Output*.
KDD	*Knowledge Discorevy from Data*.
QoS	*Quality of Service* — Qualidade do Serviço.
REST	*Representational State Transfer*.
SPM	*Sequential Pattern Mining*.
STB	*Set-Top-Box*.
URI	Uniform Resource Identifier.

Capítulo 1

Introdução

1.1 Enquadramento e Motivações

Hoje em dia, a popularidade da Internet obrigou a que as operadoras de rede disponibilizassem ofertas que incluem, na sua grande maioria, plataformas de acesso a vídeo, que normalmente são baseadas numa estrutura de servidores que utilizam *RESTful APIs*. Este tipo de APIs são baseados na tecnologia REST e que usam mensagens *Hypertext Transfer Protocol* (HTTP) para fazer a gestão dos pedidos dos clientes. Nesta arquitetura, os pedidos feitos pelos clientes são geralmente tratados pelo *front-end* (FE) (normalmente são as plataformas utilizadas para interagir com os utilizadores) que emitirá uma mensagem ao *back-end* (BE) (sistema utilizado para gerir os dados da rede), de forma a que este saiba o que é pretendido e envie uma resposta com a informação.

A qualidade do serviço prestado pelas operadoras é um assunto de grande importância, uma vez que os consumidores cada vez são mais exigentes quanto ao serviço que estão a subscrever. Garantir a melhor qualidade de serviço (QoS) por parte das operadoras requer a análise de uma enorme quantidade de dados gerada a partir das interações entre o utilizador e os servidores, que permitem a extração de informação relevante para uma melhor gestão do sistema. Tentar prever as ações dos utilizadores, envolve analisar sequências de pedidos de utilizadores, para descobrir padrões relevantes, com a finalidade de diminuir os tempos de resposta aos pedidos e também de detetar sequências anómalas que possam ser prejudiciais ao sistema.

Tendo em conta a quantidade de tempo despendido no tratamento destes dados, a "mineração" de dados tornou-se uma ferramenta de grande importância para a análise de dados, oferecendo uma panóplia de algoritmos capazes de processar dados autonomamente.

Em HTTP, todos os pedidos efetuados a partir do cliente e as respetivas respostas são

guardadas em ficheiros de registo (ficheiros *.log*), ficheiros estes que são praticamente inelegíveis antes de serem submetidos a um processo de análise sintática (*parsing*), que é o processo de análise de um *input*, segundo uma determinada gramática formal, para determinar a sua estrutura gramatical.

Posto isto, nasce a necessidade de encontrar uma solução fiável para fazer uma análise minuciosa dos dados e tirar conclusões concretas acerca dos resultados obtidos, para que as empresas possam melhorar o serviço prestado aos clientes.

O grande desafio desta dissertação passa por correlacionar pedidos entre o servidor de Frontend e o servidor de Backend sem que estes tenham algum tipo de identificador comum aos dois.

1.2 Objetivos e Contribuições

Os objetivos da presente dissertação são os seguintes:

- Identificação de sequências de pedidos, do cliente para o *front end* e subsequencialmente entre serviços internos (*front end* - *back end* - base de dados);
- Desenvolvimento de *parsers*, com o intuito de tornar legíveis os ficheiros que contém os registos;
- Desenvolvimento de um algoritmo com o objetivo de identificar anomalias;
- Correlação de dados sem ID comum;

Esta dissertação contribuiu ainda com diversos estudos estatísticos que podem indicar alguns erros/problemas na rede da operadora e também com um algoritmo capaz de correlacionar os pedidos que não existia até à data.

Capítulo 2

Trabalho relacionado

2.1 Introdução

De um modo simples, *data mining* refere-se à extração ou "mineração", a partir de grandes quantidades de informação, de conhecimento previamente desconhecido e com potencial utilidade. Seria mais apropriado denominar-se *knowledge mining from data* ou em português, "Extração de conhecimento a partir de dados", o que torna, a sua pronuncia torna-se demasiado longa. Por outro lado, *knowledge mining* (mineração/extração de conhecimento), não reflete o ênfase que é suposto dar à "mineração" a partir de grandes quantidades de dados. Ainda assim, existem outros termos que carregam um significado similar ou ligeiramente diferente, como por exemplo "Extração de conhecimento a partir de dados", "Extração de conhecimento", "Análise de dados/padrões", "Arqueologia de dados", entre outros [1][2]. *Data mining* é por muitos considerado um sinónimo de outro termo bastante usado, *Knowledge Discovery from Data*, ou *KDD*. Alternativamente, na visão de outros, *data mining* é simplesmente um passo essencial no processo de *Knowledge Discovery*, que consiste numa sequência iterativa dos seguintes passos [2]:

1. Limpeza de dados (remoção de inconsistências);

2. Integração de dados (onde múltiplas fontes podem ser combinadas);

3. Seleção de dados (onde os dados relevantes para a tarefa de análise são retirados da base de dados (BD);

4. Transformação de dados (onde os dados são transformados ou consolidados em formas apropriadas para a mineração, realizando operações de sumarização ou agregação, por exemplo);

5. Mineração de dados (um processo essencial onde métodos inteligentes são aplicados com o objetivo de extrair padrões);

6. Avaliação de padrões (para identificar os padrões verdadeiramente interessantes);

7. Apresentação de conhecimento (são utilizadas técnicas de representação e visualização para apresentar o conhecimento adquirido ao utilizador).

Os passos de 1 a 4 são métodos de pré processamento de dados, os quais são preparados para serem "minerados". O passo número 4 pode interagir com o utilizador ou com uma base de conhecimento. Os padrões que suscitem algum interesse são apresentados ao utilizador e podem ser guardados como novo conhecimento. Repare-se que segundo este ponto de vista, *data mining* é apenas um passo de um processo inteiro, embora seja um passo essencial porque permite a descoberta de padrões para avaliação [2].

2.1.1 Tipos de dados

Devido ao facto de existir diferentes tipos de dados, o conhecimento extraído será diferente, e consequentemente a técnica aplicada ao tratamento desses dados será também diferente, o que fará com que a distinção dos tipos de dados que serão tratados seja feita corretamente [1].

- **Base de dados relacional** — Até agora, a maior parte dos dados são armazenados neste tipo de base de dados. Estes são repositórios de dados altamente estruturados. Os dados são descritos como um conjunto de atributos e armazenados em tabelas. Com o desenvolvimento de linguagens *query*, explorar estes dados tornou-se muito mais simples. A exploração de dados neste tipo de bases de dados foca-se principalmente na descoberta de padrões e tendências;

- **Base de dados Transacional** — Refere-se ao conjunto dos registos de transações. Aqui o foco é a exploração de regras de associação, descobrir correlações entre objetos de registos transacionais;

- **Base de dados Espacial** — Geralmente contém, não só tipos de dados tradicionais mas também localizações e informação geográfica acerca dos correspondentes dados. As regras de associação espacial descrevem a relação ente um conjunto de características e outro conjunto de características numa base de dados espacial;

- **Base de dados temporal ou de séries temporais** — Difere de uma base de dados tradicional na medida em que nesta, para cada objeto temporal, existe um atributo relacionado ao tempo correspondente que é associado. As regras de associação temporal podem ser mais úteis e providenciar mais informação que as associações básicas.

Tendo em conta o problema que esta dissertação se propõe a resolver, podemos definir a nossa base de dados inicial como temporal, devido ao facto de se estar a trabalhar com o protocolo HTTP, cujos pedidos efetuados pelos clientes e as devidas respostas contém um parâmetro de carácter temporal, neste caso data, hora e tempo de propagação da mensagem.

2.1.2 Tipos de mining

De um modo geral, há duas classes de métodos de *data mining*, que podem ser descritivos e prescritivos. A abordagem descritiva, é utilizada para sumarizar ou caracterizar propriedades gerais dos dados num certo repositório. Há quem defenda [3] que este tipo de abordagem é considerada *data mining* enquanto que a prescritiva, que serve para realizar inferências acerca da informação atual, com o objectivo de fazer previsões baseadas nos dados históricos, é condiderada *machine learning*, uma vez que existe uma aprendizagem envolvida. Como a descoberta de modelos de aprendizagem simplesmente a partir dos dados que se encontram no repositório ou a dependência de suposições pontuais não satisfazem os conceitos de uma análise sólida e fidedigna, uma combinação entre estes dois métodos pode vir a ser vantajosa para a criação de modelos preditivos. Há vários tipos de técnicas incluídas nestas classes, como por exemplo: regras de associação, classificação, agregação, análise de padrões sequenciais [1].

- **Regras de Associação** — Uma das mais importantes e mais investigadas técnicas, foi pela primeira vez introduzida por [4] e procura extrair correlações interessantes, padrões frequentes ou associações dentro de um conjunto de elementos inserido num determinado repositório. É um processo composto por duas partes. Primeiro procede-se à identificação dos conjuntos de estudo dentro da base de dados, depois já é possível realizar inferências acerca desses conjuntos [5];

- **Regras de Classificação** — é o processo de construção (automática) de um modelo que consiga descrever e distinguir classes de dados ou conceitos, com o objetivo de ser usado para prever a classe de objetos que não é conhecida [2]. Este processo é constituído por duas fases. A primeira, baseada no processo de recolha de um conjunto de dados , é construído um modelo que descreve as características das classes ou dos conceitos do conjunto estudado. Como estas classes ou conceitos estão predefinidos, esta etapa também é conhecida por aprendizagem supervisionada. Na segunda etapa, o modelo anteriormente criado é utilizado para prever as classes dos futuros objetos [1].
  O modelo de classificação pode ser representado através de várias técnicas como as regras básicas de classificação (IF-THEN), árvores de decisão, fórmulas matemáticas, ou redes neuronais. Uma árvore de decisão é um fluxograma em estrutura de árvore, onde cada nó representa um teste a um atributo, cada ramificação representa o resultado desse teste e as "folhas" representam as classes. As árvores de decisão

podem ser facilmente convertidas em regras básicas de classificação. Uma rede neuronal, quando usada para classificar, é tipicamente uma colecção de unidades de processamento semelhante a neurónios, cujas conexões entre si têm diferentes pesos. Existem muitos outros métodos para construir modelos de classificação, como a classificação Bayesiana, o *k-nearest neighbor* e *support vector machines* [1][2];

- **Regras de Agregação** — é uma técnica semelhante à classificação. No entanto, a agregação é um processo não supervisionado. É um processo de agrupamento de um conjunto de objetos reais ou abstratos em classes às quais estes se assemelhem, ou seja, os objetos dentro do mesmo grupo têm de ser similares em algum aspecto, e ainda, têm de ser diferentes dos objetos que pertençam a outros grupos. A diferença entre classificação e agregação é que enquanto na primeira os objetos são classificados por classes previamente definidas, aqui os objetos não são "rotulados" mas sim agrupados consoante a sua semelhança entre si [1][2];

- **Padrões sequenciais** — Esta técnica pretende descobrir de que modo um elemento pode suceder a um conjunto de elementos numa sequência temporal de sessões ou episódios [6]. São estudadas quais as sequências de eventos que são encontradas frequentemente juntas e descobrir padrões.

Atendendo ao que foi dito nesta secção, em 1.1 e aos objetivos do trabalho a realizar, optou-se por estudar um tema com grande incidência na deteção de padrões e sequências relevantes, assim como na descoberta de anomalias nas sequências anteriormente referidas, os Padrões Sequenciais (do inglês *Sequential Pattern Mining – SPM*) que irá ser abordada na secção seguinte.

2.2 Padrões sequenciais

A técnica *SPM* foi introduzida pela primeira vez por [4]. Esta é uma técnica de *data mining* muito importante e, como será mostrado mais à frente, tem diversas aplicações.

2.2.1 Definição do problema

Esta secção irá definir o problema em torno da exploração de padrões sequenciais com base em [7] e [8], bem como explicar alguns conceitos utilizados, definições e alguns algoritmos bastante utilizados para a descoberta de padrões e sequências frequentes.

Para entender melhor o problema, será introduzido algum vocabulário e algumas definições, que representam as bases do *SPM* e subsequentemente dos algoritmos abrangidos. Seja:

(i) D um conjunto de sequências que representa uma base de dados temporal;

(ii) ξ um valor de suporte (*treshold*) mínimo definido previamente;

(iii) $E = \{e_1, e_2, ..., e_k\}$ um conjunto de k eventos únicos.

O objetivo do *SPM* passa essencialmente por descobrir todas as sequências S que aparecem frequentemente na base de dados em estudo, ou seja, que ultrapassem o valor de *treshold* mínimo definido, com a finalidade de prever futuros elementos [1].

Definição 1 *Dado um conjunto de sequências, onde cada sequência consiste numa lista de eventos, e cada evento consiste num conjunto de elementos, e dado um treshold mínimo, ξ, o SPM descobre todas as subsequências frequentes, ou seja, aquelas cuja sua ocorrência se verifique no mínimo ξ vezes* [4][9][10][11].

Definição 2 *Seja E o conjunto de todos os eventos, com E = {e_1, e_2, ..., e_k}. Um conjunto pode ser visto como uma coleção não vazia e desordenada de elementos.*

Um evento é considerado um conjunto de elementos, denotado $(x_1 x_2 \cdots x_q)$, em que x_k é um elemento pertencente ao conjunto. Os parênteses são omitidos se o conjunto for composto apenas por um elemento. A este conjunto chama-se conjunto de elemento único.

