UNIFACS UNIVERSIDADE SALVADOR
MESTRADO ACADÊMICO EM SISTEMAS E COMPUTAÇÃO

SERGIO SAMPAIO SPINOLA

GESTÃO DE CONTEXTO APLICADA AO ENCAMINHAMENTO ADAPTATIVO EM SOLUÇÕES CONVERGENTES

Salvador
2015

SERGIO SAMPAIO SPINOLA

GESTÃO DE CONTEXTO APLICADA AO ENCAMINHAMENTO ADAPTATIVO EM SOLUÇÕES CONVERGENTES

Dissertação apresentada ao curso de mestrado acadêmico em sistemas e computação, da UNIFACS Universidade Salvador, Laureate International Universities como requisito parcial para obtenção do título de Mestre em Sistemas e Computação.

Orientador: Prof. Dr. Paulo Nazareno Maia Sampaio.

Salvador
2015

FICHA CATALOGRÁFICA
Elaborada pelo Sistema de Bibliotecas da UNIFACS Universidade Salvador, Laureate International Universities

Spinola, Sergio Sampaio

Gestão de contexto aplicada ao encaminhamento adaptativo em soluções convergentes. / Sergio Sampaio Spinola. – Salvador, 2015.

179 p.: il.

Dissertação apresentada ao Curso de Mestrado em Sistemas e Computação, UNIFACS Universidade Salvador, Laureate International Universities como requisito parcial para obtenção do grau de Mestre.

Orientador: Prof. Dr. Paulo Nazareno Maia Sampaio.

1. Roteamento Adaptativo. 2. Redes de computadores. 3.Convergência . I. Sampaio, Paulo Nazareno Maia, orient. II. Título.

CDD: 004

SERGIO SAMPAIO SPINOLA

GESTÃO DE CONTEXTO APLICADA AO ENCAMINHAMENTO ADAPTATIVO EM SOLUÇÕES CONVERGENTES

Dissertação, mestrado acadêmico em sistemas e computação, da Universidade Salvador - UNIFACS, para a obtenção do título de Mestre em Sistemas e Computação.

Paulo Nazareno Maia Sampaio – Orientador _____
Doutorado em Informatique et Telecommunications pelo Université Toulouse III Paul Sabatier, França
UNIFACS Universidade Salvador
Laureate International Universities

Joberto S. B. Martins _____
Doutorado em Doutorado em Computação (PhD) pelo Université Pierre et Marie Curie, França
UNIFACS Universidade Salvador
Laureate International Universities

Marcos Rogério Salvador _____
Doutorado em Ciência da Computação pela University Of Twente, Holand
Lenovo Innovation Center

Salvaodr, 16 de dezembro de 2015

AGRADECIMENTOS

Após esse período de dedicação aos estudos e pesquisa, deixo registrado a seguir os meus sinceros agradecimentos:

Primeiramente, quero agradecer aos meus familiares pelo incentivo, e principalmente aos meus pais, a quem devo a base de todas as minhas conquistas. Agradeço especialmente à minha esposa Carolina, grande incentivadora e motor impulsionador, sobretudo pela paciência e estado de espírito que tornaram mais fácil empreender as tarefas. Aos colegas do grupo de pesquisas, Fred, André, Joatham e também Rogerio, pelo espírito de cooperação e pela amizade.

Ao meu orientador, professor Dr. Paulo Sampaio, meus sinceros agradecimentos pelo suporte, diligente trabalho de acompanhamento e auxílio nos momentos cruciais que tornaram possível levar a cabo este trabalho.

À Fapesb pelo apoio oferecido através da concessão de bolsa destinada a alunos de pós-graduação.

RESUMO

Desde o advento da Internet, inúmeros avanços tecnológicos contribuíram para viabilizar diversos tipos de serviços disponibilizados aos usuários de redes, dentre os quais destacam-se as aplicações multimídia de vídeo e voz interativos, em uma escala que não havia sido sequer vislumbrada nos primóridos da era da Internet. Diversos mecanismos de controle para acomodar estes novos usos foram propostos e implementados, a maioria dos quais ainda se vale das extensões dos mecanismos clássicos de QoS que foram estendidos às redes MPLS, assim como técnicas de engenharia de tráfego que muitas vezes são aplicadas estaticamente. A massificação dos aparelhos móveis e portáteis com capacidade multimídia ampliada representa um novo desafio, na medida em que controle da rede através de métodos tradicionais não mais atende à expectativa do usuário, que se traduz em qualidade de experiência. Por outro lado, a quantidade de variáveis com que é preciso lidar cresce continuamente, determinando a necessidade de delimitar e controlar aquilo que é definido como variáveis e eventos de contexto. Além disso, controles efetivos que levem em conta parâmetros tão diversos como a percepção do usuário devem ser contemplados nas soluções para o encaminhamento adaptativo em soluções convergentes. Portanto, os principais objetivos deste trabalho são: a proposta de um arcabouço para o gerenciamento de contexto aplicado ao encaminhamento adaptativo, especificamente na concepção de um módulo de tratamento e gestão de contexto, compreendendo a modelagem e representação do contexto, a modelagem da arquitetura que embasará a implementação da solução proposta e a posterior validação de cenários orientados às aplicações convergentes. De uma forma geral, o presente trabalho se destaca na temática de controle e gerência de tráfego de redes orientado a contexto, contemplando o processo de concepção de cenários de redes, desde a modelagem dos cenários até efetiva experimentação sobre os mesmos.

Palavras-chave: Roteamento Adaptativo. Contexto. Qualidade de Contexto. Qualidade de Experiência. Redes Sensíveis ao Contexto. Convergência.

ABSTRACT

Since the advent of Internet, many technological advances have contributed to enable a great variety of services available to network users, among which multimedia video applications and interactive voice were deployed on a huge scale, in a way never been envisioned in Internet before. Several control mechanisms were proposed and developed in order to support these new applications, most of which carried out through extensions of the classical QoS mechanisms, such as extended MPLS networks controls and traffic engineering techniques, which are mostly deployed statically. The massification of mobile devices and laptops with enhanced multimedia capabilities represent a new challenge, in a way that network controls using traditional methods are no longer capable of meeting users' expectations (translated into quality of experience). In opposite, the number of variables that must be dealt with are continously growing, determining the need to define and control what is understood as events and context variables. In addition, effective controls that take into account various parameters such as user's perception should be implemented by adaptive routing in converging solutions. In this work the main goals are: the proposal of a framework for context management applied to adaptive routing, specifically in the design of a context treatment and management processing module. comprising modeling and context representation; modeling of the architecture that embraces the implementation of the proposed solution and the subsequent validation of the implemented architecture for the traffic management of converging applications oriented scenarios. In general, this work focus on the thematic of context oriented network traffic management and control, covering the design life cycle of network scenarios, from modeling through their effective experimentation.

Keywords: Adaptive Routing. Context. Quality of Context. Quality of Experience. Context Sensitive Networks. Convergence.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTAS DE ABREVIATURAS E SIGLAS

API	*Application Program Interface*
ATM	*Asynchronous Transfer Mode*
BSON	*Binary JSON*
CAARF	*Context-Aware Adaptive Routing Framewok*
CBQ	*Class Based Queue*
CBR	*Constraint Based Routing*
CDR	*Call Detailed Records*
CoS	*Class of Service*
CSCF	*Call Session Control Function*
CSPF	*Constraint Shortest Path First*
EC	*Elementos Contextuais*
ESM	*Experience Sampling Method*
Diffserv	*Differentiated Services*
DS-TE	*Diffserv Traffic Engineering*
GMPLS	*Generalized Multiprotocol Label Switching*
GPS	*Global Positioning Syste*
HTB	*Hierarchical Token Bucket*
IP	*Internet Protocol*
IS-IS	*Intermediate System to Intermediate System*
IMS	*IP Multimedia Subsystem*
JSON	*Java Script Object Notation*
LDAP	*Lightweight Directory Access Protocol*
LMP	*Link Management Protocol*
LSP	*Label Switch Path*
LSR	*Label Swich Router*
MOS	*Mean Opinion Score*
MPLS	*Multiprotocol Label Switching*
MPQM	*Moving Picture Quality Metrics*
NGN	*Next Generation Networks*
NOSQL	*Not Only SQL*

NQM	Noise Quality Metric
OXC	*Optical Cross Connect*
PAAS	*Plataform as a Service*
PBX	*Private Branch Exchange*
PESQ	*Perceptual Evalution Of Speech Quality*
PSNR	*Peak Signal to Noise Ratio*
QOC	*Quality of Context*
QOD	*Quality of Device*
QOE	*Quality of Experience*
QOS	*Quality of Service*
RED	*Random Early Discard*
RFC	*Request for Comments*
RTCP	*Real Time Transport Control Protocol*
RTP	*Real Time Transport Protocol*
RTT	*Round Trip Time*
RSVP	*Resource Reservation Protocol*
RWA	*Routing Wavelength Assignment*
SDN	*Softwre Defined Network*
SDP	*Session Description Protocol*
SDP	*Service Delivery Platform*
SIP	*Session Initiation Protocol*
SLA	*Service Level Agreement*
SLS	*Service Level Specification*
SPB	*Shortest Path Bridging*
SPF	*Shortest Path First*
SSIM	*Structural Similarity Index*
TCP/IP	*Transmission Control Protocol/Internet Protocol*
TE	*Traffic Engineering*
TUQ	*User Testing Perceived QoS*
UAC	*User Agent Client*
UAS	*User Agent Server*
UDP	*User Datagram Protocol*
UNI	*User Network Interface*

VoIP	*Voice over IP*
WDM	*Wavelength Division Multiplexing*
XML	*Extended Markup Language*

SUMÁRIO

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

Nos dias atuais, a utilização de diversos serviços aplicacionais sobre plataformas e infraestruturas de redes baseadas no protocolo IP (*Internet Protocol*), MPLS (*Multiprotocol Label Switching*), GMPLS (*Generalized Multiprotocol Label Switching*) e também nas arquiteturas emergentes, superam amplamente os usos previstos na concepção inicial das primeiras redes baseadas em TCP/IP (*Transport Control Protocol/Internet Protocol*).

Dentre as principais evoluções encontramos a crescente disseminação das aplicações ditas convergentes, ou unificadas, cujo acesso pode ser realizado em qualquer lugar e a qualquer momento. Da mesma forma, com a massificação do uso de aparelhos móveis dotados de grande capacidade de processamento e armazenamento, dotados de conectividade IP, foi viabilizada a utilização de aplicações multimídia sofisticadas nesses aparelhos, o que era impossível há apenas uma década atrás.

Em decorrência destas novas possibilidades, novas questões, como aquelas relacionadas ao trabalhador remoto e móvel, que sequer eram antes consideradas, agora representam desafios importantes para a produtividade dos trabalhadores e empresas.

Neste contexto, o cenário de utilização típico abordado nesta dissertação de mestrado é constituído por usuários e aplicações que requerem tratamento diferenciado para as aplicações multimídia, convergentes, interativas, sensíveis e que apresentam também desafios relacionados à mobilidade.