Definição 3 *Considera-se que uma sequência é uma lista ordenada de eventos, $S = \langle e_1 e_2 e_3 \cdots e_m \rangle$ em que a ordem dos eventos é um atributo com especial importância. Neste caso e_1 ocorre antes de e_2, que por sua vez ocorre antes de e_3 e assim sucessivamente.*

Tomando de exemplo a Sequência 1 da tabela 2.1, $S = \langle a(abc)(ac)d(cf) \rangle$. Esta sequência tem 5 eventos distintos, *(a)*, *(abc)*, *(ac)*, *(d)* e *(cf)*, que ocorrem por esta ordem. Um elemento pode ter apenas uma ocorrência num conjunto, mas pode aparecer múltiplas vezes em conjuntos diferentes dentro da mesma sequência. O elemento *(a)* tem três ocorrências, em *(a)*, *(ac)* e em *(abc)*. Assim, se fosse inserido um conjunto/evento *(beb)* a esta sequência, não seria válida, visto que tem duas ocorrências do mesmo elemento.

Definição 4 *Considerando a sequência S, o valor de suporte, ξ, pode ser definido através de:*

$$\xi = \frac{Número\ de\ sequências\ sobre\ as\ quais\ S\ é\ uma\ subsequência}{Número\ total\ de\ sequências\ em\ D} \tag{2.1}$$

Onde a contagem de suporte de S é o número total de sequência em D, sobre as quais S é subsequência [1].

Uma sequência é considerada como frequente, se a sua contagem de suporte (frequência) for igual ou superior a ξ. Para ser considerada, a sequência terá de aparecer, pela mesma ordem da sequência de teste, podendo ter conjuntos intermédios que não pertençam à sequência de teste. Tomando $T = \langle (ab)c(de)f \rangle$ como exemplo de uma sequência observada e $s_1 = \langle (ab)f \rangle$ a sequência de teste, podemos admitir que esta se encontra inserida em T, pois os subconjuntos *(ab)* e *(f)* encontram-se pela ordem correta na sequência

observada, apesar de não aparecerem seguidos um do outro (estão separados pelos conjuntos *(c)* e *(de)*) [2].

Considerando um conjunto de eventos únicos distintos, sob as condições acima descritas, $E = \{a, b, c, d, e, f\}$. Um exemplo de conjunto de sequências será apresentado na Tabela 2.1 [2].

Tabela 2.1: Exemplo de conjunto de sequências.

ID	Sequência
1	⟨a(abc)(ac)d(cf)⟩
2	⟨(ad)c(bc)(ae)⟩
3	⟨(ef)(ab)(df)cb⟩
4	⟨eg(af)cbc⟩

Observando a subsequência $s = \langle (ab)c \rangle$ e assumindo que o valor de ξ é 2, então, as únicas sequências da Tabela 2.1 nas quais a subsequência s está incluída são a 1 e 3, ou seja, satisfazem a condição de suporte de 2 e por isso são consideradas sequências frequentes.

2.2.2 Taxionomia

Ainda seguindo [7], nesta secção ir-se-á adotar uma taxionomia semelhante, fazendo uma análise comparativa entre as três categorias gerais nas quais os algoritmos de *SPM* estão inseridos: *Apriori-based*, *Pattern-growth* e *Early-prunning*.

2.2.2.1 *Apriori-based*

De uma maneira simples, esta categoria é definida da seguinte forma: "Todas os subconjuntos não vazios pertencentes a um conjunto frequente, têm obrigatoriamente que ser conjuntos frequentes" [7].

- ***Breadth-first search*** é uma característica intrínseca dos algoritmos *apriori-based* devido ao facto de estes construírem todas as sequências-k possíveis em cada k-ésima iteração, enquanto percorrem a base de dados. Isto quer dizer que para resolver a k-ésima sequência, todas as sequências anteriores ($(k-1)$, $(k-2)$) tem necessariamente de estar resolvidas;
- ***Generate-and-test*** foi uma das primeiras características a ser introduzidas por [12]. Consiste em fazer crescer cada sequência, um elemento de cada vez, testado perante o valor de suporte mínimo. Isto faz com que seja gerada uma enorme quantidade de sequências candidatas, consumindo muita memória e tempo nas fases iniciais;
- ***Multiple scans of the database*** consiste em examinar a base de dados sempre que for necessário verificar se a sequencia candidata é frequente ou não. Esta é uma característica bastante inconveniente da maioria dos algoritmos *apriori-based* e exige um enorme tempo de processamento e custo I/O.

O número de sequências candidatas geradas a partir de algoritmos baseados no método *apriori-based*, é no pior dos casos, exponencial [13]. Como tal, considera-se que este método é pouco eficiente no que toca ao tempo de execução e memória consumida.

2.2.2.2 *Pattern-growth*

Esta classe de algoritmos tenta aliviar os problemas de complexidade temporal e espacial dos algoritmos *apriori-based*. A ideia principal é evitar certos passos que ocorrem no *apriori-based*, como a geração e supressão de candidatos e limitar repetidamente o espaço de procura, reduzindo a base de dados original em outras menores, que serão tratadas separadamente [14][13].

Os algoritmos pertencentes a esta categoria, são mais rápidos e consomem menos memória quando tratam grandes volumes de dados, por isso, são também mais complexos de desenvolver [2][10].

Existem vários métodos associados a esta classe de algoritmos, no entanto só iremos mencionar os mais importantes.

- ***Depth-first traversal*** utiliza um consumo de memória consideravelmente mais baixo, um espaço de procura mais direto e gera menos sequências candidatas que o *breadth-first*. Todas as sequências de um caminho necessitam de estar resolvidas antes de avançar para o caminho seguinte, ou seja, cada padrão é construido separadamente de forma recursiva [15][16];

- ***Tree projection*** é um método usado nos algoritmos de *pattern-growth*. Aqui é implementada uma estrutura de árvore em representação do espaço de busca, que depois será submetido a procuras *breadth-first* ou *depth-first* para obter sequências frequentes. Cada nó da árvore representa um elemento numa sequência juntamente com o seu valor de suporte, cada ramo representa uma sequência candidata completa;

- ***Search space partitioning*** é utilizado em alguns algoritmos *apriori-based*, mas é considerado um método característico dos algoritmos *pattern-growth*. Permite fazer partições no espaço de busca de grandes sequências candidatas, com o objetivo de fazer uma gestão da memória mais eficiente;

- Os algoritmos ***candidate sequence pruning*** utilizam uma estrutura de dados que permita fazer uma seleção das sequências candidatas numa fase prematura do processo de "mineração", com o objetivo de reduzir o tamanho da àrvore e consequentemente diminuir o tempo despendido no tratamento de sequências que não tenham potencial valor;

- ***Suffix/prefix growth*** é um método em que as sequências frequentes podem ser descobertas através da adição de sufixos ou prefixos que por si só sejam considerados frequentes. Os algoritmos que dependem de bases de dados projetadas ou de procuras condicionais em árvores, inicialmente procuram as sequências-k, geralmente

sequências de elemento único e agregam cada elemento frequente como prefixo ou sufixo e começam a construir sequências candidatas em torno desses elementos de forma recursiva. Isto reduz significativamente a quantidade de memória necessária para armazenar todas as sequências candidatas que partilham do mesmo prefixo/-sufixo.

2.2.2.3 *Early-pruning*

Pruning é uma técnica muitas vezes associada a árvores de decisão. Esta técnica reduz o tamanho das árvores de decisão através da remoção de partes da árvore que não providenciam grande capacidade de classificação, que consequentemente irá melhorar a precisão preditiva dos algoritmos.

- ***Support counting avoidance*** nasceu a partir da necessidade de reduzir o número de contagens em todo o processo de "mineração", e também como tentativa de eliminar os varrimentos iterativos das bases de dados sequenciais sempre que seja necessário calcular o valor de suporte. A maior vantagem deste método é permitir que as bases de dados sequenciais possam ser removidas da memória e não precisem de ser utilizadas novamente, uma vez que o algoritmo descobre um método de guardar as sequências candidatas juntamente com o seu valor de suporte;

- ***Vertical projection of the database*** reduz para um valor substancialmente menor (uma ou duas) as análises da base de dados. Um *bitmap* ou uma tabela de indicação de posição podem ser construídas para cada elemento frequente, obtendo um plano vertical da base de dados ao invés do habitual plano horizontal. O processo de extração de elementos/sequências frequentes utiliza apenas tabelas de plano vertical para gerar sequências candidatas;

- ***Position-coded*** é a ideia chave desta categoria de algoritmos, porque permite aos algoritmos olhar mais à frente e assim evitar gerar sequências candidatas não frequentes. Se um elemento ou sequência não for frequente, não faz sentido descobrir sufixos ou prefixos, uma vez que estes não serão frequentes.

2.3 Cadeia de Markov

A cadeia de Markov é uma técnica com bastante potencial para a descoberta de futuros estados em sequências, que será estudada de modo a prever possíveis sequências desencadeadas por pedidos de utilizadores. Uma cadeia de Markov é definida como um conjunto de estados juntamente com o conjunto de transições entre estados, onde cada arco representa uma transição à qual está associada uma probabilidade.

2.3.1 Definição de modelo discreto de uma cadeia de Markov

Um modelo discreto da cadeia de Markov pode ser definido através da tupla $\langle S, \mathbf{A}, \lambda \rangle$. S corresponde ao espaço de estados, **A** à matriz que representa as probabilidades de transição de um estado para outro, e λ a distribuição inicial de probabilidades dos estados em S.

Uma sequência de variáveis aleatórias discretas $\{C_n : n \in \mathbb{N}\}$ considera-se uma cadeia de Markov se, para todo o $n \in \mathbb{N}$ satisfizer a condição [17]:

$$P(C_{n+1}|C_{n,\ldots,1}) = P(C_{n+1}|C_n) \tag{2.2}$$

Isto é, conhecer o historial do processo até ao momento t, é equivalente a conhecer apenas o estado mais recente, C_n.

Seja o vetor $s(n)$ o vetor de probabilidades para todos os estados no momento t, então [17] [18]:

$$\hat{s}(n) = \hat{s}(n-1)\mathbf{A} \tag{2.3}$$

Se existirem n estados numa cadeia de Markov, então a matriz de probabilidades das transições A tem a dimensão $n \times n$ [18].

$$\begin{bmatrix} P_{11} & P_{12} & \ldots & P_{1n} \\ P_{21} & P_{22} & \ldots & P_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ P_{n1} & P_{n2} & \ldots & P_{nn} \end{bmatrix} \tag{2.4}$$

Uma cadeia de Markov é descrita da seguinte forma: Dado um conjunto de estados, S = $s1, s2, ..., sf$. O processo inicia-se num destes estados e move-se sequencialmente para o estado seguinte. Se a cadeia estiver no estado si , então a cadeia mover-se-á para o estado sj com uma probabilidade de p_{ij}. A probabilidade p_{ij} é chamada de probabilidades de transição. O processo pode permanecer no estado atual, sendo a probabilidade deste acontecimento é p_{ii}. Uma distribuição inicial de probabilidades, definida em S, especifica um estado em particular como estado inicial.

A matriz **A** e a distribuição inicial das probabilidades dos estados, λ, podem ser calculadas utilizando diversos métodos. Neste documento serão adoptados os seguintes [18]:

$$A(s, s') = \frac{C(s, s')}{\sum_{s''} C(s, s'')} \tag{2.5}$$

$$\lambda(s) = \frac{C(s)}{\sum_{s'} C(s')} \tag{2.6}$$

$C(s, s')$ é o número de vezes que que s é seguido de s' num conjunto de dados de teste/treino. Ainda que as cadeias de Markov sejam tradicionalmente utilizadas para caracterizar propriedades assimptóticas, a matriz de transição será utilizada para prever

sequências a curto prazo. Um elemento da matriz A, diga-se $A[s;s']$ pode ser interpretado como a probabilidade do estado s transitar diretamente para o estado s'. Do mesmo modo, um elemento de $A * A$ significa a probabilidade de transitar de um estado para outro em dois hops, e assim sucessiveamente [18].

As cadeias de Markov podem ser aplicadas à modelação de Web links. Neste caso, um estado de Markov pode corresponder a qualquer pedido feito pelo utilizador e a qualquer resposta do servidor ao pedido, sendo que estas são sempre mensagens HTTP.

2.3.2 Cadeia de Markov escondida

Os métodos utilizando cadeias de Markov escondidas (do inglês Hidden Markov Model – HMM) têm sido introduzidos para abordar o tipo de questões mencionados nas secções 1.1 e 2.1

2.3.2.1 Definição

A cadeia de Markov escondida é definida como um modelo estatístico geral para problemas lineares como sequências ou séries temporais. Uma HMM é um modelo finito que descreve a distribuição de probabilidades através de um número infinito de sequências possíveis [19].

Uma HMM trata-se de uma cadeia de Markov, e tem por base 2.3.1, com a particularidade de que essa sequência de estados não é observada: está escondida. Apenas a sequência de símbolos gerada pelos estados escondidos é visível.

O nome de cadeia de Markov escondida advém de duas propriedades gerais [20]:

Propriedade 1 *Assume-se que a observação no momento t, Y_n, foi gerada por um processo cujo estado S_n está oculto do observador.*

Propriedade 2 *Dado o valor de $S_{(n-1)}$, o estado atual S_n é independente de qualquer estado anterior a (n - 1).*

Por outras palavras, cada estado alberga toda a informação sobre a história do processo necessária para prever futuros estados do processo.

2.3.3 Aplicação

As cadeias de Markov podem ser aplicadas à modelação de *Web links*. Neste caso, um estado de Markov pode corresponder a qualquer pedido feito pelo utilizador e a qualquer resposta do servidor ao pedido. Este foi um dos temas estudados como uma possível solução para o problema que não chegou a ser implementado.

Capítulo 3

Caso de Estudo

Antigamente, as interfaces de interação com o utilizador eram bastante mais simples, mais pobres no ponto de vista estético e muito pouco *user friendly*. Com o passar dos anos, com o aumento das exigências dos clientes, e com o aumento da tecnologia, as operadoras viram-se obrigadas a evoluir também no visual do seu produto. Devido a estes requisitos também o Frontend e o Backend em si sofreram alterações neste tipo de arquiteturas, onde cada um deles desempenha uma função diferente mas estão interligados entre si, de modo a que a informação que seja necessário transmitir entre cada um, seja traduzida para que o outro consiga interpretar e gerar uma resposta adequada.