Atualmente, o controle de sessão, autenticação e gerência dos usuários e aplicações de telefonia IP e videoconferência emprega largamente o protocolo SIP (*Session Initiation Protocol*), seja para aplicações de mensageria instantânea, como para controle de sessão de vídeo e áudio-conferência, assim como a comunicação VoIP (*Voice over IP*) - por exemplo, enquanto extensão do sistema de comunicação empresarial, como ramal remoto SIP (RFC 3261, 2002) de um PBX-IP (*Public Branch Exchange*).

Nestes cenários de utilização, é importante considerar a dinâmica observada em relação aos provedores de serviços de telecomunicações, que possuem investimentos maciços em estruturas baseadas em MPLS. Por outro lado, vários provedores de serviços de telecomunicações também começam a adotar arquiteturas baseadas na separação de plano de

controle e dados, aderentes à arquitetura de Redes Definidas por Software. Necessariamente, múltiplas plataformas deverão coexistir, incluindo as redes corporativas privadas puramente IP, demandando novos métodos de gerência e controle em diferentes níveis de abstração.

1.2 PROBLEMÁTICA

Considerando um cenário de utilização diverso, complexo, dinâmico, sensível, híbrido e distribuído, a qualidade dos serviços prestados ao usuário final, e percebida por este, não pode mais ser sustentada ou garantida plenamente ante o pressuposto da existência de domínios controlados, seja na rede corporativa, através da existência de controles de QoS tradicionais, seja pela Engenharia de Tráfego em redes MPLS.

Na prática, a Engenharia de Tráfego em redes MPLS é implementada de forma estática em muitas situações pelos provedores de serviços, baseados meramente no histórico e previsão de demanda de utilização pelos clientes daqueles serviços. No entanto, a configuração estática de tráfego já não responde adequadamente à dinâmica crescente de uso nos moldes atuais, dada a maior flexibilidade e mobilidade dos usuários e aplicações.

Em paralelo a estas grandes estruturas já existentes, novas arquiteturas vêm confrontando o *status quo* baseado em MPLS e que vêm tomando o seu lugar. Em particular, é possível identificar aquelas que separam o plano de controle do plano de dados, dado que as estruturas baseadas em MPLS não conseguem fornecer respostas completas às novas demandas, requerendo novas abordagens.

Ainda é necessário levar em consideração que os provedores são constantemente confrontados quanto à existência de uma diferença significativa entre a qualidade aferida e medida pelos seus sistemas de gerenciamento. De fato, os sistemas tradicionais de medição de um serviço específico da rede, geralmente baseados em métricas de Qualidade de Serviço (QoS), não são capazes de aferir adequadamente a percepção dos assinantes, levando a uma potencial inconsistência e crônica insatisfação da parte dos usuários/assinantes de serviços de rede.

Portanto, é possível constatar que:

- Mecanismos de controle estáticos baseados em QoS não são mais plenamente suficientes;

- Existem dificuldades crescentes para traduzir diretamente em medida objetiva a qualidade de experiência (QoE), isto é, a percepção do usuário;
- Em muitas situações a percepção de qualidade diverge das medidas obtidas pela gerência do provedor;
- Há bastante divergência entre o padrão de qualidade de serviços baseado em QoS apurado pelo provedor e a percepção do usuário final (QoE), ocasionando uma severa perda de perspectiva de controle e gerência, e;
- A heteregoneidade crescente das redes em termos de estrutura de dispositivos de usuários, aplicações e expectativas dificultam a visualização, gerenciamento e controle dos fluxos e aplicações e o atendimento das expectativas sempre crescentes.

1.3 MOTIVAÇÃO

A partir dos problemas identificados em um cenário dinâmico e em constante mutação como o atual, identifica-se a necessidade de propor uma solução para o gerenciamento e controle de fluxos de encaminhamento de dados, visando a melhoria da percepção de aplicações sensíveis ao contexto, tais como as aplicações multimídia convergentes.

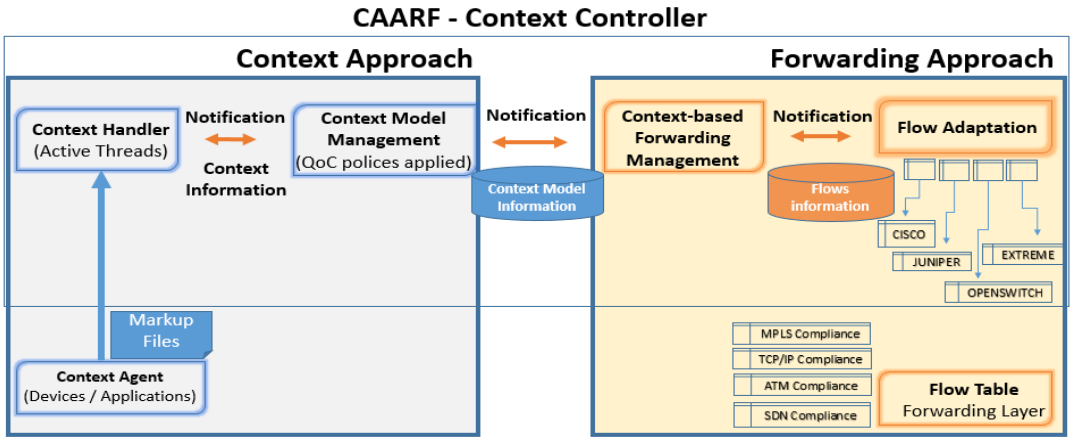
A solução proposta opera em uma pseudo-camada entre a aplicação e a rede (ALRESHOODI; WOODS, 2013) em relação àquelas propostas em arquiteturas tradicionais de rede, e que têm sido prevalentes até o presente momento para efetivar o controle de qualidade, i.e, aqueles baseados nos mecanismos tradicionais de redes de roteadores IP, tais como Diffserv (RFC 4594, 2006) e *Traffic Engineering* sobre redes MPLS, etc.

Essa solução visa proporcionar a criação de uma camada de gerência e controle de encaminhamento baseada na proposta de um arcabouço de gerenciamento de contexto e controle de encaminhamento, chamado de *Context-aware Adaptive Routing Framework* (CAARF). O CAARF representa uma alternativa para a tomada de decisão de encaminhamento em redes de computadores baseada em informações de contexto adquiridas a partir de diversos cenários de utilização.

O projeto do arcabouço CAARF é um desenvolvimento cooperativo e continuado, com a contribuição inicial de quatro alunos do Programa de Mestrado em Sistemas e Computação da Universidade Salvador (UNIFACS). Cada aluno envolvido no projeto (André Oliveira, Carlos

Muakad, Joatham Silva e Sergio Spinola) apresenta uma contribuição distinta e complementar por meio de diferentes visões e aplicações, e que no estágio atual encontra-se em desenvolvimento e validação, conforme apresentado na Figura 1-1.

Figura 1-1 Context-aware Adaptive Routing Framework (CAARF)



O Controlador de Contexto (*Context Controller*), apresentado na Figura 1-1, é constituído por dois módulos principais, denominados abordagens operacionais, chamadas Abordagem de Contexto (*Context Approach*) e Abordagem de Encaminhamento (*Forwarding Approach*). Uma terceira estrutura externa, o Agente de Contexto (*Context Agent*) reside no *Middleware*.

A *Abordagem de Contexto* é o módulo centralizador das informações contextuais adquiridas dos agentes de *Middleware*, e é responsável pela validação das políticas e regras de qualidade de contexto (QoC) e a validação da qualidade de experiência (QoE). Esse módulo também emite notificações acerca de violações de condições de *Service Level Agreement (SLA)* ao módulo de encaminhamento, informando a este a localização do documento contextual vinculado àquela violação.

A *Abordagem de Encaminhamento* é responsável pelo recebimento de notificações de alerta acerca de violações de QoE e a consulta ao documento de contexto corrente. Esse módulo visa providenciar o subsequente processamento das regras de encaminhamento para definição dos caminhos otimizados na rede, e posteriormente da efetivação da adaptação de fluxo por meio de decisões aplicadas aos dispositivos de encaminhamento.

A principal motivação para a realização deste trabalho foi a de contribuir com a definição e desenvolvimento do arcabouço proposto, mais especificamente contribuindo com a

implementação do módulo de **Abordagem de Contexto** e promovendo a validação de sua integração com a Abordagem de Encaminhamento. O objetivo geral e objetivos específicos são apresentados na próxima seção.

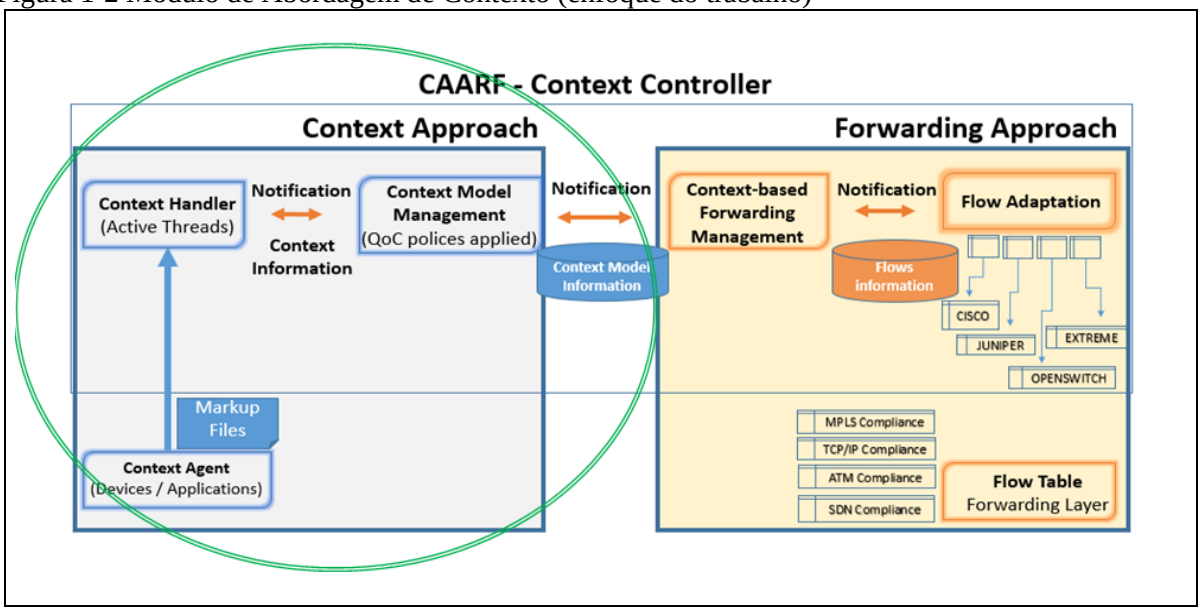
1.4 OBJETIVOS

Este trabalho de dissertação de mestrado tem como objetivo geral a proposta e implementação de uma solução para a o encaminhamento de dados baseado em informações contextuais, aplicada a soluções convergentes. Dentre os objetivos específicos deste trabalho destacamos:

- Definição de um **modelo para a descrição de contexto** e da representação das informações relevantes e necessárias para o encaminhamento de dados;
- Proposta e implementação do módulo de **Gerência de Contexto** para a validação de regras de Qualidade de Contexto (QoC) e Qualidade de Experiência (QoE);
- Proposta e implementação de um **sistema de notificações** de mudança de contexto ao módulo de encaminhamento, dando ênfase à validação de QoE, e;
- Proposta e implementação de um **protótipo de *Middleware*** a fim de promover a integração do Controlador de Contexto com um módulo de aquisição de dados externo.