Dado que a dissertação foi realizada em parceria com uma operadora, todos os dados fornecidos são provenientes dos servidores da empresa. Estes dados são pedidos em tempo real do WebService que são escritos em forma de *logs* em ficheiros de texto e guardados nos servidores. Estes dados necessitam de ser limpos e tratados de modo a que possam vir a ser inseridos numa base de dados para que a sua leitura e interpretação seja feita mais facilmente, bem como para utilizar estes dados para descobrir erros ou padrões, como é o caso deste trabalho.

O maior desafio neste projeto foi desenvolver um algoritmo que consiga interligar pedidos do *frontend* com respostas do *backend* quando não existe nenhum campo de identificação dos pedidos que seja comum entre os dois. Foi necessário estudar algumas estratégias e perceber como funcionava a arquitetura instalada para que se conseguisse chegar a um método eficaz para relacionar os pedidos com as respostas.

Foi feito um estudo estatístico que nos permite identificar quando e que pedidos têm um atraso significativo e se esse atraso é causado por algo intrínseco, quer seja por fraca otimização de uma função/método, ou por alguma versão de *set-top-box* que tenha um mau funcionamento, ou ate mesmo por algum servidor que esteja com alguma sobrecarga etc.

3.1 Arquitetura estudada

A arquitetura apresentada nesta dissertação é um modelo simples da arquitetura geral e pode ser representada em 3 grupos distintos, sendo eles o conjunto das boxes dos clientes, o conjunto de servidores de *frontend* e o conjunto de servidores de *backend*. A Figura 3.1 mostra como estes diferentes conjuntos estão interligados de modo a que a transmissão de informação seja feita entre eles.

Neste projecto apenas consideramos os pedidos das STB de pacotes de um dos modelos disponibilizados pela operadora ao cliente, e por isso iremos trabalhar maioritariamente com dados do servidor de *frontend* que atende aos pedidos das STB de interesse e de *backend Authentication, Authorization, Accounting, and Auditing* (AAAA), como tal, alguns dos restantes servidores presentes na Figura 3.1 são uma tentativa mais real de representação da arquitetura, mas não serão relevantes para o trabalho realizado.

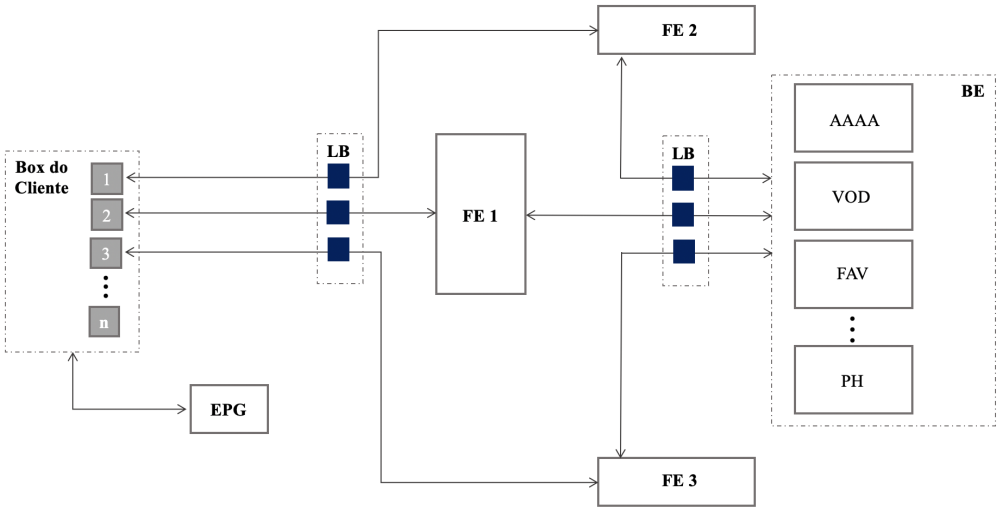

Figura 3.1: Versão simplificada da arquitetura de IPTv do operador.

Quando o utilizador efetua um pedido à box esta irá transmitir os detalhes do pedido ao FE para que possa ser gerada uma resposta adequada. Para que o pedido seja transmitido aos servidores de *FE* é necessário que passe por um *Load Balancer (LB)* para que este faça a gestão do tráfego e reencaminhe os pedidos a um servidor menos congestionado de modo a que se consiga garantir que o tráfego flui normalmente sem atrasos e para evitar a sobrecarga dos servidores. Caso seja necessário, o *FE* traduz o pedido para que seja efetuada a comunicação com o *BE* correspondente que irá gerar uma resposta e enviar de volta ao *FE* e subsequentemente à box do cliente que efetuou o pedido. O *LB* desempenha a mesma função que na comunicação da box para o *FE*. No caso de o utilizador querer aceder ao GuiaTv este envia um pedido específico diretamente ao servidor EPG sem a necessidade de passar pelo *FE*.

3.1.1 *Set-Top-Box (STB)*

Existem aproximadamente 1.5 mil milhões de aparelhos de televisão e cerca de 400 milhões de computadores em casas por todo o mundo. Uma *STB* convencional é um dispositivo que permite melhorar a experiência do utilizador incorporando nela algumas funcionalidades dos computadores, possibilitando que a TV receba e descodifique transmissões de IPTv [21].

Uma *STB* é um dispositivo existente na maioria das casas com subscrição de serviço de TV/Internet que geralmente contem uma entrada de sintonizador de TV que emite a saída para um aparelho de televisão e recebe um sinal de uma fonte exterior, transformando-o em conteúdo que possa ser exibido na tela de uma televisão ou qualquer outro dispositivo de exibição. São utilizadas em televisão por satélite, televisão por cabo, sistemas de televisão sem fios entre outros. Simplificando, é idêntico a um computador que se conecta a televisão e permite que seja utilizada uma linha telefónica ou uma conexão por cabo para navegar na Internet, através do uso de um comando remoto.

É nas STBs que são gerados os pedidos efetuados pelos utilizadores. É através deste dispositivo que o utilizador irá comunicar com o servidor maioritariamente através de um comando remoto, para que o servidor saiba qual a resposta que terá que efetuar. Este tem sido um dos dispositivos que tem sido alvo de maior atenção por parte das operadoras visto que a evolução tecnológica e a exigência do cliente tem obrigado a melhorias constantes na qualidade de experiência.

Os pedidos dos utilizadores são pedidos HTTP efetuados através das STB que procedem à chamada de uma API ao FE com os inputs necessários para que esta informação seja traduzida e transmitida aos servidores de BE, caso seja necessário.

3.1.2 Servidores *frontend* e *backend*

O *frontend* é o servidor de primeiro contacto com o utilizador. Este é responsável por receber os pedidos diretos do utilizador, invocando *APIs* que são a interface na qual o utilizador irá interagir.

Os servidores de *frontend* fornecem uma fonte única para pedidos de conteúdo, de modo a que os sistemas dos utilizadores não tenham que alterar os pedidos ou limpar a *cache* local quando o conteúdo é transferido de um servidor *backend* para outro. O FE também fornecem um nível de segurança adicional para os servidores de *backend* [22].

Após receberem o pedido, os servidores de *frontend* fazem a verificação da validade do pedido e examinam, através de um mapa global que faz o mapeamento dinâmico dos pedidos dos utilizadores com o/os respetivo/os servidor/es de *backend* que armazenam a informação. Em algumas circunstâncias o conteúdo vai estar armazenado num único servidor e o *frontend* fará o pedido diretamente a esse servidor. Por outro lado, se houver mais de um, pode ser gerada um lista de servidores *backend* onde essa informação está armazenada. Para este último caso, o *frontend* vai usar um *token* de autenticação como

chave para uma operação *hash* na lista para identificar qual o servidor capaz de fornecer o conteúdo que é pedido [22].

Neste caso, as estruturas das tabelas são baseadas nos campos formados através do *parsing* dos pedidos efetuados pelos FE. Por sua vez, cada campo pode ser utilizado para efetuar correlações das quais possam surgir padrões interessantes que devam ser observados com o objetivo de perceber de que modo essa relação está a afetar o serviço e assim reduzir os tempos de resposta para garantir a melhor qualidade de experiência aos utilizadores. Para que se entenda ao que cada campo se refere, far-se-á uma breve explicação de alguns campos mais relevantes que serão utilizados no trabalho.

- ***timestamp_fe*** – Data e hora a que o pedido foi efetuado ao FE.
- ***duration*** – Duração do pedido efetuado desde que sai do FE até que chega novamente proveniente de qualquer outro servidor de BE ou Base de Dados;
- ***mac_stb*** – Endereço *MAC* da box que efetuou o pedido ao FE;
- ***function_fe*** – nome da função a ser executada referente ao pedido efetuado pelo utilizador;
- ***hostname*** – Identificador da máquina na qual foi processado o pedido;
- ***stb_version*** – Versão da box que efetuou o pedido;
- ***httprequest_uri*** – URI do pedido HTTP efetuado pelo FE;

À semelhança do que foi feito acima com o FE, será feita uma breve explicação dos campos mais relevantes utilizados na parte do BE.

- ***timestamp_be*** – data e hora a que o pedido foi efetuado ao BE;
- ***hostname*** – Identificador da máquina na qual está a ser processado o pedido;
- ***method*** – Indica qual o método a ser executado relativamente ao pedido efetuado;
- ***isException*** – *Flag* que determina se existe uma excepção/erro;
- ***time_ms*** – Duração do pedido efetuado, em milissegundos.

Antes de iniciar a correlação dos dados, foi necessário estudar todas as vertentes da arquitetura explicadas acima de modo a que se chegasse a um caminho concreto para a resolução do problema proposto.

Inicialmente foi necessário estudar e investigar a arquitetura em torno da inserção dos dados em cada campo, como por exemplo o servidor/plataforma de origem, o mecanismo através do qual os dados são inseridos e o motivo pelo qual os dados são inseridos para que, com isto se entenda bem o funcionamento da arquitetura de modo a que se consiga facilmente detetar uma situação anómala e posteriormente criar uma ferramenta para tratar a anomalia.

3.2 Correlação de dados

Para fazer uma correta correlação dos dados de *frontend* com os de *backend* é impreterível que se entenda como são enviados os pedidos no FE como são geradas as respostas no BE.

O FE guarda a data e hora em que cada pedido foi efetuado e a sua respetiva duração, bem como o endereço MAC da STB que fez o pedido. Tendo em conta que, no BE, nem todos os pedidos contêm o endereço MAC da STB, então os campos mencionados anteriormente são insuficientes para que haja uma correlação correta dos pedidos entre FE e BE. Para ultrapassar este problema, foi utilizada uma tabela de correlação entre endereço MAC e conta de cliente para que se consiga descobrir no BE qual a STB que efetuou o pedido no FE quando esse endereço não é identificado. Devido à enorme quantidade de dados é obrigatório haver um pré-processamento para que se reduza o número de possíveis correlações. Para isso, foram criadas as seguintes regras de seleção:

1. A hora da resposta do BE está inserida no intervalo de tempo entre o pedido no FE (timestamp_fe) e o seu final (timestamp_fe + *duration*);

2. O endereço MAC é o mesmo ou a conta de cliente que aparece no campo do BE está associada ao endereço MAC da STB que efetuou o pedido.

Para se entender melhor o funcionamento e a razão das regras de seleção, o esquema da Figura simples (ver figura 3.2) que ilustra esta relação.

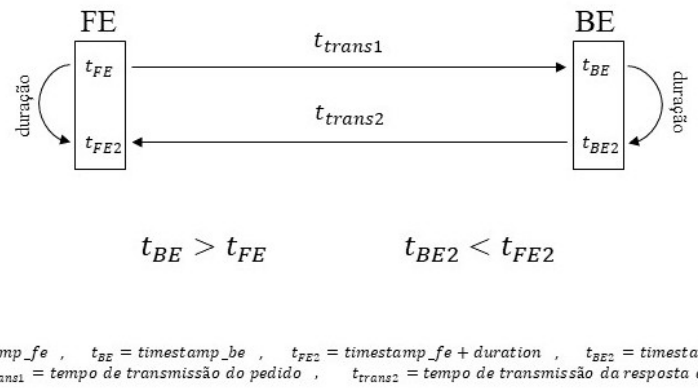

Figura 3.2: Esquema de funcionamento de uma versão simples do modelo de pedido/resposta do *FE* para o *BE* sobre a qual o trabalho foi realizado.

Após submeter os dados a estas regras, o número de potencias respostas aos pedidos que podem ser correlacionados é significativamente menor, como tal será mais fácil descobrir a correlação entre um pedido do FE e a sua respetiva resposta do BE sem que exista a necessidade de existir um ID comum às duas tabelas.

3.3 Metodologia

Os dados em bruto do FE e do BE são armazenados em tempo real nos servidores do operador, em ficheiros com formato *.log* separados minuto a minuto, para cada hora do dia. Como tal, para tratar os dados, foi necessário transferir alguns ficheiros para que fossem utilizados como amostra para a criação do algoritmo. Foi utilizada uma amostra de pedidos contínuos do mesmo dia e da mesma hora com 200000 pedidos do FE e 100000 pedidos do BE.