O escopo deste trabalho e estudos relacionados cobre essencialmente a aquisição dos dados de contexto, sua representação, arquitetura do sistema, implementação dos sistemas de notificações, análise e filtragem de dados contextuais (QoC) e notificação de violação de QoE ao sistema de Encaminhamento. Nesse sentido o enfoque deste trabalho de mestrado é realizado sobre o módulo de **Abordagem de Contexto** (*Context Approach*), ilustrado e destacado na Figura 1-2.

Figura 1-2 Módulo de Abordagem de Contexto (enfoque do trabalho)



1.5 METODOLOGIA

A seguinte metodologia foi adotada a fim de elaborar o trabalho que embasa esta dissertação:

- Realização de estudos comparativos das soluções e modelos existentes para o encaminhamento adaptativo baseado em contexto;
- Estudo de trabalhos relacionados para auxiliar na proposta e posterior definição das relações entre os módulos do arcabouço para processamento das informações contextuais e tomada de decisão sobre o encaminhamento;
- Estudo, revisão e definição do arcabouço CAARF e detalhamento das descrições acerca das interações entre os módulos e da mensageria necessária para o seu funcionamento;
- Definição do modelo de dados e formatos de representação que poderiam ser mais eficientes na implementação do arcabouço CAARF, como um banco de dados NoSQL (MongoDB) e formatos de representaçao JSON e BSON;
- Proposta dos mecanismos e regras necessários para o gerenciamento das informações de contexto;
- Definição e desenvolvimento do módulo de Abordagem de Contexto e seus sub-módulos;
- Estudo e identificação de um conjunto de ferramentas necessárias para a implementação da solução proposta;

- Validação da solução implementada através de estudos de caso baseados em cenários reais e implementados, permitindo variações de cenários baseado em alterações de métricas de QoE, QoC e QoD;
- Definição de um *testbed* baseado em um *gateway* Netfilter e *host* Debian Linux, ferramenta de emulação de chamadas SIP e *hosts* passivos e ativos;
- Modelagem da Arquitetura de Implementação do módulo de Abordagem de Contexto;
- Implementação do módulo de Abordagem de Contexto e encaminhamento baseado em contexto;
- Validação da solução implementada através de estudos de caso baseados em cenários reais, usando técnicas de tratamento de filas, e;
- Validação da solução implementada através da alimentação de dados fornecida pela ferramenta de efetivação de chamadas (SIP) e mensuração de parâmetros de cada chamada, dada pela ferramenta Startrinity.

1.6 PRINCIPAIS CONTRIBUIÇÕES

As principais contribuições realizadas com este projeto são:

- Proposta de uma solução de um **sistema de gestão de contexto** aplicada ao encaminhamento adaptativo em soluções convergentes;
- Implementação da proposta apresentada como **solução de coleta de dados**, formatação, validação e encaminhamento relacionada aos fatores objeto de estudo deste trabalho, destacando-se:
    - Fatores relacionados a dispositivos, no caso de Codecs;
    - Fatores relacionados ao usuário, no caso da captação da percepção de usuário;
    - Fatores relacionados captura de dados de rede.
- Verificação e validação da solução desenvolvida com auxílio de **cenários representativos de caso de uso real**, com dados reais, permitindo a sua replicação e a implementação de variações e extensões ao modelo aplicáveis a diversos cenários de utilização;

- Demonstração da possibilidade de uso da **percepção do usuário** quanto ao serviço que está sendo oferecido a ele, ampliando sua importância nos aspectos de avaliação da rede e seu encaminhamento de informações;
- Por fim, como contribuição direta, a **implementação do módulo de *Abordagem de Contexto*** baseado no arcabouço CAARF proposto e realização de oito **estudos de caso** baseados em cenários reais, ilustrando o funcionamento do arcabouço para demonstrar a realização da gestão de encaminhamento de tráfego usando informações contextuais diversas, destacando-se:
    - Informações de QoE;
    - Informações de percepção de usuário;
    - Informações temporais;
    - Informações de dispositivo (Codec).
- Definições na área da **qualidade do dispositivo**, por meio da validação de Codecs;
- **Integração dos módulos de Abordagem de Contexto e Abordagem de Encaminhamento**, além do protótipo de *Middleware*, eliminando as dependências entre os trabalhos, demonstrada em ciclo contínuo de funcionamento.

Por fim, destaca-se a **definição de um arcabouço para encaminhamento adaptativo** de uso geral que pode utilizar informações de contexto de variadas naturezas, devidamente qualificadas, enriquecendo o sistema de tomadas de decisão, apresentando os conceitos, definições de estrutura, funcionamento, interação entre os módulos e métodos de trocas de mensagens de notificações.

1.7 ORGANIZAÇÃO DA DISSERTAÇÃO

Em sequência ao Capítulo introdutório, esta dissertação está organizada em mais 5 (cinco) capítulos adicionais, conforme apresentados a seguir:

- Capítulo 2 (**Revisão Conceitual e Trabalhos Relacionados**): onde são introduzidos e discutidos os aspectos teóricos e conceituais utilizados ao longo desta dissertação, além do relacionamento e estudo dos principais trabalhos relacionados existentes na literatura;

- Capítulo 3 **(Arquitetura de Encaminhamento Baseado em Contexto**): é apresentada a descrição dos principais conceitos e características do arcabouço CAARF;
- Capítulo 4 **(Implementação: Coleta e Gestão de Contexto):** é apresentada a modelagem da arquitetura de implementação e a modelagem da representação orientada à implementação, descrição dos fluxos de funcionamento e interação entre os principais módulos e respectivos sistemas de mensageria relacionados à implementação de uma arquitetura para o arcabouço *Context-Aware Adaptative Routing Framework* (CAARF), mais especificamente do módulo funcional de Abordagem de Contexto;
- Capítulo 5 (**Validação e Testes**): são apresentados oito estudos de casos baseados no funcionamento real sobre um *testbed* como forma de validar o funcionamento da arquitetura proposta para o arcabouço CAARF, proporcionando melhores soluções para os serviços baseados na experiência do usuário e conclusões acerca destes estudos, e;
- Capítulo 6 (**Conclusões e Perspectivas Futuras**): são discutidas as principais conclusões deste trabalho e de sua continuação através das perspectivas futuras.

2 REVISÃO CONCEITUAL E DE LITERATURA

2.1 INTRODUÇÃO

Este Capítulo aborda os principais conceitos empregados ao longo desta dissertação, tais como Qualidade de Serviço, Qualidade de Experiência, Contexto e Qualidade de Contexto, e também aspectos sobre técnicas de Engenharia de Tráfego e a arquitetura das Redes Definidas por Software. Além desses conceitos, são abordadas as principais métricas e parâmetros considerados no embasamento da modelagem e implementação, os quais foram submetidos a experimentos visando resultados práticos. Por fim, dois tópicos são dedicados ao estudo de Banco de Dados e à linguagem de implementação empregados e suas características.

Inicialmente são elencados os principais parâmetros e métricas pertinentes ao Gerenciamento de Contexto e posteriormente é apresentado um estudo acerca do Estado da Arte das redes, abordando o aspecto do roteamento adaptativo.

2.2 REVISÃO CONCEITUAL

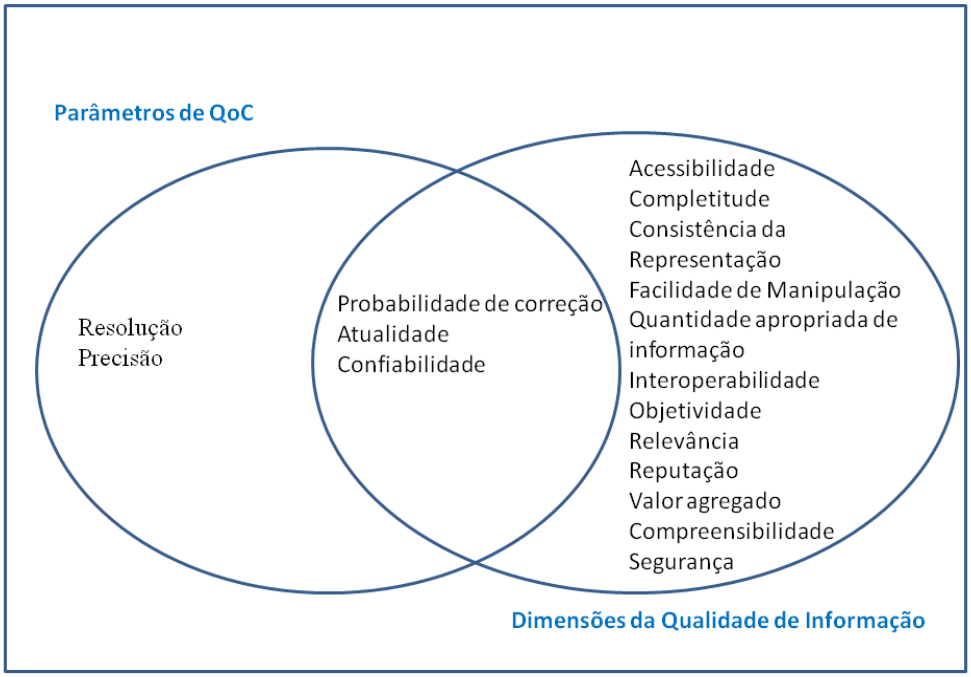
Nesta seção são apresentados os principais conceitos empregados ao longo desta dissertação e que embasam a sua utilização:

2.2.1 Contexto e Qualidade de Contexto

Dentre as definições aplicadas a contexto ressaltam-se algumas, como aquela apresentada em (DEY, 2011), como “qualquer informação que pode ser usada para caracterizar uma situação de uma entidade, que é entendida como uma pessoa, lugar ou objeto que é considerada relevante na interação entre um usuário e a aplicação, incluindo o próprio usuário e a aplicação”. Uma outra descrição apresentada em (KORKEA-AHO, 2000) descreve o contexto como uma “informação situacional, praticamente toda informação disponível durante a interação, que pode ser compreendida como informação de contexto”. Em outra definição apresentada em (CHEN; KOTZ, 2000) “uma aplicação automaticamente adapta-se ao contexto descoberto alterando o comportamento da aplicação. No caso da sensibilidade passiva ao contexto, a aplicação apresenta ao usuário interessado o contexto atualizado ou novo, ou torna o contexto persistente para a recuperação em momento ulterior”.