3.3.1 *Parsing* e Criacao de Tabelas

Antes de iniciar o processo de criação de um algoritmo capaz de identificar e relacionar os pedidos dos diferentes servidores foi necessário separar os dados em bruto e inserilos em tabelas na base de dados para que depois se possa trabalhar sobre essas tabelas. Como se pode observar nas Figuras 3.3 e 3.4 os dados estavam praticamente ilegíveis e só se conseguiria identificar algum parâmetro com algum esforço e pesquisa.

```
2019-06-15T02:52:47.650Z:INFO:3:00D03765B32D:getText:SVLNDIPAP23:ZON_NDS_HDZ_4_9_0:irisfe_ods.logs:sibs:10.144.19.97:http://10.144.19.97/NDS/Appsrv.ashx?
function=getText&textversion=0&boxId=000000000000&bootreason=nominalboot&textId=termsofuse&language=por&version=ZON_NDS_HDZ_4_9_0;15/06/19
```

Figura 3.3: Exemplo dos pedidos em ficheiros *.log* armazenados nos servidores FE.

```
[2019-06-18 10:59:00,011|AA.AccountManager.AccountManager|DEBUG|33][{Realm=AAAA, log4net:HostName=SVLAAIPBE01, TimeMs=9, Method=GetAuthorization, Type=GetAuthorization}]:GetAuthorization(Input:
{entityId:00d03759278b,entityType:device,productId:pvr,productCategory:features,productCategoryTags:NULL}, Return:False,IsException:False,TimeMs:9)||
[2019-06-18 10:59:00,005|AA.AccountManager.AccountManager|DEBUG|56][{log4net:HostName=WSTPAAAA215, TimeMs=6, Realm=AAAA, Type=GetAccount, Method=GetAccount}]:GetAccount(Input:{id:a0c5624b50da,idType:boxId},
Return:641387780Z,IsException:False,TimeMs:6)||
[2019-06-18 10:59:00,011|AA.AccountManager.AccountManager|DEBUG|70][{Realm=AAAA, log4net:HostName=SVLAAIPBE01, TimeMs=7, Method=GetAuthorization, Type=GetAuthorization}]:GetAuthorization(Input:
{entityId:54e2e0543e98,entityType:device,productId:pvr,productCategory:features,productCategoryTags:NULL}, Return:False,IsException:False,TimeMs:7)||
```

Figura 3.4: Exemplo dos pedidos em ficheiros *.log* armazenados nos servidores BE.

Para tornar mais legível e de forma a facilitar a interpretação dos dados, criou-se um algoritmo de *parsing* para separar os dados por campos e inserir nas respetivas tabelas.

Como resultante, surge a Figura 3.5 relativamente ao tratamento dos dados do FE e as Figuras 3.6(a) e 3.6(b) ao do BE. Estas tabela são mais legíveis e de muito mais fácil interpretação. As Figuras 3.5 e 3.6 contém todos os possíveis campos existentes em cada pedido, na subsecção 3.1.2 foram introduzidos e explicados os campos mais importantes e que vão ser utilizados mais à frente para a criação do algoritmo de correlação de dados.

timestamp_fe	severity	duration	mac_stb	function_fe	hostname	stb_version	type	dc	vip	httprequest_uri
2019-07-07 11:38:11.444	INFO	14	7085C6B9D4B7	getBulkEPGImages	SVLNDIPAP50	NDS_HDZ_4_10_0	irisfe_ods.logs	sibs	10.144.19.102	http://10.144.19.102/ND
2019-07-07 11:38:11.718	INFO	3	446AB7943EC4	getChannelApps	SVLNDIPAP51	NDS_HDZ_4_10_0	irisfe_ods.logs	sibs	10.144.19.98	http://10.144.19.98:81/
2019-07-07 11:38:11.756	INFO	12	54E2E028A8F9	getRemoteRecordingIDs	SVLNDIPAP13	NDS_HDZ_4_10_0	irisfe_ods.logs	sibs	10.144.19.102	http://10.144.19.102/ND
2019-07-07 11:38:11.469	INFO	0	D82522C16518	getEpgConfigFile	SVLNDIPAP51	ALL	irisfe_ods.logs	sibs	10.144.19.97	http://10.144.19.97/NDS
2019-07-07 11:38:11.750	INFO	2	7085C6DC41A6	getChannelApps	SVLNDIPAP51	NDS_HDZ_4_10_0	irisfe_ods.logs	sibs	10.144.19.102	http://10.144.19.102/ND
2019-07-07 11:38:11.491	INFO	2	54E2E054E5CF	getLibraryNodeInfo	SVLNDIPAP13	NDS_HDZ_4_10_0	irisfe_ods.logs	sibs	10.144.19.97	http://10.144.19.97/NDS

Figura 3.5: Tabela resultante do *parsing* dos ficheiros do FE após inserção no servidor.

timestamp_be	logger	severity	thread	hostname	time_ms	realm	method	method_type
2019-07-07 17:16:06,632	AA.AccountManager.AccountManager	DEBUG	64	WSTPAAAA215	7	AAAA	GetAccount	GetAccount
2019-07-07 17:16:06,617	AA.AccountManager.AccountManager	DEBUG	76	SVLAAIPBE13	12	AAAA	GetAuthorization, Type=GetAuthorization	GetAuthorization)]:GetAuthorization(Inpu
2019-07-07 17:16:06,648	AA.AccountManager.AccountManager	DEBUG	5	WSTPAAAA215	5	AAAA	GetAccount	GetAccount
2019-07-07 17:16:06,632	AA.PortfolioManager.PortfolioManager	DEBUG	81	SVLAAIPBE13	17	AAAA	GetPortfolio	GetPortfolio

(a) Primeira metade da tabela.

id	id_type	category_id	category_tags	context_tags	product_type	product_hidden	return	is_exception	day_part
44AAF5FDCDBC	boxId	null	null	null	null	null	s839366602	False	2019-07-07
7085C6DC2217	device	null	null	null	null	null	True	False	2019-07-07
S232543001	account	null	null	null	null	null	s232543001	False	2019-07-07
44AAF5C122A8	device	null	System.String[](Count=1)	null	channel	null	AA.Types.Portfolio	False	2019-07-07

(b) Segunda metade da tabela.

Figura 3.6: Exemplo da tabela gerada após o parsing e inserção na base de dados.

Mas, como se observa na Figura 3.6(a), nos campos "*method*" e "*method_type*" existem exemplos de inconsistências, na segunda linha, os campos deveriam conter apenas "getAuthorization" e no entanto têm conteúdo que não tem interesse. Estas inconsistências teriam que ser tratadas antes de se poder passar à criação do algoritmo e da análise estatística.

Isto foi um passo importante porque era um passo necessário que ainda não havia sido implementado. A partir daqui passa a ser possível observar-se facilmente os valores pertencentes a cada campo e abre também a possibilidade de ser feito o tratamento estatístico dos dados.

3.3.2 Limpeza de dados e Criação de *Datasets*

Após criado o código para fazer o *parsing* do ficheiro *.log*, criar as tabelas e fazer a inserção na base de dados da empresa, foi necessário proceder à limpeza e remoção de inconsistências, como a remoção de erros ou falhas na inserção dos dados nas tabelas.

Como é possível observar na Figura 3.7, este é outro exemplo de um dos diversos tipos diferentes de inconsistências encontradas na base de dados que tiveram de ser tratadas e removidas, sendo necessário integrar diferentes dados de diferentes BD Como explicado no Capítulo 1 estes são os primeiros passos no processo de *Data Mining*.

timestamp_be	logger	severity	thread	hostname	time_ms	realm	method	method_type	id	id_type	category_id	category_ta...	context_tags	product_type
null	null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null	null
null			null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null	null
null			null	null	null	null	null	null	null	null	null	null	null	null

Figura 3.7: Exemplo de pedidos inconsistentes existentes nas tabelas logo após a inserção e antes do tratamento dos dados.

Para a remoção deste tipo de inconsistências percorreu-se a base de dados inteira para verificar o que saía fora dos padrões *standard* de cada campo. Removeram-se todos os valores "*null*" existentes em campos onde não deveriam aparecer, foram modificados campos que continham demasiada informação ou informação errada, para que apenas contivessem informação relevante e suficiente. Deste modo, cada campo passou a conter apenas a informação desejada, para não causar dúvidas ou mais à frente, erros na correlação dos dados.

Após a remoção de todas as inconsistências e finalizado o processo de limpeza de dados, procedeu-se à criação de *datasets* consistentes para cada conjunto de dados que iria ser utilizado. Foram selecionados os campos de dados relevantes e consolidados de forma apropriada e obteve-se uma tabela bastante mais perceptível (Figura 3.8) e apenas com os dados necessários que serão utilizados para o passo seguinte, que seria o mais importante deste trabalho.

timestamp_be	time_ms	id	id_type	method
2019-07-07 17:16:06.304	18.0	641269DD04B1	boxId	GetAccount
2019-07-07 17:16:06.291	9.0	98F7D7198F88	boxId	GetAccount
2019-07-07 17:16:06.291	6.0	A0C5624B2D79	boxId	GetAccount
2019-07-07 17:16:06.288	18.0	7085C6AAB849	device	GetAuthorization
2019-07-07 17:16:06.304	18.0	S828093664	account	GetAuthorization

Figura 3.8: Exemplo da tabela final do BE que será utilizada em diante.

3.3.3 Algoritmo de Interligação de pedidos do FE e respostas do BE

Como mencionado anteriormente, segue-se o passo mais importante do projeto, a tarefa que envolveu maior estudo e despendeu mais tempo para chegar a uma solução viável e eficiente, que possa ser implementada em tempo real ou não, para descobrir falhas ou possíveis atrasos nas respostas aos pedidos.

Para criar este algoritmo foram utilizadas 2 tabelas auxiliares que contém dados essenciais para a interligação entre os pedidos, a tabela "mac_sa" (Figura **??**) que contém todos os "mac_address" de todos os dispositivos associados a cada conta de cliente e a tabela "ph_ods" (Figura 3.10) que, entre muitas outras informações, contém informação específica sobre o modelo da STB, que serve para excluir STBs que não se pretende analisar.

Tendo em conta que o maior desafio seria encontrar pedidos e respostas entre um servidor de FE e de um de BE quando não existe um ID comum, foi necessário trabalhar com diferentes dados e tabelas para conseguir chegar a relações entre os pedidos/respostas. Como tal, o algoritmo foi dividido em três objetivos menores, cada um deles imprescindível para sua realização.

mac_stb	mac_cm	sa
001E690D2C83	001E690D2C82	S830625158
001E690D2D8F	001E690D2D8E	S434272401
001E690D311D	001E690D311C	S830374752
001E690D4053	001E690D4052	S830161775
001E690D4B91	001E690D4B90	S825327910

Figura 3.9: Exemplo da tabela auxiliar "mac_cm" que contém todos os *MAC addresses* correspondentes a cada conta de cliente.

mac_cm	state	software_version	hardware_version	serial_number	cas_id	usage_id	mac_stb	model_name	product_class
7085C6DD3715	0	191/4.10_21_17	0003	LA311407138589	18 1548 6760 61	0x0	70:85:c6:dd:37:16	dcr7151/24	dcr7151/24
80F50399558B	0	191/4.10_21_17	0010	FA311448011916	18 1578 6326 33	0x0	80:f5:03:99:55:8c	dcr7151/24	dcr7151/24
2CA17D66E179	0	6.4.69.1.4	1.0	2CA17D66E179	null	null	2c:a1:7d:66:e1:7c	puma6-d7ec	tg2492no
2CA17D664E3D	0	6.4.69.1.4	1.0	2CA17D664E3D	null	null	2c:a1:7d:66:4e:40	puma6-d7ec	tg2492no

(a) Primeira metade da tabela.

technology	bootfile	uptime	last_reconnect	first_connected	extraction_date	device_id	day_part
DOCSIS	null	3	2020-02-27 03:57:18.247	2015-07-07 07:57:49.839	20200227	1028858	2020-02-27
DOCSIS	null	10	2020-02-27 05:38:15.172	2015-07-18 10:49:42.972	20200227	1920648	2020-02-27
Ethernet	null	187	2020-02-27 21:56:52.381	2018-10-15 17:34:48.29	20200227	2964604	2020-02-27
Ethernet	null	340692	2020-02-27 20:40:45.953	2018-10-16 09:51:04.816	20200227	2964807	2020-02-27

(b) Segunda metade da tabela.

Figura 3.10: Exemplo da tabela auxiliar "ph_ods" completa, que contém a informação dos dispositivos de interesse.

Ao inicio foi necessário separar todas as respostas do BE que não contenham um MAC no campo ID, ou seja, que o "ID_type" seja "account", o que significa que o tipo de ID é uma conta de cliente e o ID será o número de conta ao invés de ser o MAC da STB. Este passo é importante porque este tipo de respostas são os que geram maiores dificuldades em correlacionar com os pedidos do FE devido ao facto de o identificador do pedido e da resposta não ser o mesmo. Guardou-se todas as respostas válidas numa *dataframe* para prosseguir com o algoritmo.

mac_stb	mac_cm	sa	timestamp_be	time_ms	id	id_type
0024D1E7AFAD	0024D1E7AFAC	S840724207	2019-07-07 17:16:39.961	14.0	S840724207	account
0024D1E7AFAD	0024D1E7AFAC	S840724207	2019-07-07 17:16:40.055	10.0	S840724207	account
8C5BF075996E	8C5BF075996D	S840724207	2019-07-07 17:16:39.961	14.0	S840724207	account
8C5BF075996E	8C5BF075996D	S840724207	2019-07-07 17:16:40.055	10.0	S840724207	account
0024D1EA3403	0024D1EA3402	S824342713	2019-07-07 19:43:59.180	6.0	S824342713	account

Figura 3.11: Exemplo da tabela utilizada para fazer a junção entre as contas de cliente e todos os possíveis MAC associados.