Tomando como base as definições anteriores, podemos inferir que o **contexto** é o conjunto das informações relevantes que são associadas a uma aplicação e a um determinado usuário. Estas informações podem ser aquelas diretamente usadas pela aplicação juntamente com aquelas que influenciam direta ou indiretamente na performance e resultado da aplicação e sua percepção por parte do usuário. No âmbito deste trabalho, será ampliado o foco sobre os fatores contextuais relacionados ao Qualidade de Experiência (QoE), descritos mais adiante.

Figura 2-1 Relacionamento entre as dimensões da qualidade de informação e parâmetros de QoC

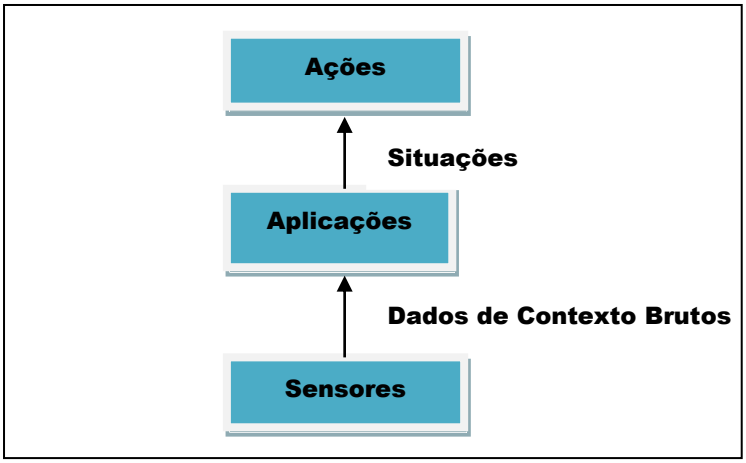


Fonte: Kim e Lee (2006).

Parâmetros relevantes de QoC são apresentados na Figura 2-1, tais como resolução, precisão, probabilidade de correção, atualidade e confiabilidade. Estes parâmetros definem se um determinado contexto proveniente de uma informação apurada deve ser considerado para efeito computacional ou se deve ser descartado. Cálculos estatísticos podem ser executados a fim de tomar decisões com base na completitude de uma informação considerada no âmbito das dimensões do QoC. Deve ser notado também que fatores como imprecisão e segurança da fonte originadora da informação, ou seja, quem gerou a informação e se esta é fidedigna também compõem o cenário de avaliação de um determinado conjunto de informações de contexto.

Em (BUCHHOLZ et al., 2003), por sua vez, Qualidade de Contexto (QoC) é definida como "qualquer informação que descreve a qualidade da informação que é usada como informação de contexto. Desta forma, QoC refere-se à informação."

Figura 2-2 - Fluxo da informação de contexto



Fonte: Kim e Lee (2006).

Neste trabalho, consideraremos a **Qualidade de Contexto (QoC)** como uma apuração da qualidade da informação, no âmbito de uma aplicação associada a um fluxo de dados e a um usuário. Essas informações devem ser coletadas por sensores, tal como disposto na Figura 2-2, onde sensores distribuídos coletam a informação não processada/não tratada, e que são interpretadas em uma camada de aplicação a fim de definir situações que desencadeiam ações a serem tomadas.

2.2.2 Qualidade de Serviço (QoS)

Qualidade de Serviço é um termo que possui definições amplas, e que aborda desde características intrínsecas de uma rede até as percepções de experiência de usuário (STANKIEWICZ; JAJSCZYK, 2011). Em termos de parâmetros perceptíveis ao usuário, a QoS é descrita pelo ITU-T (ITU-T REC G.1000, 2001), ao passo que a descrição do IETF a define como um conjunto de requerimentos de serviços que devem ser obedecidos pela rede durante o transporte de um fluxo (CRAWLEY et al., 1998). Estes requisitos, quando aplicados a soluções sensíveis às garantias de entrega, desempenho e banda de comunicação disponíveis no sistema de encaminhamento de pacotes, como é o caso da interação por voz e vídeo em tempo real, podem ser determinados, dentre outros parâmetros, pelas taxas de atraso

na entrega dos pacotes, no ordenamento da entrega destes pacotes, na variação deste atraso e na taxa de perda de pacotes (MAMMERI, 2005).

De forma geral, **Qualidade de Servico (QoS)** designa as técnicas, mecanismos, protocolos e métricas que visam garantir que um determinado serviço baseado em um fluxo de dados atenda a requisitos pré-estabelecidos. Este conceito é relativamente antigo no universo das redes locais e foi implementado inicialmente, no caso das redes IP, por meio da arquitetura *Integrated Services Architecture* (IntServ) (BRADEN et al., 1994) e seu protocolo de sinalização associado *Resources Reservation Protocol* (RSVP) (BRADEN et al., 1997), e mais tarde sucedido pelo *Differentiated Services Architecture* (DiffServ) (BLAKE et al., 1997). O RSVP foi desenvolvido como um protocolo de sinalização destinado a reservar recursos para fluxos de tráfego, porém o modelo IntServ/RSVP apresenta restrições de escalabilidade que comprometeram o seu uso na prática em redes de Provedores de Serviço WAN, motivo pelo qual o DiffServ foi proposto. A abordagem do DiffServ é modular, incremental e escalável, e nela acordos de nível de serviços, *Service Level Agreements* (SLAs)**,** são fixados entre os clientes e o domínio DiffServ, especificando aspectos como taxas de pico, médias de utilização de banda de comunicação, métricas de atraso, taxas de erro, e condicionantes de tráfego (MAMMERI, 2005).

Estes conceitos foram inicialmente aplicados às redes exclusivamente baseadas em protocolo IP, plenamente aplicáveis em redes de âmbito corporativo e de controle fim-a-fim, onde podem ser exercidos atualmente, com êxito, sobretudo em redes corporativas privadas. Contudo, a dinâmica das novas arquiteturas atualmente existentes, empregadas pelos provedores de serviços externos de enlace, baseadas em MPLS (*Multiprotocol Label Switching*) (RFC 3031, 2001) e GMPLS (*Generalized MPLS*) (RFC 3495, 2004) adotadas pelos provedores de interconexão, os Provedores de Serviço, passaram a exigir uma evolução destas abordagens, notadamente por meio da Engenharia de tráfego. São desafios que também cercam as novas arquiteturas propostas e baseadas em Redes Definidas por Software (SDN).

2.2.3 Qualidade de experiência

Qualidade de Experiência (QoE) é definida pelo ITU-T (ITU-T Rec P.10, 2006) como a aceitabilidade de uma aplicação ou serviço conforme percebida subjetivamente pelo usuário final. QoE trata da quantificação da percepção do subjetivo, mudando o ponto de vista da aferição de qualidade da rede para o usuário. Portanto, o tráfego de rede passa de "*network-*

centric" (orientada à rede) a ser "*user-centric*" (orientada ao usuário), envolvendo fatores cognitivos, comportamentais e psicológicos (MITRA et al., 2013). Assim são estudados a perspectiva e compreensão da percepção do usuário, contemplando expectativas e experiência com a aplicação.

O desafio que se apresenta na compreensão dos fatores de QoE é determinar que parâmetros se pretende medir objetivamente, mapeando, antecipando e prevendo a percepção subjetiva com base em métricas objetivas. Deseja-se, portanto, inferir o nível subjetivo de satisfação do usuário a partir de métricas objetivas.

Em outras palavras, o problema com o qual os provedores se defrontam é que não necessariamente boas práticas de QoS ou engenharia de tráfego (*Traffic Engineering*), notadamente em redes MPLS, resultam em satisfação do usuário, isto é, o QoE, ainda que os parâmetros de QoS aparentemente estejam bem ajustados.

De maneira geral, o QoS é "centrado na rede", e o QoE é "centrado no usuário" (ALRESHOODI; WOODS, 2013). O QoE pode ser entendido também como uma cobertura de quatro domínios: contextual, tecnológico, negócios e humano (LAGHARI ; CONNELLY, 2012), entendido como uma pseudo-camada entre a aplicação e a rede (ALRESHOODI; WOODS, 2013).

Além das métricas já conhecidas do QoS, como *perda de pacotes, variacão de atraso (jitter), atraso*, entre outras, devemos acrescer as métricas pertinentes às aplicações multimídia. A métrica mais conhecida correlacionada ao QoE é o MOS (*Mean Opinion Score*) (Carvalho et al., 2005), que classifica a experiência entre ruim, pobre, aceitável, boa e excelente (Shaikh et al. 2010). Trata-se de uma métrica subjetiva que pode ser obtida por meio de pesquisa de satisfação ou por inferência a partir da correlação com outros parâmetros. No campo das aplicações multimídia recorre-se ao parâmetro frequentemente adotado para qualificar o QoE dado pelo *Peak Signal Noise Ratio (PSNR)*, e o seu mapeamento em MOS (VENKATARAMAN et al., 2007) e (WANG, 2006). Outra abordagem parte do mapeamento PESQ (*Perceptual Evalution Of Speech Quality*) em MOS foi apresentada em (HOßFELD et al., 2007).

A mensuração da satisfação dos usuários destas aplicações sob a ótica destes representa um desafio ampliado em termos de medição, requerendo a definição de métricas e aferições que não podem ser obtidas diretamente a partir dos parâmetros clássicos de QoS isoladamente. O

que se tem observado é que a percepção do usuário cada vez mais contraria a convicção de que políticas e regras de QoS bem definidas são uma garantia absoluta de aprovação da qualidade de experiência do usuário.

De forma geral, as pesquisas atuais objetivam mapear parâmetros de QoS e parâmetros já existentes ligados a análise e aferição de QoE por meio de fórmulas matemáticas e parâmetros já existentes como proposto em (HOßFELD et al., 2007).

Algumas proposições consideram que a natureza da percepção de QoE parte de hipóteses que definem a correlação entre QoS e QoE, envolvendo parâmetros já existentes, os quais são em geral aplicados a aferição de aplicações multimídia. Dentre estas destaca-se a Hipótese de IQX, que define uma correlação entre QoS e QoE, baseada no princípio de que a variação da qualidade de experiência em um dado momento é proporcional à Qualidade de Serviço existente naquele instante (HOßFELD et al., 2008).

A correlação entre QoE e QoS através da aplicação da hipótese de IQX é também explicada pela Lei de Weber-Fechner (Reichl et al., 2010). Um dos princípios da hipótese IQX é dado pelo fato de a variação da Qualidade de Experiência ser tão maior quanto maior o nível de serviço em que nos encontramos. Desta forma, a percepção de perda de qualidade é muito grande quando a qualidade é grande e pequena quando a qualidade já é reduzida.

Tabela 2-1 - Fatores traduzíveis em QoE

Autor	(a)	(b)	(c)	(d)	(e)	(f)
PSNR	☑	☑	☐	☑	☐	☑
VQM	☑	☑	☐	☑	☐	☑
MPQM	☐	☐	☐	☑	☐	☑
SSIM	☐	☐	☐	☑	☐	☑
NQM	☐	☐	☐	☑	☐	☐
MOS	☑	☑	☐	☑	☑	☐
TUQ	☑	☐	☐	☐	☐	☐
SSQ	☑	☐	☐	☐	☐	☐
PESQ	☐	☐	☑	☐	☑	☐
Sincronização na Entrega	☐	☐	☐	☐	☐	☑
Tempo de Resposta	☐	☐	☐	☐	☐	☑
Vivacidade	☐	☐	☐	☐	☐	☑

(a) [Alreshoodi e Woods, 2001], (b) [Venkataraman et al., 2007], (c) [Reichl et al., 2007]
(d) [Wang, 2006], (e) [Hoßfeld et al., 2007] , (f) [Graciá et al., 2007]

Outras abordagens requerem a pesquisa direta com o usuário, inclusive empregando *crowdsourcing* (CHEN et al., 2010), subentendendo *crowdsourcing* como o processo de aquisição de informações a partir de grandes grupos (ou multidões) de usuários online. Uma outra abordagem ainda faz uso de "*probes*" (SHAIKH et al., 2010) estrategicamente distribuídas. Uma *probe* específica é implementada pela ferramenta PRTG (PRTG, 2015) em que um sensor de QoS envia pacotes entre duas *probes* e efetiva a mensuração da qualidade de conexão entre as elas (PAESSLER, 2015).

As Tabelas 2-1, 2-2 e 2-3 apresentam uma classificação das métricas por tipo e perfil de utilização, conforme descrição de vários autores. A Tabela 2-1 relaciona métricas que podem ser traduzidas em QoE (no caso de multimídia), a Tabela 2-2 refere-se a fatores de QoS que podem ser traduzidos em QoE e a Tabela 2-3 relaciona os fatores de percepção subjetiva humana correlacionados ao QoE.

Tabela 2-2 - Fatores de QoS relacionados às medidas de QoE

Autor	(a)	(b)	(c)	(d)	(e)
Pacotes fora de sequência	☐	☐	☐	☑	☐
Jitter	☐	☑	☐	☐	☑
Perda de pacotes	☐	☑	☐	☐	☑
Latência	☑	☐	☐	☑	☐
Atraso	☐	☑	☐	☐	☑
Atraso ida-e-volta	☐	☐	☐	☑	☐
Capacidade	☐	☑	☐	☐	☐
Taxa de Erro	☐	☐	☑	☐	☐
Banda	☐	☑	☐	☐	☐

(a) [Alreshoodi e Woods, 2001], (b) [Mitra et al.,2011], (c) [Venkataraman et al., 2007], (d) [Saikh et al., 2010], (e) [Graciá et al., 2007]

Este desafio pode ser exemplificado pelo caso do usuário em uma estação de metrô. Se por um lado este usuário pode reportar uma má experiência com a qualidade de voz, por outro lado o ruído de fundo pode influenciar negativamente nesta percepção humana. Se o sistema for capaz de captar os dados referentes ao ruído de fundo composto com a localização e as métricas de QoS, então a tomada de decisões por parte do controlador de contexto poderá levar em conta fatores externos que poderão resultar em uma decisão distinta daquela que

seria tomada no caso do ambiente ideal ou típico de utilização previstos no sistema, i.e, um ambiente silencioso.

Tabela 2-3 - Parâmetros Cognitivos, Comportamentais, psicológicos, do Usuário e Ambiente

Autor	(a)	(b)	(c)	(d)
Desfrute	☐	☐	☑	☑
Atenção	☐	☐	☑	☑
Concentração	☐	☐	☑	☐
Satisfação	☐	☐	☑	☑
Aceitação Tecnológica	☐	☐	☑	☐
Eficiência	☐	☐	☑	☐
Velocidade	☐	☐	☑	☐
Acuidade	☐	☐	☑	☑
Percepção de facilidade	☐	☐	☑	☑
Estado Emocional	☐	☐	☑	☑
Humor	☐	☐	☑	☑
Localização	☑	☑	☑	☐
Temperatura	☐	☐	☑	☑
Contexto Social	☐	☑	☑	☐
Vizinhança	☐	☐	☑	☑
Ruído de Fundo	☐	☑	☑	☐

(a) [Alreshoodi e Woods, 2013], (b) [Skorin e Varela, 2012], (c) [Mitra et al., 2011], (d) [Laghari e Connelly, 2012]

A seguir são apresentadas as principais métricas empregadas no cálculo de QoE, e que serão amplamente utilizadas neste trabalho para efeito de validação de regras de QoE.

O *Mean Opinion Score* (MOS) (ITU-T P.800.1, 2003) foi concebido historicamente como uma métrica que consiste em atribuir um valor numérico indicador da qualidade percebida da voz em um sistema de telefonia tradicional. Originalmente a medida de MOS correspondia simplesmente a uma média aritmética adquirida de levantamentos realizados entre indivíduos, por meio de questionamentos diretos em entrevistas, acerca da qualidade de uma conversação. Embora a métrica tenha sido criada com o intuito de atribuir uma classificação a partir de pesquisa diretamente com os usuários, envolvendo a participação humana, foi posteriormente substituída por softwares capazes de inferir o MOS a partir da coleta de dados de QoS (CARVALHO et al., 2005).

Posteriormente a métrica passou a ser adotada na aferição de qualidade de chamadas VoIP, cuja escala é constituída por uma nota de avaliação compreendida entre 1 e 5. Em geral, as chamadas VoIP de qualidade considerada aceitável recaem na faixa de 3.5 a 4.2 em termos de escala MOS, em razão de limitações impostas pelos Codecs. A Tabela 2-4 mostra a correlação entre percepção de qualidade e valores de MOS atribuídos.

Tabela 2-4 - Relação entre satisfação e MOS aferido

Nível de Satisfação	MOS
Máximo (para o Codec G.711)	**4.4**
Muito Satisfeito	**4.3-4.4**
Satisfeito	**4.0-4.3**
Satisfação moderada e algumas Insatisfações	**3.6-4.0**
Insatisfação	**3.1-3.6**
Muita insatisfação	**2.6-3.1**
Não é recomendado	**1.0-2.6**

Fonte: Tamos (2015).

O MOS é uma métrica subjetiva e, por este motivo, a fim de se obter uma correlação entre a métrica subjetiva e uma aferição precisa é necessário analisar parâmetros como atraso na rede, perda de pacotes e jitter em conjugação com outros fatores. A fim de conhecer o MOS, portanto, foi adotada uma medição menos subjetiva, denominada R-Factor (CARVALHO et al., 2005), abordada mais adiante e cuja relação com o MOS pode ser visualizada na Tabela 2-5.

Tabela 2-5 - Relação entre MOS, equivalência R-Factor e qualidade percebida

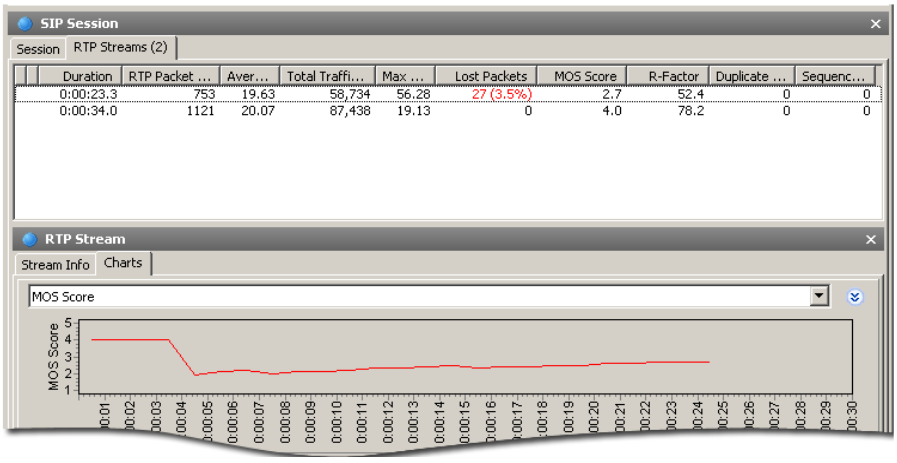
Nível de Satisfação do Usuário	MOS	R-Factor
Máximo Utilizando o G.711	4.4	93
Muito Satisfeito	4.3-5.0	90-100
Satisfeito	4.0-4.3	80-90
Razoável	3.6-4.0	70-80
Alguma insatisfação	3.1-3.6	60-70
Muita insatisfação	2.6-3.1	50-60
Não Aceitável	1.0-2.6	Menos que 50

Fonte: Tamos (2015).

MOS é portanto uma escala relativa que leva em consideração vários fatores que podem afetar a qualidade de voz. As medidas, no caso de chamada VoIP, levam em consideração o atraso unidirecional (*delay one-way*), a latência da conexão, perda de pacotes e *jitter*. A partir dos dados coletados é possível calcular o R-Factor, que então é utilizado para estimar o MOS, o que pode ser realizado por um sensor de rede.

O **R-Factor** surgiu como uma pontuação (*score*) objetiva para mensurar a qualidade de uma chamada de voz. Existem variações na medição do R-Factor que levam em conta parâmetros mais detalhados relacionados com a acústica e fatores eletro-magnéticos relacionados com dispositivos de conversão de sinal e áudio. Neste trabalho iremos considerar a utilização do R-Factor relacionada com a mensuração de qualidade de chamadas de voz sobre IP (VoIP) exclusivamente. A escala do R-Factor compreende uma atribuição de valores compreendida entre 0 e 100. Variados analisadores de rede são capazes de calcular os dois fatores, como visto na Figura 2-3. Na captura de tela obtida a partir de uma analisador de tráfego multimídia (TAMOS, 2015) são analisados dois fluxos RTP (RFC 3551, 2003), um em cada direção, compreendendo uma chamada SIP com MOS avaliado em 4.0 em uma direção e 2.7 na direção inversa.

Figura 2-3 - Sensor analisa o streaming RTP e calcula os fatores MOS, R-Factor, entre outros parâmetros



Fonte: Tamos (2015).

O **E-Model** está baseado no conceito de que cada fator de pertubação que afeta uma chamada de voz pode ser computado separadamente, ainda que estejam correlacionados. O *score* proposto é denominado R-factor, e que é relacionado com o MOS através da seguinte formulação (MITRA et al., 2011):

- Para $R < 0$: $MOS = 1$, e;
- Para R entre 0 e 100 : $MOS = 1 + 0.035R + 7.10^{-6}R(R - 60)(100 - R)$, e;
- Para $R > 100$: $MOS = 4.5$.

O maior motivo para se empregar o E-model como um modelo de medição de fatores de QoE é que este leva em conta as estatísticas de transmissão (associados a atrasos no transporte, perda de pacotes, entre outros), assim como características da aplicação de voz, como qualidade do *codec*, robustez do *codec*, confrontados com a perda de pacotes e descarte dos mesmos.