De seguida, fez-se uma junção entre as respostas anteriormente guardadas e a tabela auxiliar "mac_sa" com o objetivo de retirar todos os possíveis *MAC address* associados a cada conta, para mais tarde poder fazer a verificação (Figura 3.11). Foi criada uma lista para guardar todas estas contas e após ter esta lista criada e todas as variáveis guardadas, procedeu-se à troca dos números das contas de cliente por todos os possíveis *MAC address* associados. Neste momento, existem várias possibilidades de *MAC address* para cada conta de cliente, o próximo objetivo será identificar quais delas podem ser consideradas como possibilidade. O primeiro objetivo deste algoritmo está então concluído, todas as respostas do BE tem *MAC address* na coluna "ID", como mostra a Figura 3.12 e portanto já são comparáveis com os pedidos do FE.

timestamp_be	time_ms	id	id_type	method
2019-07-07 17:16:06.304	18.0	641269DD04B1	boxId	GetAccount
2019-07-07 17:16:06.291	9.0	98F7D7198F88	boxId	GetAccount
2019-07-07 17:16:06.291	6.0	A0C5624B2D79	boxId	GetAccount
2019-07-07 17:16:06.288	18.0	7085C6AAB849	device	GetAuthorization
2019-07-07 17:16:06.304	18.0	FC6FB7089ACD	account	GetAuthorization

Figura 3.12: Exemplo da tabela final do BE após a troca do número de conta pelo MAC. Esta será a tabela utilizada em diante.

O segundo objetivo consiste em ignorar/remover todas as STB que não pertencem ao modelo a estudar. Para isto foi necessário utilizar mais uma tabela adicional, que contém a versão da STB que, sabendo previamente qual o modelo de STB que está designado a pacotes de interesse, é possível fazer uma seleção das contas que se podem utilizar e com isso reduzir a base de dados. Aqui fez-se uma comparação dos *MAC addresses* da lista anteriormente criada e mencionada no passo anterior com os da tabela "ph_ods", em que se verifica o modelo da STB através do MAC do dispositivo. Caso o modelo seja "dcr7151/24" ou "dcr8151/24" são dispositivos de interesse, caso contrario serão descartados.

Na Figura 3.13 podemos observar que das 100000 respostas iniciais, já reduzimos em quase metade do número inicial, tendo agora apenas 55774 respostas.

```
timestamp_be    55774 non-null datetime64[ns]
time_ms         55774 non-null float64
id              55774 non-null object
id_type         55774 non-null object
method          55774 non-null object
```

Figura 3.13: Número de respostas no BE restantes após aplicar as regras de seleção.

Estes dois passos de pré-processamento dos dados da tabela "df_be" resultam no conjunto de dados a que são aplicadas as regras de seleção anteriormente mencionadas na Secção 3.2.

O passo final do pré-processamento de dados consiste na criação de uma nova *data-frame* com todos os possíveis pedidos que tenham o *MAC address* em comum, chamada "final_output" que irá conter todos os dados encontrados que reúnam as condições necessárias para serem considerados resposta do BE ao pedido do FE.

Antes de serem submetidas às regras de seleção, todas as possíveis respostas são guardadas, até chegar a um único candidato. Este processo permite obter facilmente todos os possíveis candidatos e tê-los guardados no mesmo lugar, fazendo uma junção entre a *dataframe* do FE e do BE.

timestamp_fe	duration	mac_stb	function	hostname	stb_version	timestamp_be	time_ms	id	id_type	method
2019-07-07 17:15:00.087	2.0	FC8E7E984E7C	getBulkEPGImages	SVLNDIPAP45	ZON_NDS_PVR_4_9_0	2019-07-07 17:18:49.328	12.0	FC8E7E984E7C	account	GetPortfolio
2019-07-07 17:15:25.047	0.0	FC8E7E984E7C	getEPGItemInfo	SVLNDIPAP45	ZON_NDS_PVR_4_9_0	2019-07-07 17:18:49.328	12.0	FC8E7E984E7C	account	GetPortfolio
2019-07-07 17:15:00.252	1.0	00D037C197F6	notifyAssetView	SVLNDIPAP32	ZON_NDS_HDZ_4_10_0	2019-07-07 17:16:58.614	10.0	00D037C197F6	boxId	GetAccount
2019-07-07 17:16:11.329	1.0	00D037C197F6	getEPGRating	SVLNDIPAP14	ZON_NDS_HDZ_4_10_0	2019-07-07 17:16:58.614	10.0	00D037C197F6	boxId	GetAccount
2019-07-07 17:16:36.318	1.0	00D037C197F6	notifyAssetView	SVLNDIPAP32	ZON_NDS_HDZ_4_10_0	2019-07-07 17:16:58.614	10.0	00D037C197F6	boxId	GetAccount

Figura 3.14: Exemplo da tabela com todos os possíveis pares de pedidos e os respetivos candidatos a resposta.

Finalmente, já existem condições reunidas para fazer o emparelhamento dos pedidos com as respostas, que será o terceiro e último objetivo deste algoritmo. Para ligar os pedidos com as respostas iremos utilizar a lógica apresentada no esquema da Figura 3.2.

Primeiro foi necessário traduzir os *timestamps* do FE e do BE para segundos para ser possível fazer comparações entre eles. Depois de fazer as comparações mencionadas na secção 3.2 adicionou-se uma coluna chamada "diff" à tabela a ser utilizada (designada de "final_output") onde é inserida a diferença de tempo entre o pedido no FE e a resposta no BE como se observa na figura 3.15. Esta diferença tem de ser um valor positivo, o que significa que a resposta tem de ter ocorrido mais tarde que o pedido, para este ser considerado válido.

Por fim, ordenou-se cada par de correspondências e através da menor diferença de tempo escolheu-se o candidato mais provável como resposta para o devido pedido.

2019-07-07 17:15:11.457	54E2E02899EA	setBookmarks	SVLNDIPAP24	ZON_NDS_HDZ_4_10_0	113.0	2019-07-07 17:16:13.186	6.0	54E2E02899EA	account	GetPortfolio	61.729
2019-07-07 17:15:11.646	00D037F2DB65	getConfigurations	SVLNDIPAP51	ZON_NDS_HDZ_4_10_0	143.0	2019-07-07 17:16:58.661	5.0	00D037F2DB65	boxId	GetAccount	107.015
2019-07-07 17:15:14.158	00D037520E5A	getFavorites	SVLNDIPAP01	ZON_NDS_HDZ_4_10_0	1621.0	2019-07-07 17:18:16.119	8.0	00D037520E5A	account	GetPortfolio	181.961
2019-07-07 17:15:14.158	00D037520E5A	getFavorites	SVLNDIPAP01	ZON_NDS_HDZ_4_10_0	1621.0	2019-07-07 17:18:16.213	13.0	00D037520E5A	account	GetPortfolio	182.055
2019-07-07 17:15:14.158	00D037520E5A	getFavorites	SVLNDIPAP01	ZON_NDS_HDZ_4_10_0	1621.0	2019-07-07 17:18:18.791	7.0	00D037520E5A	boxId	GetAccount	184.633
2019-07-07 17:15:24.220	7085C630FCB2	setBookmarks	SVLNDIPAP31	ZON_NDS_HDZ_4_10_0	95.0	2019-07-07 17:16:34.679	9.0	7085C630FCB2	device	GetAuthorization	70.459

Figura 3.15: Exemplo da tabela final com as diferenças entre o *timestamp* do pedido e de todos as respostas candidatas.

2019-07-07 17:15:11.457	54E2E02899EA	setBookmarks	SVLNDIPAP24	ZON_NDS_HDZ_4_10_0	113.0	2019-07-07 17:16:13.186	6.0	54E2E02899EA	account	GetPortfolio	61.729
2019-07-07 17:15:11.646	00D037F2DB65	getConfigurations	SVLNDIPAP51	ZON_NDS_HDZ_4_10_0	143.0	2019-07-07 17:16:58.661	5.0	00D037F2DB65	boxId	GetAccount	107.015
2019-07-07 17:15:14.158	00D037520E5A	getFavorites	SVLNDIPAP01	ZON_NDS_HDZ_4_10_0	1621.0	2019-07-07 17:18:16.119	8.0	00D037520E5A	account	GetPortfolio	181.961
2019-07-07 17:15:24.220	7085C630FCB2	setBookmarks	SVLNDIPAP31	ZON_NDS_HDZ_4_10_0	95.0	2019-07-07 17:16:34.679	9.0	7085C630FCB2	device	GetAuthorization	70.459
2019-07-07 17:15:30.049	00D0374FA76C	setBookmarks	SVLNDIPAP30	ZON_NDS_HDZ_4_10_0	162.0	2019-07-07 17:16:54.540	9.0	00D0374FA76C	account	GetPortfolio	84.491

Figura 3.16: Exemplo da tabela final com as diferenças entre o *timestamp* do pedido e da resposta escolhida como candidato final.

Como se pode reparar, na Figura 3.15 existem 3 respostas possíveis diferentes para o pedido cujo "mac_stb" é "00D037520E5A", mais à frente, após aplicar as últimas regras de seleção, na Figura 3.16 já se observa que existe apenas uma única resposta para esse pedido, como tal, verifica-se que o algoritmo foi terminado com sucesso.

Este algoritmo foi criado para ser versátil o suficiente para ser usado para todos os dados que possam aparecer provenientes destes servidores de FE e BE. É um algoritmo automatizado e optimizado para que demore o mínimo tempo possível para bases de dados grandes.

Dá-se por finalizado com sucesso o objetivo mais importante do trabalho, a correlação dos pedidos do FE com as respostas do BE a partir dos dados dos ficheiros *.log* e far-se-á uma análise estatística no capítulo seguinte afim de descobrir algumas correlações que possam ter interesse com o objetivo de minimizar o tempo de resposta aos pedidos ou diminuir o número de erros.

Capítulo 4

Análise e visualização de resultados

Após a implementação do algoritmo de pré-processamento, procedeu-se a realização de código para analisar os dados e extrair conclusões acerca dos atrasos dos pedidos e das respostas. Esta análise foi feita em todos os dados brutos iniciais antes de qualquer tipo de tratamento. Faz sentido que seja feita a análise para os dados na íntegra, mesmo não tendo encontrado correspondências para todos os pedidos, porque esses dados fornecem informação valiosa sobre os tempos dos pedidos.

Inicialmente, criaram-se três tabelas divididas em várias colunas:

- ***Function*** - Campo que identifica qual a função inerente ao pedido;
- **Total** – Total de pedidos de cada função específica;
- **Min** – Menor duração de pedido encontrada na base de dados;
- **Avg** – Tempo médio da duração dos pedidos desta função;
- **p25** – Valor do percentil 25+ para o tempo da duração do pedido;
- **p50** – Valor do percentil 50 para o tempo da duração do pedido;
- **p75** – Valor do percentil 75 para o tempo da duração do pedido;
- **p90** – Valor do percentil 90 para o tempo da duração do pedido;
- **p95** – Valor do percentil 95 para o tempo da duração do pedido;
- **p99** – Valor do percentil 99 para o tempo da duração do pedido;
- **Max** – Maior duração de pedido encontrada na base de dados.

As duas tabelas restantes, são constituídas pelos campos referidos acima excepto o campo "Function", que foi alterado para "stb_version" na tabela referente à versão da STB e para "Hostname" na tabela referente à máquina onde foi processado o pedido. O objetivo com estas tabelas é tentar perceber quais são os valores de duração admissíveis, para que mais à frente se faça uma análise acerca de potenciais erros de desenvolvimento. Estas tabelas contém uma filtragem, que consiste em mostrar apenas as linhas que contenham mais de 1000 pedidos. Isto foi feito porque quando o número de amostras é baixo, as conclusões que se pode vir a tirar dessa amostra podem não ser precisas a análise feita poderá estar incoerente.

stb_version	Total	Min	Avg	p25	p50	p75	p90	p95	p99	Max
NDS_HDZ_4_10_0	144903	0.0	32.899692	1.0	8.0	31.0	104.0	157.00	232.00	3490.0
NDS_PVR_4_9_0	19547	0.0	30.197780	1.0	6.0	27.5	99.0	151.00	230.00	3025.0
NDS_NPVR_ON_4_9_0	8143	0.0	33.452413	2.0	10.0	32.0	103.0	157.00	238.58	4639.0
NDS_HDZ_4_9_0	5166	0.0	43.831204	1.0	11.0	29.0	85.0	132.75	229.00	6836.0

Figura 4.1: Exemplo da tabela com os percentis dos tempos que demoraram os pedidos do FE, agrupado por versão da STB.

Na Figura 4.1 observa-se como esperado que apenas existem 4 modelos diferentes de STB que fazem parte do estudo sendo que a maior parte dos pedidos são efetuados pela versão "NDS_HDZ_4_10_0". Podemos ainda verificar que os tempos são muito similares nas diferentes versões para todos os percentis, o que provavelmente significa que, no geral, as versões das STB não têm qualquer tipo de problema intrínseco.

Na Figura 4.2 existem bastantes máquinas diferentes (32 precisamente) no entanto, também se consegue perceber que os valores são todos muito similares para as percentagens, e como tal, também não se consegue tirar nenhuma conclusão antecipada. Da mesma maneira que na tabela anterior, apenas é possível assumir que a probabilidade de não haver nenhum problema intrínseco às máquinas é elevada.

Na Figura 4.3 já se consegue verificar diferenças nos tempos de duração das diferentes funções. Isto é completamente normal porque cada pedido é diferente e como tal, cada função chamada para executar um pedido demora sempre o seu tempo normal de execução, quer seja por ter que carregar imagens, como por ter que esperar pela resposta de um BE de autenticação para avaliar se a conta tem acesso ao que o utilizador pretende, etc.