Outros esforços descritos na literatura proporcionam a padronização de protocolos e métricas para a implementação de aplicações VoIP e disponibilização de serviços multimídia em geral, tal como IMS. A arquitetura IMS (IP *Multimedia Subsystem*), também denominada Multimídia NGN (Next Generation Network), definida em camadas, é um padrão da indústria de telecomunicações e que objetiva descrever uma plataforma de Fornecimento de Serviços, SDP (Service Delivery Platform) (PAVLOVSKI, 2007).

Através desta plataforma de serviços, fortemente associada ao protocolo SIP, efetiva-se o fornecimento de sessões multimídia em tempo real de voz, vídeo, serviços de televisão via internet e videoconferência, assim como um conjunto de serviços adicionais, tais como o de troca instantânea de mensagens, através de diferentes tipos de redes, provendo a convergência de serviços e infraestrutura, notadamente em redes móveis (ZNATY ; DAUPHIN, 2015).

No entanto, para fins de validação da abordagem realizada neste trabalho apenas as métricas MOS e R-Factor foram adotadas.

2.2.4 Qualidade de Dispositivo

A **Qualidade de Dispositivo (QoD)** está relacionada com o fornecimento de informações sobre as características técnicas de cada dispositivo e suas capacidades (VIEIRA et al. 2009), e está diretamente relacionada com a precisão e conformidade do dispositivo computacional a partir do qual informações relevantes de contexto serão coletadas. Um exemplo da aplicação deste conceito pode ser verificado no mecanismo de GPS (*Global Positioning System*) no que concerne ao nível de precisão e acuidade, ou então na capacidade/incapacidade de um dispositivo fornecer dados requeridos na faixa de precisão estabelecida, total ou parcial, de coletar informações conforme os requisitos de uma determinada aplicação (MASCOLO; MUSELI, 2006).

Esta abordagem tem se ampliado em função da difusão cada vez maior dos dispositivos móveis e sem-fio (*wireless*) com alto poder computacional. Para estes cenários de utilização a aquisição de dados acerca do dispositivo compreende parâmetros tais como: estabilidade do enlace de comunicação (*link wireless*), banda de comunicação disponível, autonomia de baterias, capacidade de armazenamento e limitações de processamento, cobertura, potência de sinal, assim como dados obtidos por outros tipos de sensores embarcados em aparelhos móveis celulares, capazes de coletar:

- Dados ambientais: como temperatura, altitude, pressão atmosférica, ruído;
- Dados locacionais/geográficos: latitude, longitude, velocidade, e;
- Dados corporais (humanos): batimento cardíaco e temperatura corporal.

A relevância dos dados e sua aplicabilidade serão definidos pelo tipo de aplicação associada, e dados que são relevantes ou essenciais em um determinado caso, podem ser descartados em outro e vice-versa. Desta forma, é a combinação usuário/aplicação que irá determinar as regras e políticas que devem ser evidentemente específicas a fim de fornecer a análise correta da informação contextual, para que esta esteja em conformidade com o QoC (MASCOLO; MUSELI, 2006).

Os *codec*s, por exemplo, podem ser considerados como um caso especial de componente de um dispositivo, pertencentes basicamente a duas categorias, os *codec*s de áudio e os *codec*s de vídeo. O *codec* de áudio é o elemento que, em aplicações multimída, é o responsável pelo processo de conversão e reconversão da voz humana entre sinais analógicos e digitais, compreendendo a quantização da voz, por meio de amostragem realizada com uma frequência de 8KHz (no caso da telefonia tradicional). Dentre os *codecs* de áudio mais utilizados em redes de dados, destacam-se o G.711, G729 e G.723 (COLLINS, 2003), além de G.726 e G.728 e o AMR em redes móveis (*GSM specification 06.90*) (COLLINS, 2003).

Em vídeo encontramos *codec*s em aplicações profissionais de vídeo conferência como MPEG-4 (RFC 6416, 2011), H.264 AVC (RFC 6184, 2011) e H.264 SVC (RFC 6190, 2011), entre outros. No caso dos *codec*s de vídeo cabe considerar que a relação entre a quantidade de banda requerida para um dado *codec* e qualidade de vídeo efetiva podem variar bastante, assim como a tolerância a erros. Como exemplo, cabe ressaltar que a codificação efetuada com o *codec* H.264 AVC admite uma taxa de erro máxima da ordem de 2 a 3%, ao passo que no caso do SVC é possível admitir taxas de erro bem elevadas, como de até 20% (VITEL, 2015) para um mesmo resultado de QoE. Estes são fatores que ressaltam a importância do

codec, ou seja, o dispositivo possui um peso significativo quando correlacionado com o QoS a fim de determinar o QoE de uma aplicação.

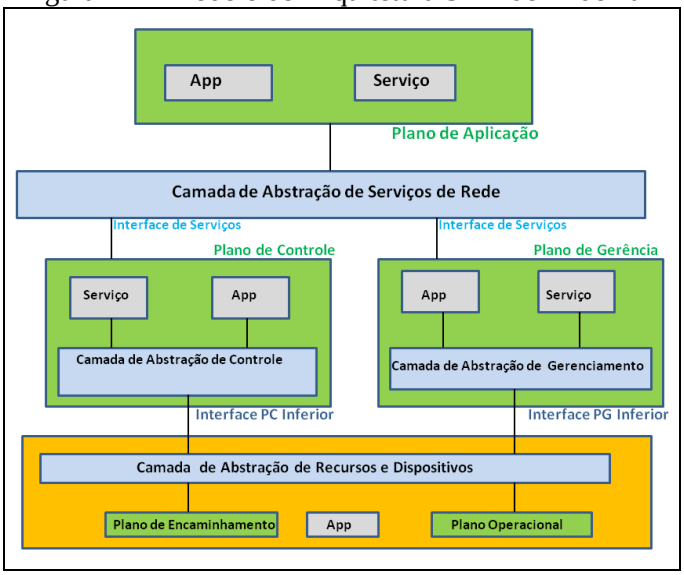
Outro paradigma para a definição de qualidade e flexibilidade nas redes de computadores está associado à utilização das Redes Definidas por Software (SDNs). A próxima seção introduz esse paradigma.

2.2.5 Redes Definidas por Software

Redes Definidas por Software – *Software Defined Networks* (SDN) refere-se a uma abordagem através da programabilidade da rede, ou seja, tornar possível inicializar, alterar e gerenciar o comportamento da rede dinamicamente através de interfaces abertas (RFC 7426, 2015). A proposta inicial do paradigma de redes SDN foi proposta originalmente em (CASADO; MCKEWON, 2005) apresentada em Stanford.

É portanto uma arquitetura que compreende o desacoplamento do controle de equipamento de rede das funções de encaminhamento, promovendo a separação física entre o plano de dados e o plano de controle de redes, e que possui atualmente algumas visões distintas. Uma destas visões pode ser visualizada na Figura 2-4, uma versão ilustrada do diagrama de arquitetura de camadas SDN que apresenta a separação entre as camadas de aplicação, controle, gerenciamento e infraestrutura (comutação). De acordo com *Open Networking Foundation* (ONF, 2015), são características da arquitetura SDN:

Figura 2-4 - Modelo de Arquitetura SDN definido na RFC 7426



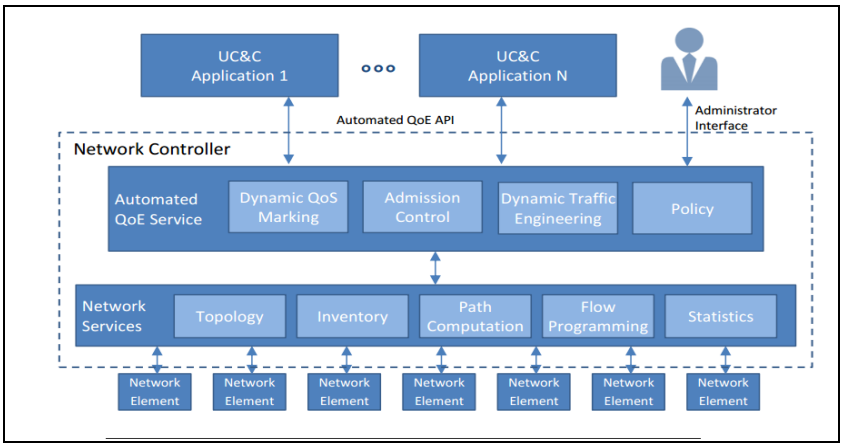
- **Programação direta:** O controle de redes é diretamente programável, pois é desacoplado das funções de encaminhamento;

- **Agilidade**: A abstração do plano de controle do plano de encaminhamento permite aos administradores ajustarem o tráfego de rede adequando-o à dinâmica de necessidades;

- **Gerenciamento Centralizado**: A inteligência de rede é centralizada em controladores SDN, que mantêm uma visão global da rede, que aparentam ser, do ponto de vista de aplicações e mecanismos policiadores, um único switch lógico;
- **Configuração Programática**: o SDN permite aos gerentes de rede configurar, gerenciar, assegurar, e otimizar recursos de rede rapidamente através de programas SDN automatizados, que podem ser escritos livremente e são independentes de software proprietário, e;
- **Baseado em padrões abertos e neutros**: Baseado na ideia de que os projetos serão simplificados por obedecerem um padrão geral e não serem dependentes de especificidades de fabricantes diversos.

Esta proposição abre a possibilidade, dentre outras, para que parâmetros especificados dinamicamente por uma aplicação convergente possam ser aplicados através de uma API exposta por um serviço de automação de QoE, conforme visualizado na Figura 2-5, extraída do trabalho original em (LAUWERS et al., 2014) que apresenta esta proposição. Dessa forma, um serviço automatizado de QoE traduz as requisições QoS fim-a-fim provenientes das aplicações convergentes, em políticas de QoS dinâmicas aplicadas a cada um dos elementos habilitados para SDN no ingresso da rede.

Figura 2-5 - Diagrama esquemático de APIs QoE agindo sobre o plano de controle de uma rede SDN



Fonte: Lauwers (2014).

Como visto nas seções anteriores a proposição da separação do plano de controle do plano de dados é característico em outras arquiteturas, como no próprio GMPLS, o que permite concluir que variadas estruturas baseadas em diferentes arquiteturas deverão conviver por bastante tempo. Assim, novas soluções de otimização de encaminhamento devem considerar esta heterogeneidade em que variações de arquiteturas baseadas em redes definidas por software, em seu sentido mais amplo, deverão coexistir.

2.2.6 Engenharia de Tráfego

A **Engenharia de Tráfego em redes privadas (em um domínio)** é definida como o aspecto da engenharia de redes que lida com a questão da avaliação e otimização de performance compreendendo os princípios científicos e da tecnologia para medir, caracterizar modelar e controlar o tráfego Internet (AWDUCHE et al., 2002). A Engenharia de Táfego (*Traffic Engineering*), ou simplesmente "TE" trata da questão referente à otimização de redes operacionais, envolvendo a avaliação de desempenho e sua otimização em redes de roteadores e comutadores de provedores de serviços (*carriers*), e acabou por constituir na prática uma extensão aplicada às redes MPLS, cujo protocolo carecia de um controle de QoS inerente em suas primeiras versões.

Adicionalmente, a meta da engenharia de tráfego é a minimização da vulnerabilidade da rede decorrente de erros, falhas, e rupturas internas à infraestrutura. Em última instância, o objetivo é entregar aos usuários finais dos serviços um desempenho condizente com suas expectativas

e o acordo de nível de serviço, mais conhecido como SLA (*Service Level Agreement*) contratado. Entre fatores a serem considerados estão: largura de banda dos enlaces, capacidade dos buffers, recursos computacionais, gerenciamento de filas, estratégias de despacho/escalonamento, e outras funções que regulam o fluxo do tráfego através da rede.

A otimização dos aspectos da engenharia de tráfego pode ser vista a partir da perspectiva do controle, que poderá ser proativo ou reativo. No caso proativo, o sistema de controle de engenharia de tráfego deverá tomar ações preventivas que antecipem estados desfavoráveis no estado da rede. Ao agir reativamente, o sistema de controle responde corretivamente e, de forma ideal, se adaptará aos eventos já instalados (roteamento adaptativo), como no caso do congestionamento, o que é um dos problemas mais significativos em uma rede operacional baseda em IP (AWDUCHE et al., 2002).

O MPLS–TE (COLLINS, 2003) está associado com a ideia do roteamento baseado em restrições, *constraint based routing* (CBR), que é responsável pela seleção de caminhos considerados de alguma forma ótimos, que satisfarão a um dado conjunto de restrições e requisitos. Métricas utilizadas no CBR incluem o custo financeiro, número de nós intermediários, taxa de transmissão, confiabilidade, atraso fim-a-fim e variação de atraso fim-a-fim.

O CR-LDP (RFC 3213, 2002) foi adotado inicialmente, porém o grupo de trabalho MPLS do IETF decidiu por preteri-lo em favor do RSVP-TE (AWDUCHE et al., 2002), incorporado ao MPLS-TE. Quando a escolha do caminho a ser utilizado é feita através do algoritmo de menor caminho, o método é denominado CSPF (*Constrained Shortest Path First).*

Combinando as duas tecnologias (MPLS e DiffServ), compõe-se a DS-TE, e torna-se possível impor restrições (constraints) de banda para diferentes Classes de Servicos (CoS). O MPLS-TE (RFC 2702, 1999) possibilita a reserva de recursos em um caminho através dos protocolos RSVP-TE (RFC3209, 2001), (RFC 5151, 2008), ou *Constraint-Based Routing with Label Distribution Protocol* (CR-LDP) (RFC 3213, 2002). Uma abordagem para a realização de CBR nos LSRs é utilizar uma extensão do *shortest path algorithm* (SPF), denominada *constraint-based, shortest path first* (CSPF).

O *Generalized* MPLS (GMPLS) estende o MPLS para fornecer um plano de controle e sinalização para dispositivos que realizam comutação (*switching*) em qualquer dos seguintes domínios: pacotes, temporal (TDM), comprimento de onda e fibra. Este plano de controle

comum objetiva simplificar a operação de redes e o gerenciamento através da automatização do provisionamento fim-a-fim de conexões, gerenciamento dos recursos de rede e fornecer o nível de QoS que é esperado. Propicia portanto a ampliação do escopo de aplicações para uma ampla gama de tecnologias de comutação, como comutação TDM, lambda WDM e comutação espacial (fibra, porta).

São implementados no GMPLS melhorias nos protocolos de sinalização RSVP-TE e CR-LDP – para fins de engenharia de tráfego, além de OSPF-TE e IS-IS-TE – propiciando a divulgação dos recursos óticos, como banda em λs, tipos de proteção de links, e identificadores de interfaces e fibras, além do do protocolo LMP (*Link Management Protocol*) (RFC 4204, 2005), destinado a lidar com o gerenciamento do enlace físico de fibra ótica.

Em ambientes multicamada, um plano de controle GMPLS único controla várias camadas de comutação distintas ao mesmo tempo, por meio de uma unificação coerente, tomando partido do benefício da propriedade comum do GMPLS. O GMPLS possibilita, portanto, uma engenharia de tráfego (TE) para redes multicamadas, constituindo uma maneira de integrar o plano de controle de redes e tarefas centrais de gerenciamento de redes (VIGOUREAUX, 2005).

2.2.7 Banco de Dados - MongoDB

MongoDB, é um acrônimo derivado do termo *humongous* – colossal, em tradução livre, um banco de dados orientado a documentos, classificado como NoSQL. É um banco de dados livre e aberto (*open-source*), que fornece alto desempenho, expressividade, alta disponibilidade e escalabilidade automática.

Um registro no MongoDB equivale a um documento, que é uma estrutura de dados composta por um par {**campo:valor}**. Documentos MongoDB são similares a objetos JSON (JSON, 2015), e os valores dos campos podem incluir outros documentos arrays e arrays de documentos.

O MongoDB obedece aos princípios delineados pela filosofia de desenvolvimento baseada em *sprints* ágeis, onde a tônica do processo de desenvolvimento é a entrega de código, por parte de equipes pequenas, com uma frequência que pode atingir várias vezes ao dia.

Neste sistema os documentos são armazenados em uma representação binária chamada BSON (Binary JSON) (BSON, 2015). Documentos BSON contêm um ou mais campos que por sua vez podem conter um ou mais sub-campos, onde cada campo contém o valor de um tipo específico de dado, incluindo arrays, dados binários e sub-documentos. Documentos que compartilham a mesma estrutura são denominados coleções (*collections*).

Com o modelo de documentos um dado registro é mais "local", reduzindo significativamente a necessidade de realizar o "*JOIN*" - concatenando tabelas separadas, o que influencia positivamente na performance e escalabilidade através de hardware padrão, dado que uma leitura única no banco de dados, pode recuperar em memória o documento completo, recuperando todos os dados relacionados.

A escalabilidade da base de dados em hardware comum ou na nuvem é alcançada empregando um processo denominado "sharding", que é transparente para as aplicações e consiste na distribuição de dados através de múltiplas partições físicas denominadas *shards*. Este é um processo que permite superar limitações de hardware de um único servidor através de várias máquinas sem adicionar complexidade à aplicação. Os dados são automaticamente balanceados no cluster "*sharded*" na medida do crescimento/redução do tamanho da base de dados. O processo de *sharding* é automático e embutido na base de dados (MONGODB UNIVERSITY, 2015).

O MongoDB apresenta um *schema* flexível. Distintamente de outros bancos de dados SQL, onde é preciso declarar o *schema* da tabela antes de inserir dados, a estrutura de dados não requer, nem exige uma estrutura formal de documentos. Esta flexibilidade facilita enormemente o mapeamento de documentos em uma entidade ou objeto, mesmo se os dados apresentarem uma variabilidade substancial.

Em termos gerais, não é requerida uma definição formal baseada em declarações para a modelagem de dados. Este tipo de estruturação pode ser bastante adequada em situações onde existe uma grande variabilidade de dados e uma dinâmica na estruturação. Apesar desta grande flexibilidade, a boa prática recomendada para a modelagem de dados em bancos de dados orientados a documentos aconselha que os documentos de uma coleção (*collection*) compartilhem de uma mesma estrutura.

A outra abordagem constitui no armazenamento destas relações em um único documento. A estrutura de dados torna possível embutir estruturas de documentos em campos e arrays

dentro de outro documento. Estes modelos são ditos não normalizados, que na prática são mais apropriados ao MongoDB e a rigor, seria possível representar toda a estrutura em um único documento. Desta forma, todos os dados podem ser recuperados em uma única operação e manipulados em memória, o que é mais eficiente (DATA MODEL MONGODB, 2015).

2.3 ESTADO DA ARTE – ROTEAMENTO ADAPTATIVO

O conceito de roteamento adaptativo remete aos fundamentos do roteamento IP, cujo principal objetivo é viabilizar o trânsito de dados entre a origem e o destino (COLLINS, 2003). O roteamento pode se valer de rotas estáticas, previamente definidas, ou dinâmicas, que dizem respeito à capacidade que a estrutura de rotamento tem de se adaptar a falhas ou contingências nos enlaces que compõem uma rota.

O roteamento refere-se à seleção do caminho (*path*) otimizado e eficiente a partir de alterações de prioridade, eventuais falhas nos dispositivos de roteamento que podem ocorrer nos nós de rede ou em outros componentes da rede. Esta implementação é dependente de protocolos de roteamento que suportam esta escolha de rotas. Dentre estes protocolos destacamos o RIP, OSPF, BGP, IS-IS, CSPF, IGRP, EGRP (COLLINS, 2003). Entretanto, nenhum destes protocolos constitui um roteamento adaptativo por si.

Em contraposição ao roteamento adaptativo, o roteamento baseado em rotas estáticas não provê a adaptabilidade, estando portanto sujeito à interrupção de encaminhamento dos pacotes em caso de falha observada em qualquer enlace da rede (COLLINS, 2003).

Uma das técnicas mais empregadas em disponibilização de serviços de provimento de enlaces é a Engenharia de Tráfego, que em geral congrega também técnicas de roteamento dinâmicas aplicadas a estruturas de serviços de rede baseados em MPLS (AWDUCHE et al., 2002) e GMPLS (VIGOUREUX, 2005). As técnicas aplicadas em engenharia de tráfego, embora proponham um tipo de roteamento adaptativo que podemos considerar dinâmico, utilizando técnicas baseadas em RSVP-TE, consideram basicamente aspectos relacionados a métricas de QoS (ALRESHOODI; WOODS, 2013) e roteamento baseado em restrições *Constraint Shortest Path First* (CSPF).

Moltchanov (2011) ainda categorizou os protocolos e abordagens existentes baseadas no informação do estado da rede em pró-ativas (orientadas à utilização do estado da rede

armazenado em tabelas), reativas (cujo otimização do tráfego é calculada sob demanda a partir de uma necessidade identificada na rede) e híbridas, que agrega as vantagens de ambas abordagens reativas e pró-ativas. Em particular, as abordagens híbridas proporcionam uma rápida convergência, menor sobrecarga na troca de informações do estado da rede, maior escalabilidade e um menor requisito de armazenamento e processamento. No geral, algumas das contribuições existentes na literatura são baseadas em:

- Estado do sistema (usuário, rede e dispositivos de comunicação e apresentação) (MUSOLESI et al., 2005) (WENNING et al., 2009) (PETZ et al., 2012) (KABBANI et al, 2014) (ZHAO; RAYCHAUDHURI, 2006) (OH et al, 2014);
- Engenharia de tráfego (FISCHER et al, 2006) (KANDULA et al, 2005) (KARTHIGA; BALAMURUGAN, 2013) (WANG et al, 2012), e;
- Dados probabilísticos (XIE et al, 2004).