Estas tabelas são úteis na medida em que é possível observar os valores concretos da distribuição da duração dos pedidos bem como o número total de pedidos de cada função. Consegue-se perceber logo à partida que há pedidos que irão sempre demorar mais pelos valores mínimos e médios, como é o caso do "getConfigurations" que tem um tempo mínimo para esta amostra de 57 milissegundos e um tempo médio de 176 milissegundos enquanto que outros, como por exemplo o "getBookmarks" tem um tempo mínimo de 1 milissegundo e um tempo médio de 12 milissegundos. Os valores mínimos não devem ser

tidos em conta para análises detalhadas devido ao facto desses valores não serem valores normais para qualquer desses pedidos. Mais à frente será explicado como é feita a análise tendo como base esta tabela e os gráficos seguintes.

Hostname	Total	Min	Avg	p25	p50	p75	p90	p95	p99	Max
SVLNDIPAP23	11277	0.0	35.665248	2.0	7.0	36.0	104.0	150.00	258.00	4067.0
SVLNDIPAP29	22882	0.0	38.615287	2.0	7.0	33.0	130.0	208.00	240.00	4216.0
SVLNDIPAP24	12644	0.0	36.413160	1.0	7.0	33.0	113.0	184.00	241.57	3025.0
SVLNDIPAP83	9659	0.0	34.748421	1.0	9.0	32.0	103.0	162.00	253.00	4639.0
LX3PRDIRI003	3138	0.0	27.323136	2.0	11.0	32.0	62.0	94.00	186.00	5930.0
LX3PRDIRI008	3019	0.0	29.528983	1.0	9.0	31.0	71.0	134.00	198.82	5530.0
LX3PRDIRI009	3009	0.0	25.902293	1.0	10.0	31.0	60.0	90.00	192.00	6536.0
SVLNDIPAP32	11498	0.0	37.001826	1.0	7.0	32.0	118.0	191.00	231.00	4648.0
LX3PRDIRI007	2719	0.0	28.036778	1.0	12.0	36.0	70.0	132.20	203.00	472.0
SVLNDIPAP01	11719	0.0	35.043348	1.0	8.0	31.0	112.0	168.00	231.00	2450.0
SVLNDIPAP45	15933	0.0	30.210004	2.0	6.0	27.0	99.0	139.00	235.00	4127.0
LX3PRDIRI010	2784	0.0	27.648707	1.0	10.0	31.0	64.7	102.85	201.85	5947.0
LX3PRDIRI011	2924	0.0	27.274624	1.0	11.0	31.0	65.0	106.85	191.77	5898.0
LX3PRDIRI017	2821	0.0	27.895427	1.0	10.0	30.0	66.0	103.00	201.00	5605.0
LX3PRDIRI013	3102	0.0	26.413604	1.0	10.0	32.0	68.0	105.00	191.00	2498.0
LX3PRDIRI005	2672	0.0	29.454716	1.0	13.0	36.0	68.0	108.45	185.29	6836.0
SVLNDIPAP14	10808	0.0	36.330403	1.0	7.0	29.0	112.0	184.00	250.93	5050.0
LX3PRDIRI015	2855	0.0	26.575832	1.0	10.0	31.0	69.0	124.30	199.92	1080.0
LX3PRDIRI002	2945	0.0	27.789134	1.0	12.0	31.0	64.0	108.60	210.56	4793.0
SVLNDIPAP30	12330	0.0	36.284590	1.0	7.0	34.0	111.0	165.00	229.71	4363.0
LX3PRDIRI001	2794	0.0	28.772727	1.0	10.0	32.0	72.0	128.35	195.14	2585.0
LX3PRDIRI006	3143	0.0	22.459752	1.0	9.0	26.0	57.0	84.00	194.58	1112.0
LX3PRDIRI012	3109	0.0	30.621422	1.0	12.0	33.0	65.0	115.60	212.84	5045.0
LX3PRDIRI004	2935	0.0	23.472913	1.0	10.0	31.0	61.0	90.00	183.00	354.0
LX3PRDIRI014	2975	0.0	24.874958	1.0	10.0	30.0	64.6	94.00	177.52	1970.0
SVLNDIPAP52	2470	0.0	30.465182	2.0	7.0	27.0	102.0	142.00	219.17	2083.0
SVLNDIPAP21	2558	0.0	30.308444	1.0	4.0	30.0	106.0	141.00	229.15	387.0
SVLNDIPAP50	2627	0.0	31.057480	2.0	6.0	29.0	110.0	152.70	239.48	427.0
SVLNDIPAP22	2192	0.0	33.091241	1.0	7.0	32.0	106.0	139.00	238.81	1133.0
SVLNDIPAP51	2249	0.0	31.336149	2.0	8.0	30.0	104.0	142.60	216.56	1287.0
SVLNDIPAP31	2017	0.0	36.293505	2.0	8.0	32.0	109.4	178.40	274.52	905.0
SVLNDIPAP13	2078	0.0	39.398460	1.0	8.0	39.0	123.6	211.15	250.00	1012.0

Figura 4.2: Exemplo da tabela com os percentis dos tempos que demoraram os pedidos do FE, agrupado por máquina onde foi processado o pedido.

A partir destas tabelas foram criados alguns gráficos, sendo que, para cada linha de cada tabela existem 3 gráficos deste tipo que mostram exatamente como se encontra a distribuição dos valores de duração. Através da interpretação de gráficos fica muito mais fácil de retirar alguma informação prévia acerca dos tempos de duração.

Function	Total	Min	Avg	p25	p50	p75	p90	p95	p99	Max
getBulkEPGImages	35057	0.0	9.660724	1.00	2.0	16.0	26.0	36.00	58.00	1022.0
getConfigurations	8285	57.0	176.786119	144.00	172.0	206.0	236.0	255.00	306.16	1287.0
getEPGRating	30758	1.0	22.173711	2.00	2.0	18.0	40.0	211.00	230.00	1035.0
getVodNodeInfo	14814	1.0	23.105643	13.00	18.0	27.0	34.0	43.00	96.00	1749.0
getMenuItems	2810	0.0	44.715658	21.00	38.0	71.0	76.0	78.00	108.00	336.0
notifyAssetView	25405	0.0	1.142452	1.00	1.0	1.0	1.0	2.00	5.00	107.0
getFavorites	1801	5.0	172.358134	10.00	16.0	216.0	232.0	410.00	2601.00	6836.0
getEPGItemInfo	4718	0.0	0.877490	0.00	0.0	0.0	0.0	0.00	26.00	50.0
getChannelApps	9296	1.0	4.796579	2.00	2.0	4.0	9.0	21.00	34.00	118.0
setBookmarks	18938	21.0	96.015313	76.00	94.0	109.0	125.0	146.00	215.00	1666.0
getStartOverItem	2040	6.0	10.532353	9.00	9.0	11.0	12.0	13.00	20.61	317.0
getRemoteRecordingIDs	4543	8.0	28.689412	13.00	16.0	26.0	52.0	84.90	234.58	929.0
getAuthorization	1975	0.0	34.341266	23.00	26.0	32.5	47.0	57.00	104.78	3490.0
inboxGetMessage	1776	4.0	8.048986	7.00	8.0	9.0	10.0	10.00	12.00	189.0
getUserRights	1314	20.0	129.487062	93.00	114.0	159.0	192.0	210.00	254.74	1188.0
getBookmarks	3918	1.0	12.518632	7.00	8.0	9.0	16.0	48.15	97.00	214.0
getLibraryNodeInfo	3716	1.0	41.620291	12.75	33.0	51.0	77.0	120.00	207.85	1011.0
getBulkVodImages	1857	0.0	14.275714	1.00	2.0	15.0	32.0	61.20	183.16	516.0
PFloadProfile	1016	9.0	18.996063	12.00	15.0	18.0	20.0	23.00	33.00	3025.0

Figura 4.3: Exemplo da tabela com os percentis dos tempos que demoraram os pedidos do FE, agrupado pelo nome da função.

Na Figura 4.4 só é possível observar que a grande maioria dos pontos estão concentrados nos valores de duração mais baixos, mas não se consegue distinguir ao certo quantos pontos existem nos valores de duração mais elevados. É um gráfico útil para se observar e tentar tirar conclusões rápidas sem muito detalhe. Caso houvesse muitos pontos noutra zona do gráfico então provavelmente seria melhor observar com mais detalhe esta função específica.

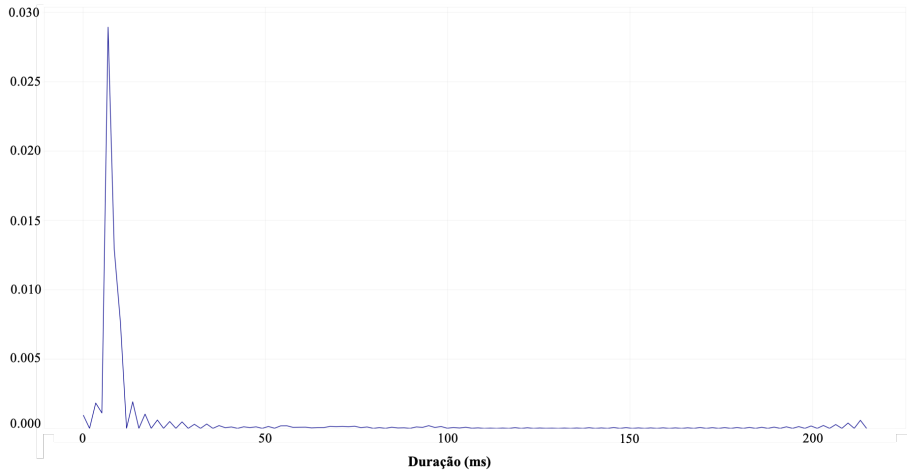

Figura 4.4: Gráfico da densidade da distribuição da duração dos pedidos para a função "getBookmarks".

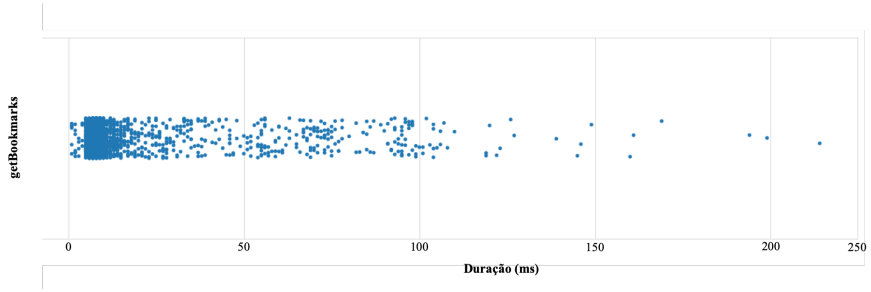

Figura 4.5: Gráfico da distribuição concreta da duração dos pedidos para a função "getBookmarks".

No gráfico apresentado na Figura 4.5 já se consegue observar a distribuição ponto a ponto da duração de cada pedido relativo à função "getBookmarks". Já é possível ver mais detalhadamente onde se encontram os pontos e retirar algumas conclusões extra. Como por exemplo, percebe-se que a partir dos valores a rondar os 100 já começam a haver pontos muitos distantes e mais sozinhos o que pode significar que a partir desse valor já são pedidos que demoraram mais tempo que o previsto.

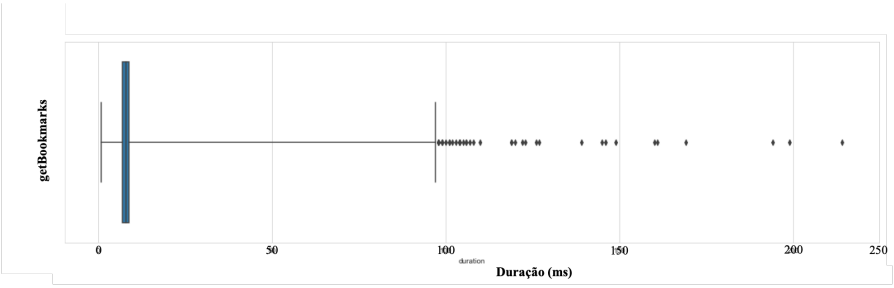

Figura 4.6: Boxplot da distribuição da duração dos pedidos para a função "getBookmarks".

Para ter uma melhor percepção desta distribuição temos ainda o gráfico da Figura 4.6 que mostra um *boxplot* onde se pode observar os valores dentro dos quartis 1 e 3, ou seja o percentil 25, 75 e o percentil 99 e mostra também os *outliers* que são os valores superiores ao valor correspondente ao percentil 99, ou seja os valores afastados do máximo considerado como aceitável.

Para que se entenda melhor as possíveis diferenças entre os gráficos de cada função, vai ser analisada também a função "getBookmarks" da mesma forma que a anterior.

Ao contrário do que se observou na Figura 4.4, na Figura 4.7 observam-se dois picos de densidade o que nos pode levar a pensar que algo está errado com este pedido, mas como este gráfico não é suficiente para uma análise detalhada, temos que observar o gráfico da distribuição da duração para tentar perceber melhor o funcionamento da mesma.

Ao observar a Figura 4.8 entende-se a razão de existirem dois picos de densidade no gráfico. A distribuição dos pedidos está condensada em duas zonas distintas (uma na casa dos 20/30ms e outra na casa dos 60/70ms. Como explicado na análise anterior vamos observar o *boxplot* para entender se esta distribuição está correta.

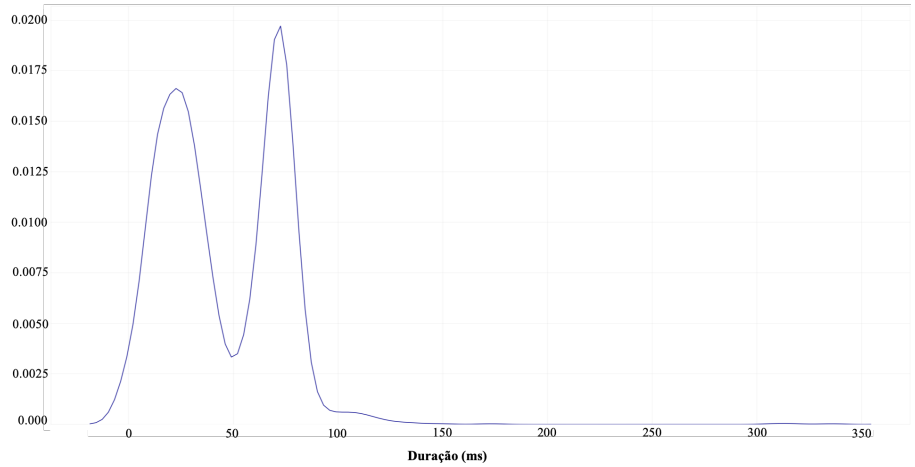

Figura 4.7: Gráfico da densidade da distribuição da duração dos pedidos para a função "getMenuItems".

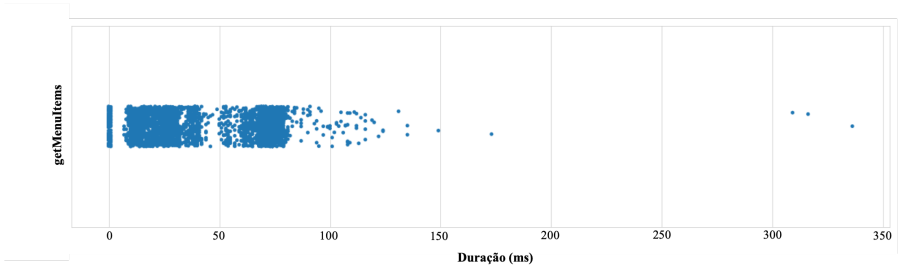

Figura 4.8: Gráfico da distribuição concreta da duração dos pedidos para a função "getMenuItems".

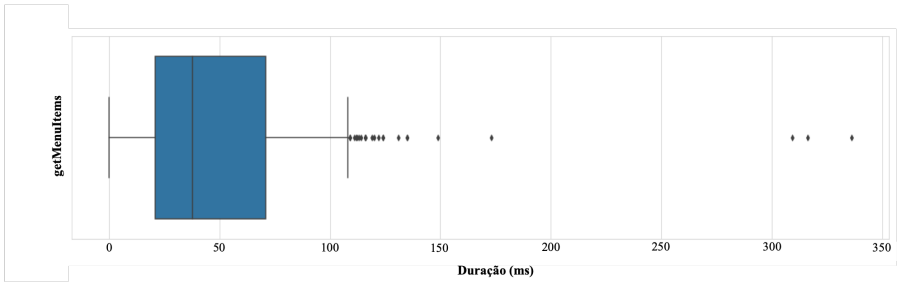

Figura 4.9: Boxplot da distribuição da duração dos pedidos para a função "getMenuItems".

Apesar de os pontos na Figura 4.8 estarem concentrados em duas zonas distintas, ao observar a Figura 4.9, vemos que estes estão inseridos dentro da zona dos valores normais definidos, pois esses valores estão inseridos entre o 0 e o valor do percentil 99 que neste caso é 108. Esta última análise podia ser feita apenas interpretando os valores da tabela da Figura 4.3, mas torna-se muito mais fácil e rápido observando o gráfico.

Do mesmo modo que foi feita a análise para cada função, também foi feita a análise para cada modelo de STB através da tabela apresentada na Figura 4.1 e para cada máquina onde é processado o pedido no *datacenter* através da tabela apresentada na Figura 4.2.

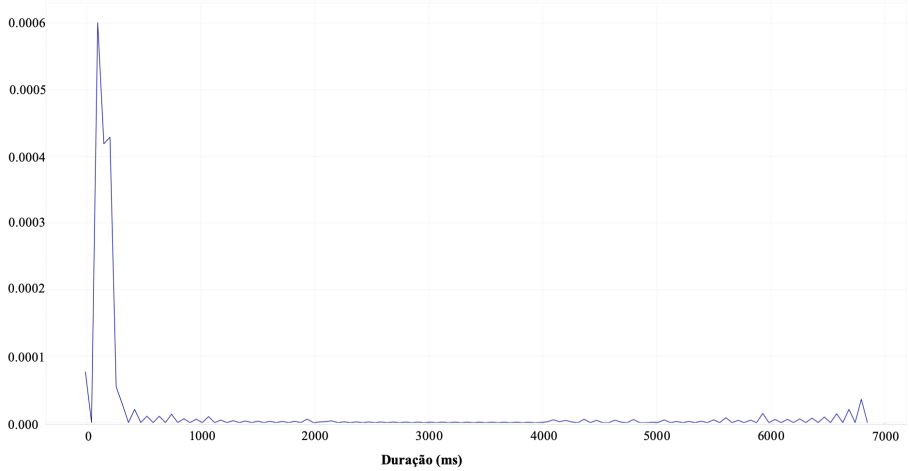

Figura 4.10: Gráfico da densidade da distribuição da duração dos pedidos para o modelo "NDS_HDZ_4_9_0".

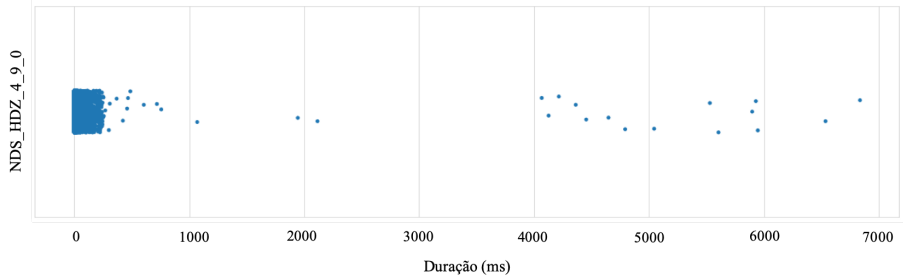

Figura 4.11: Gráfico da distribuição concreta da duração dos pedidos para o modelo "NDS_HDZ_4_9_0"

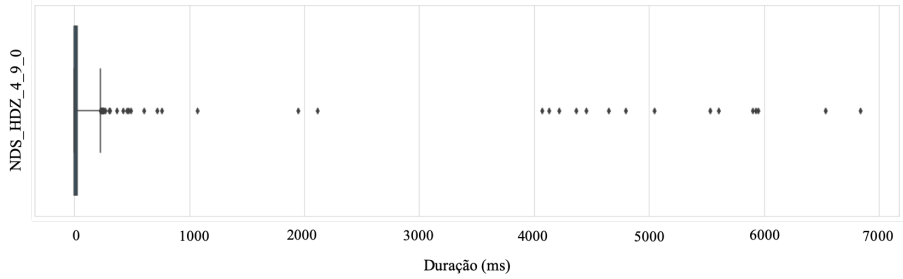

Figura 4.12: Boxplot da distribuição da duração dos pedidos para para o modelo "NDS_HDZ_4_9_0"

A única conclusão que se pode retirar das Figuras 4.10, 4.11 e 4.12 é que para este modelo não existe nenhum problema significativo aparente com este modelo de STB. Os gráficos mostram precisamente que os pontos estão praticamente todos contidos dentro

do que são os valores normais, existem apenas alguns pontos dispersos nos valores mais altos, da mesma forma que acontece com os outros 3 gráficos relativos às STBs restantes e como tal, a única conclusão possível de retirar nesta análise é que os modelos das STBs, à partida não tem nenhum problema significativo.

Seguidamente ir-se-á analisar da mesma forma os gráficos relativos a uma das máquinas onde são processados os pedidos para tentar retirar algumas conclusões prévias.

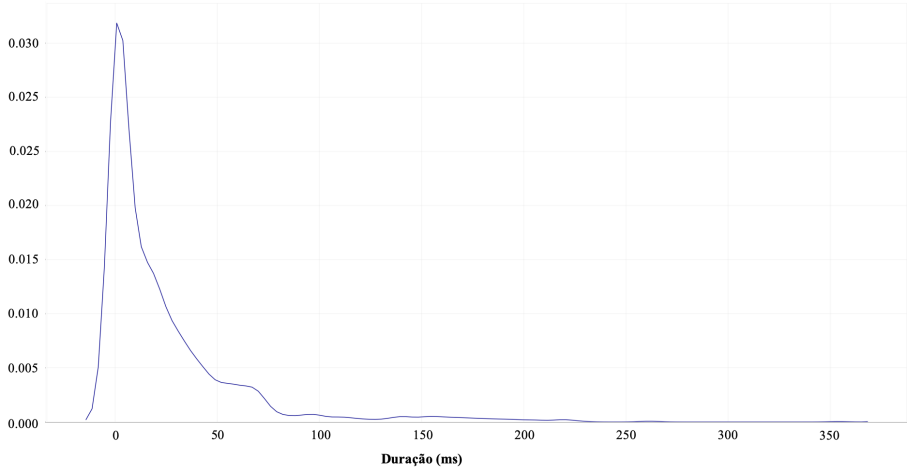

Figura 4.13: Boxplot da distribuição da duração dos pedidos para a máquina "LX3PRDIRI004".

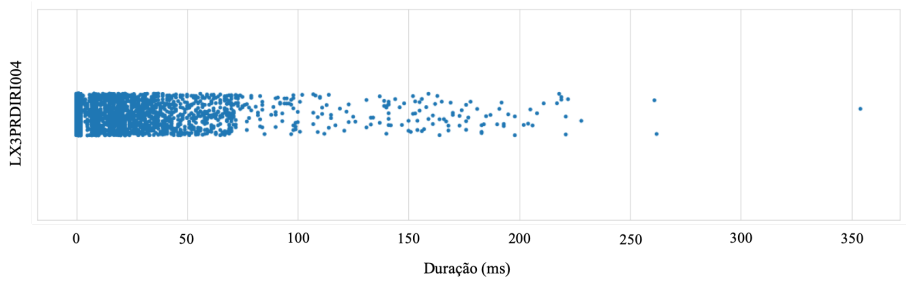

Figura 4.14: Gráfico da distribuição concreta da duração dos pedidos para a máquina "LX3PRDIRI004"

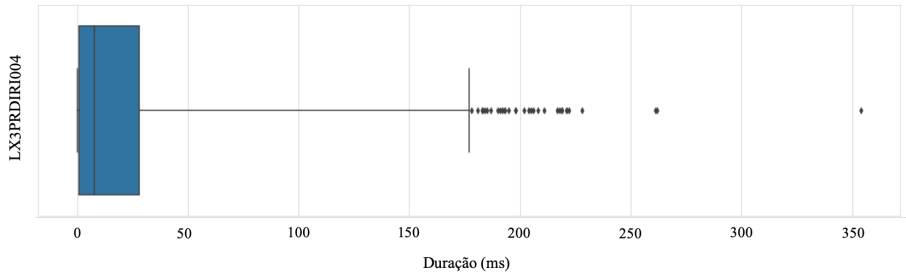

Figura 4.15: Boxplot da distribuição da duração dos pedidos para a máquina "LX3PRDIRI004".

Analisando os gráficos das máquinas (Figuras 4.13, 4.14 e 4.15), a conclusão que se retira para já é a mesma que na análise anterior. Aparentemente não existe nenhum problema intrínseco às máquinas.

De acordo com os percentis das tabelas e com a análise feita em todos os gráficos, criou-se uma lista com os pedidos que demoraram mais tempo que o esperado. Para decidir o *treshold* do tempo para cada tipo de pedido, fez-se uma análise das tabelas dos percentis (Figuras 4.1, 4.2 e 4.3) e dos gráficos e observando todos os valores, foi decidido que o melhor valor de *treshold* seria 2.5 vezes o valor da duração média do tempo de cada função.

Após feita esta análise acerca do valor de *treshold*, que foi útil para retirar uma prévia conclusão geral acerca do funcionamento dos modelos das STB e das máquinas onde os pedidos são processados, fez-se outra análise acerca da quantidade de pedidos efetuados com resposta encontrada no BE que ultrapassaram o tempo máximo considerado como aceitável. Criou-se uma tabela (Figura 4.16) para armazenar essa informação com o intuito de fazer outros tipos de análise posteriores. Neste caso foram utilizados apenas os pedidos que tiveram respostas com sucesso visto que estes seriam os que nos interessavam analisar a partir daqui.

duration	function	hostname	id	id_type	mac_stb	method	stb_version	time_ms
26.0	getBulkEPGImages	SVLNDIPAP14	00D037C197F6	boxId	00D037C197F6	GetAccount	NDS_HDZ_4_10_0	10.0
1165.0	getConfigurations	SVLNDIPAP32	446AB792BB4F	account	446AB792BB4F	GetAccount	NDS_HDZ_4_10_0	4.0
212.0	getEPGRating	SVLNDIPAP32	00D037F00E95	boxId	00D037F00E95	GetAccount	NDS_PVR_4_9_0	8.0
217.0	getEPGRating	SVLNDIPAP32	00D037F00E95	boxId	00D037F00E95	GetAccount	NDS_PVR_4_9_0	8.0
224.0	getEPGRating	SVLNDIPAP32	00D037F00E95	boxId	00D037F00E95	GetAccount	NDS_PVR_4_9_0	8.0

Figura 4.16: Exemplo da tabela com todos os possíveis atrasos significativos encontrados para pedidos cuja resposta do BE tenha sido encontrada.

Foram encontrados 42 pedidos com atraso significativo e através da tabela acima, fez-se uma análise detalhada sobre a origem do pedido (qual o modelo da STB, qual a função chamada e qual a máquina que processou o pedido) para tentar perceber se existe alguma correlação entre os atrasos significativos e algum dos três campos mencionados.

Na Figura 4.17 estão representados os 42 pedidos com atrasos significativos encontrados e divididos por função. Consegue facilmente perceber-se que grande parte destes pedidos são provenientes de pedidos com a função "getEPGRating" e como tal vamos analisar com mais detalhe esses 22 pedidos. Para isso, antes iremos verificar também se existe alguma STB que tenha valores muito superiores.

Na Figura 4.18 não é possível retirar qualquer conclusão, apesar de conseguirmos também verificar que grande parte dos pedidos com atraso significativo estão concentrados no modelo "NDS_PVR_4_9_0" e por coincidência, é o mesmo número verificado para as funções anteriores. No entanto, vamos verificar se existe algo nas STBs dos 22 pedidos da função "getEPGRating" que nos possa indicar se existe alguma relação entre alguma delas.

```
getEPGRating             22
getFavorites              6
getBookmarks              3
getVodNodeInfo            3
getLibraryNodeInfo        1
getChannelApps            1
setBookmarks              1
notifyAssetView           1
getConfigurations         1
getAuthorization          1
getBulkEPGImages          1
getRemoteRecordingIDs     1
```

Figura 4.17: Visualização dos pedidos cuja duração foi superior ao valor definido como máximo admissível, agrupados pela função do pedido.

```
NDS_HDZ_4_10_0        22
NDS_NPVR_ON_4_9_0      8
NDS_PVR_4_9_0          7
NDS_HDZ_4_9_0          5
```

Figura 4.18: Visualização dos pedidos cuja duração foi superior ao valor definido como máximo admissível, agrupados pelo modelo da STB que efetuou o pedido.

```
SVLNDIPAP32    15
SVLNDIPAP83     7
SVLNDIPAP23     5
SVLNDIPAP30     4
SVLNDIPAP45     4
SVLNDIPAP01     3
SVLNDIPAP14     2
SVLNDIPAP50     1
SVLNDIPAP29     1
```

Figura 4.19: Visualização dos pedidos cuja duração foi superior ao valor definido como máximo admissível, agrupados pela máquina que processou o pedido.

Da mesma maneira, na Figura 4.19 também não é possivel concluir nada acerca dos atrasos nas máquinas. No entanto, verifica-se que a máquina "SVLNDIPAP32" tem bastantes mais casos que qualquer um dos outros. e por isso também vamos tentar analisar este caso.

```
NDS_HDZ_4_10_0        9
NDS_PVR_4_9_0         7
NDS_NPVR_ON_4_9_0     6
```

Figura 4.20: Visualização e distinção dos 22 pedidos cuja função é "getEPGRating" agrupados por modelo da STB que efetuou o pedido.

Neste momento já conseguimos descobrir algo importante. É possível observar que apenas 7 do total de 42 pedidos são efetuados pelo modelo "NDS_PVR_4_9_0" e dos 22 pedidos "getEPGRating", 7 são efetuados pelo modelo "NDS_PVR_4_9_0". Isto significa que todos os pedidos com erro do modelo "NDS_PVR_4_9_0" são pedidos cuja função é "getEPGRating". Isto pode significar que a função "getEPGRating" está mal otimizada para o modelo de STB em questão ou que esse modelo de STB tem um problema no processamento do pedido "getEPGRating".

```
NDS_PVR_4_9_0          7
NDS_NPVR_ON_4_9_0      6
NDS_HDZ_4_10_0         1
NDS_HDZ_4_9_0          1
```

Figura 4.21: Visualização e distinção dos 15 pedidos cuja máquina onde foram processados os pedidos é "SVLNDIPAP32" agrupados por modelo de STB.

Com outra análise, descobriu-se ainda que todos esses 7 pedidos foram processados pela máquina "SVLNDIPAP32", todos são provenientes do modelo "NDS_PVR_4_9_0" e da mesma forma, todos são pedidos cuja função é "getEPGRating". Isto pode significar ainda que esta máquina está com alguns problemas em processar este tipo de pedidos. É conveniente que a operadora faça alguns testes a este modelo de STB ou faça uma melhor análise acerca desta descoberta com o intuito de confirmar a veracidade destas afirmações.

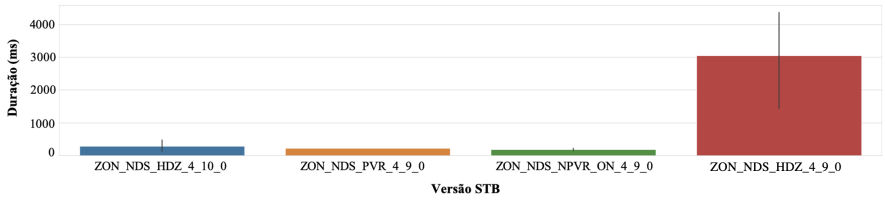

Figura 4.22: Gráfico da diferença da duração entre modelos das STB.

Por último, no gráfico da Figura 4.22 fez-se um gráfico de todos os pedidos na lista de pedidos com atraso significativo, onde mostra a duração média e máxima desses pedidos grupados por modelo da STB. É notório que existe um atraso muito maior nos pedidos provenientes das STB do modelo "NDS_HDZ_4_10_0" em comparação com os outros modelos de STB. Esta análise merece uma melhor observação posterior por parte da operadora de modo a descobrir qual a causa deste acontecimento, pois a discrepância de valor é demasiado grande para ser normal.

Note-se que toda esta análise é apenas uma análise estatística e para comprovar a veracidade desta análise teriam de ser efetuados testes às STBs ou à máquina que fez o processamento do pedido. Neste caso tentámos descobrir correlações para que os desenvolvedores de software possam ter o trabalho facilitado aquando da procura de erros.

Capítulo 5

Conclusão

A secção 5.1 deste capítulo fala-se sobre as principais conclusões retiradas do trabalho desenvolvido na presente dissertação. Para além disso, na secção 5.2, também são sugeridas algumas propostas para trabalhos futuros que possam vir a ser desempenhados, com o intuito de aprofundar e confirmar as análises estatísticas desenvolvidas na presente dissertação.

5.1 Síntese conclusiva

Nos dias que correm, com o avanço tecnológico, as operadoras foram obrigadas a disponibilizar ofertas que incluem, na sua grande maioria, plataformas de acesso a vídeo. Os APIs que são utilizados nestas plataformas são baseados na tecnologia REST e comunicam através de mensagens HTTP para fazer a gestão dos pedidos dos clientes. Estes APIs são invocados pelos FE como plataforma de interação com o cliente que irão traduzir a mensagem do pedido e transmiti-la ao devido BE que irá tratar dessa função para gerar a devida resposta.

Com o aumento da exigência dos clientes e devido à grande competitividade entre as empresas do setor, as operadoras têm de garantir a melhor qualidade de experiência possível para agradar ao consumidor. Para isso, é necessário sistemas em tempo real que consigam analisar grandes quantidades de dados gerados através da interação entre o utilizador e os servidores para que, deste modo, consigam extrair informação relevante para melhorar o serviço prestado, quer em termos de qualidade do conteúdo como em termos da diminuição dos tempos de resposta aos pedidos.

A quantidade de dados proveniente das interações de centenas de milhares de pessoas com os servidores, a todas as horas, em tempo real é enorme e para fazer uma análise eficiente desta quantidade de dados, nasceu o termo "mineração de dados", que se tornou uma àrea de grande importância devido à grande variedade de ferramentas disponíveis para fazer o tratamento dos dados.

Atualmente esta operadora não possui um sistema que faça a ligação dos pedidos do FE com as respostas do BE e que faça uma análise dos dados provenientes destes pedidos e respostas. De modo a preencher esta lacuna, foi desenvolvido um algoritmo que tem a possibilidade de trabalhar em tempo real ou não, que faça a ligação dos pedidos do FE com as respostas do BE, onde não há nenhum campo com o ID comum nas tabelas dos pedidos dos diferentes servidores. Após a implementação deste algoritmo, foi também feita uma análise estatística para descobrir a relação entre os atrasos nos tempos de resposta aos pedidos dos utilizadores, com a finalidade de encontrar possíveis erros nos modelos das STB, nos servidores de processamento ou na implementação e otimização das funções dos pedidos.

O algoritmo tem como base duas regras essenciais para a descoberta da relação entre os pedidos e as respostas. A primeira é que a hora da resposta do BE tem de estar inserido no intervalo de tempo entre o pedido do FE e o seu final e a segunda, que o endereço MAC seja o mesmo no pedido e na resposta. Tendo em conta que o campo de ID da resposta nem sempre apresenta um endereço MAC e grande parte das vezes apresenta uma conta de cliente, o maior desafio seria interligar a conta de cliente com o MAC correto. Este desafio foi concretizado com sucesso assim concluiu-se o principal objetivo do trabalho.

Por fim, analisando os dados extraídos e verificando algumas relações, percebe-se que o modelo "NDS_HDZ_4_10_0" é o que apresenta um valor médio de duração dos pedidos com atraso mais elevado. Por isso, poderá vir a ser relevante fazer alguns testes

neste modelo para verificar se realmente existe um problema intrínseco ao modelo da STB.

Verifica-se ainda que existe um problema em particular com a função "getEPGRating" no modelo "NDS_PVR_4_9_0". Existem algumas possibilidades em relação à causa destes atrasos, nomeadasmente a função pode não estar muito bem otimizada para este modelo da STB ou haver algum problema com o modelo em si ao processar o pedido que contenha esta função.

5.2 Trabalho futuro

No sentido de aprofundar e melhorar o estudo desenvolvido na presente dissertação, sugerem-se algumas diretrizes para trabalhos futuros:

- Executar testes reais nas STB e nas funções para perceber se este estudo e as sua conclusões estão corretas;
- Proceder a um estudo estatístico semelhante para outros modelos de STB e para outros servidores de *frontend* e de *backend*;
- Sabendo os tempos de resposta dos pedidos no FE (para o BE), prever um tempo de resposta de uma STB.

Bibliografia

[1] Q. Zhao e S. S. Bhowmick. "Sequential pattern mining: A survey". Em: *ITechnical Report CAIS Nayang Technological University Singapore* 1 (2003), p. 26.

[2] J. Han, J. Pei e M. Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.

[3] W. Hämäläinen e M. Vinni. "Comparison of machine learning methods for intelligent tutoring systems". Em: *International Conference on Intelligent Tutoring Systems*. Springer. 2006, pp. 525–534.

[4] R. Agrawal e R. Srikant. "Mining sequential patterns". Em: *Data Engineering, 1995. Proceedings of the Eleventh International Conference on*. IEEE. 1995, pp. 3–14.

[5] A. Ceglar e J. F. Roddick. "Association mining". Em: *ACM Computing Surveys (CSUR)* 38.2 (2006), p. 5.

[6] J. Srivastava, R. Cooley, M. Deshpande e P.-N. Tan. "Web usage mining: Discovery and applications of usage patterns from web data". Em: *Acm Sigkdd Explorations Newsletter* 1.2 (2000), pp. 12–23.

[7] N. R. Mabroukeh e C. I. Ezeife. "A taxonomy of sequential pattern mining algorithms". Em: *ACM Computing Surveys (CSUR)* 43.1 (2010), p. 3.

[8] H. Pinto, J. Han, J. Pei, K. Wang, Q. Chen e U. Dayal. "Multi-dimensional sequential pattern mining". Em: *Proceedings of the tenth international conference on Information and knowledge management*. ACM. 2001, pp. 81–88.

[9] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal e M.-C. Hsu. "Mining sequential patterns by pattern-growth: The prefixspan approach". Em: *IEEE Transactions on Knowledge & Data Engineering* 11 (2004), pp. 1424–1440.

[10] C. H. Mooney e J. F. Roddick. "Sequential pattern mining–approaches and algorithms". Em: *ACM Computing Surveys (CSUR)* 45.2 (2013), p. 19.

[11] R. M.N. C. Matos. *Sequential Protocols' Behaviour Analysis, Tese de Mestrado, Faculdade de Ciências e Tecnologia - Universidade Nova de Lisboa*. 2018.

[12] R. Agrawal, T. Imieliński e A. Swami. "Mining association rules between sets of items in large databases". Em: *Acm sigmod record*. Vol. 22. 2. ACM. 1993, pp. 207–216.

[13] J. Han e J. Pei. "Mining frequent patterns by pattern-growth: methodology and implications". Em: *ACM SIGKDD explorations newsletter* 2.2 (2000), pp. 14–20.

[14] J. Han e J. Pei. "Mining Frequent Patterns by Pattern-growth: Methodology and Implications". Em: *SIGKDD Explor. Newsl.* 2.2 (dez. de 2000), pp. 14–20. ISSN: 1931-0145. DOI: 10.1145/380995.381002. URL: http://doi.acm.org/10.1145/380995.381002.

[15] M. J. Zaki. "Efficient enumeration of frequent sequences". Em: *Proceedings of the seventh international conference on Information and knowledge management*. ACM. 1998, pp. 68–75.

[16] J. Ayres, J. Flannick, J. Gehrke e T. Yiu. "Sequential pattern mining using a bitmap representation". Em: *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2002, pp. 429–435.

[17] W. Zucchini, I. L. MacDonald e R. Langrock. *Hidden Markov models for time series: an introduction using R*. Chapman e Hall/CRC, 2016.

[18] R. R. Sarukkai. "Link prediction and path analysis using Markov chains". Em: *Computer Networks* 33.1-6 (2000), pp. 377–386.

[19] S. R. Eddy. "Profile hidden Markov models." Em: *Bioinformatics (Oxford, England)* 14.9 (1998), pp. 755–763.

[20] Z. Ghahramani. "An introduction to hidden Markov models and Bayesian networks". Em: *International journal of pattern recognition and artificial intelligence* 15.01 (2001), pp. 9–42.

[21] J. O'Brien, T. Rodden, M. Rouncefield e J. Hughes. "At home with the technology: an ethnographic study of a set-top-box trial". Em: *ACM Transactions on Computer-Human Interaction (TOCHI)* 6.3 (1999), pp. 282–308.

[22] B. Deen, A. Hopmann e J. Soderberg. *Routing client requests to back-end servers*. US Patent 6,823,391. 2004.