Em geral, as abordagens relacionadas com roteamento adaptativo baseado em contexto aplicam diferentes métricas comparáveis àquelas aplicadas em Qualidade de Serviço, como atraso, variação do atraso perda de pacotes, etc. Por outro lado os algorítmos e soluções implementadas como protocolos baseados em contexto, em geral, têm se valido de outras métricas como, por exemplo, o nível de bateria (MU et al., 2014), em que o tempo de vida dos equipamentos em termos de carga de bateria é otimizado através do balanceamento de cargas de energia, uma característica associada a redes de sensores sem fio.

Vários trabalhos apresentam enfoque específico aplicado às redes de sensores sem fio, em questões como otimização de roteamento de priorização baseado na quantidade de parceiros e carga de bateria (MASCOLO et al., 2006) , ou ainda baseados em qualidade dos sinais sem fio captados versus quantidades de saltos até o destino (WENNING et al., 2009).

Outra proposição de arcabouço propõe métricas como posicionamento do nós, direção do movimento, e tempo de atividade (YASAR et al., 2010), enquanto no trabalho de (PETZ et al, 2012) o arcabouço proposto envolve métricas relacionadas a taxas de transmissão, e equilíbrio entre os vizinhos conectados.

Já no trabalho apresentado por (CORREDOR et al., 2011) é proposta uma abordagem de controle intermediário baseada em *Middleware* para uma estrutura de rede de sensores sem fio, em um cenário voltado para redes hospitalares. Nesta proposta o trânsito de dados entre

os sensores e o *Middleware* em formato JSON, ao passo que APIs Restful propiciam a comunicação com a gerência geral do sistema através da nuvem.

No trabalho apresentado em (LAUWERS et al., 2014), compreendendo Redes Definidas por Software (SDN) é proposta uma arquitetura que envolve o uso de APIs provenientes das aplicações ditas unificadas, onde um serviço automatizado de APIs baseadas em métricas de QoE atua sobre o Controlador de Rede (Network Controller). Através de mecanismos de engenharia de tráfego dinâmicos, políticas e controle de admissão e marcação dinâmica de QoS, é realizada a interação entre a camada de serviços de rede e o controlador SDN, a fim de computar rotas e programar fluxos de dados em função do QoE, uma abordagem em linha com a abordagem de contexto pesquisada neste trabalho.

2.4 CONCLUSÃO

Neste capítulo foram abordadas questões conceituais acerca dos fundamentos que estão relacionados com as principais métricas de QoS e QoE empregadas neste trabalho, dentre as quais destacam-se os parâmetros MOS e R-Factor, além dos fatores conceituais referentes ao QoC e que são determinantes a fim de definir os princípios da arquitetura e a própria lógica de validação de QoC que será adotada. Também foram destacados alguns fatores relacionados ao QoD, compreendendo uma relação de *codec*s de áudio e vídeo e sua influência no gerenciamento de contexto.

Foram elencadas as principais arquiteturas e tecnologias hoje adotadas pelos provedores de serviços de telecomunicações, devendo ser ressaltado que o MPLS ainda possui ampla utilização atualmente. Por outro lado, a ideia da separação do plano de controles do plano de dados é também central no caso do GMPLs, aplicado às redes óticas. E em paralelo foi avaliada a arquitetura proposta pelas redes definidas por software (SDN) e suas implicações quanto aos sistemas de gerenciamento.

Os estudos aqui realizados revelam uma ampla gama de implementações de estruturas de redes WAN que deverão ainda coexistir por um tempo indeterminado, devendo portanto serem consideradas no âmbito das arquiteturas planejadas para controle e gerência que venha a ser proposta. Por outro lado, os estudos aqui realizados não somente subsidiam as métricas que serão empregadas posteriormente neste trabalho, mas também justificam a necessidade de

abordarmos o problema da gestão otimizada de tráfego sensível a partir de um nível de abstração mais elevado, o que poderia ser também entendido como uma "gestão da gestão".

3 ARQUITETURA DE ENCAMINHAMENTO BASEADO EM CONTEXTO

3.1 INTRODUÇÃO

Neste capítulo é apresentado o arcabouço para o roteamento adaptativo baseado em contexto, denominado *Context-aware Adaptive Routing Framework (CAARF),* sua estrutura modular e a visão operacional do seu sistema de controle e o modelo de contexto proposto para a descrição do estado do usuário, dos dispositivos e das condições de tráfego.

A solução proposta para este arcabouço (*framework*) apresenta-se como uma camada de gestão e encaminhamento que admite uma grande flexibilidade na sua implementação. Além de considerar técnicas e conceitos de engenharia de tráfego abordadas no Capítulo 2, esta solução adota uma visão holística ao incorporar uma ampla gama de variáveis que não podem ser consideradas na engenharia de tráfego tradicional. Dessa forma, é possível descrever um nível de abstração da camada de rede que o torna bastante flexível, possibilitando ainda em tese o seu acoplamento a estruturas de redes gerais, como redes definidas por software (SDN) assim como redes mais amplamente disseminadas, como aquelas baseadas no roteamento exclusivamente IP (“IP puras”), MPLS, GMPLS e suas variações.

A flexibilidade oferecida pelo modelo de contexto ao CAARF adequa-se a um cenário de utilização bem mais amplo em relação aos controles realizados exclusivamente com QoS (normalmente empregados em engenharia de tráfego) ao considerar também o estado dos dispositivos e a perspectiva de experiência do usuário.

Além de apresentar uma visão geral do arcabouço CAARF, este capítulo também dá ênfase ao módulo de Gerência de dados de contexto, em particular à validação destes dados e na efetivação de um controle de fluxos de dados orientado à Qualidade de Experiência (QoE).

3.2 CAARF: CONTEXT-AWARE ADAPTIVE ROUTING FRAMEWORK

O *Context-aware Adaptive Routing Framework (CAARF),* conforme apresentado na Figura 3.2, foi projetado para ser acoplado a diferentes estruturas de redes, e permite adotar diferentes visões para projetos que visam a sua utilização. Poderá, por exemplo, funcionar com um sistema de controle e gerência para apoio à tomada de decisões, interagindo com a

camada de controle de uma rede SDN, ou ainda embutir tomadas de decisões de roteamento no próprio gestor de encaminhamento (*Flow Adaptor*).

Em geral, o funcionamento do sistema deve ser transparente, independente da estrutura de rede que está sendo controlada, e não deve implicar na criação mandatória de um nível de dependência para o funcionamento da estrutura de roteamento como um todo.

Na visão deste trabalho, o CAARF é um sistema (refere-se à implementação do arcabouço de mesmo nome) que deve funcionar primordialmente como uma *Middleware* (camada lógica mediadora) de coleta ampla de dados dotado de inteligência aplicada à resolução de decisões de encaminhamento baseadas em contexto.

O grau de interoperabilidade oferecido pela solução permite que para cada domínio de rede os equipamentos deste domínio continuem sendo configurados por meio de seu sistema de gerenciamento e controle tradicional. Por exemplo, no caso de redes SDN é desejável que o sistema de Controle de Contexto interaja com o plano de gerência já existente, ao invés de substituir o plano gerencial já existente.

Para viabilizar o funcionamento do sistema de controle e gestão de encaminhamento são necessários os seguintes elementos:

- Coletadores de dados dos dispositivos de rede, das aplicações em execução e dos usuários relacionados, e;
- Centralizadores dos dados de contexto, disponibilizando apenas os dados relevantes para análise, e;
- Tomadores de decisão quanto ao encaminhamento baseado no contexto, e;
- Elementos que apliquem as novas regras definidas nos dispositivos de comutação e roteamento ou que notifiquem sistemas de gerência pré-existentes.

Na próxima seção é apresentado o modelo conceitual proposto para a representação do contexto (ou estado) associado às métricas de Qualidade de Dispositivo, Qualidade de Serviço e Qualidade de Experiência.

3.3 MODELO BASEADO EM CONTEXTO

O modelo baseado em contexto para roteamento adaptativo deve ser aplicável a redes de dados implementadas com tecnologias e topologias diversas, tanto daquelas sem-fio (Wi-Fi) como cabeadas. O modelo proposto é plenamente extensível e neste trabalho são apresentadas extensões que visam dotar o sistema de uma visão orientada à experiência do usuário, com foco dado pela Qualidade de Experiência (QoE).

Cabe salientar que extensões podem ser elaboradas objetivando o enfoque em outras visões do sistema, como a Qualidade de Dispositivo (QoD), por exemplo, que pode ser exercitado em situações onde a qualidade do dispositivo seja um fator de maior peso ou preponderante, como no caso de aplicações críticas em sistemas hospitalares, exemplificado em (Corredor et al., 2001). Este modelo, admite, portanto, diferentes cenários e extensões modulares em função de seu enfoque e que podem ser desenvolvidas face ao tipo de aplicação, criticidade e diferentes tipos de ambientes de utilização.

O modelo de contexto proposto descreve o estado particular de uma entidade, i.e, um usuário e a aplicação, englobando infraestrutura de transporte e encaminhamento que provê a interligação fim-a-fim, os dispositivos computacionais envolvidos e até mesmo dados ambientais. Desta forma, as seguintes características descrevem esta entidade (OLIVEIRA et al. 2015), dispostas no diagrama da Figura 3-1:

1. Individualidade– descreve uma informação particular a respeito de uma entidade, como identificadores, endereçamento, protocolos, etc.;
2. Tempo– descreve a informação temporal, como o *timestamp* relacionado ao estado de um parâmetro de uma entidade em dado instante;
3. Localização – é relacionado com a localização real, ou virtual de uma entidade, e que pode ser gerada por um sistema dotado de dispositivo de localização *Global Positioning System (GPS)*, apontando para uma referência geográfica. Também pode designar localidades cadastradas e válidas para um determinado serviço disponibilizado por uma aplicação;
4. Atividade – que permite a descrição de tarefas e ações realizados pela entidade;

5. Relações – descreve as relações entre a entidade com outras entidades, relações de dependência e conexões entre pessoas, usuários, clientes, conexões, lugares e serviços/aplicações;

6. Qualidade de Experiência (QoE) – descreve a perspectiva do usuário, compreendendo a sensação do mesmo acerca do serviço, baseado em parâmetros coletados ou outros métodos diretos e indiretos;

7. Qualidade de Dispositivo (QoD) – é relacionado com um grupo de parâmetros que descrevem características e capacidades de um determinado dispositivo, englobando precisão dos sensores disponíveis, capacidade computacional, de comunicação e aderência a requisitos de interação;

8. Qualidade de Servico (QoS) – está relacionado com todas as métricas de rede qualitativas e quantitativas consideradas no SLA entre o usuário e a plataforma, tais como largura de banda de comunicação, atrasos, perdas de pacotes, variação de atraso *(jitter)*, entre outros;

9. Precisão – nível de acuidade da informação que pode ser fornecida por um determinado elemento de coleta;

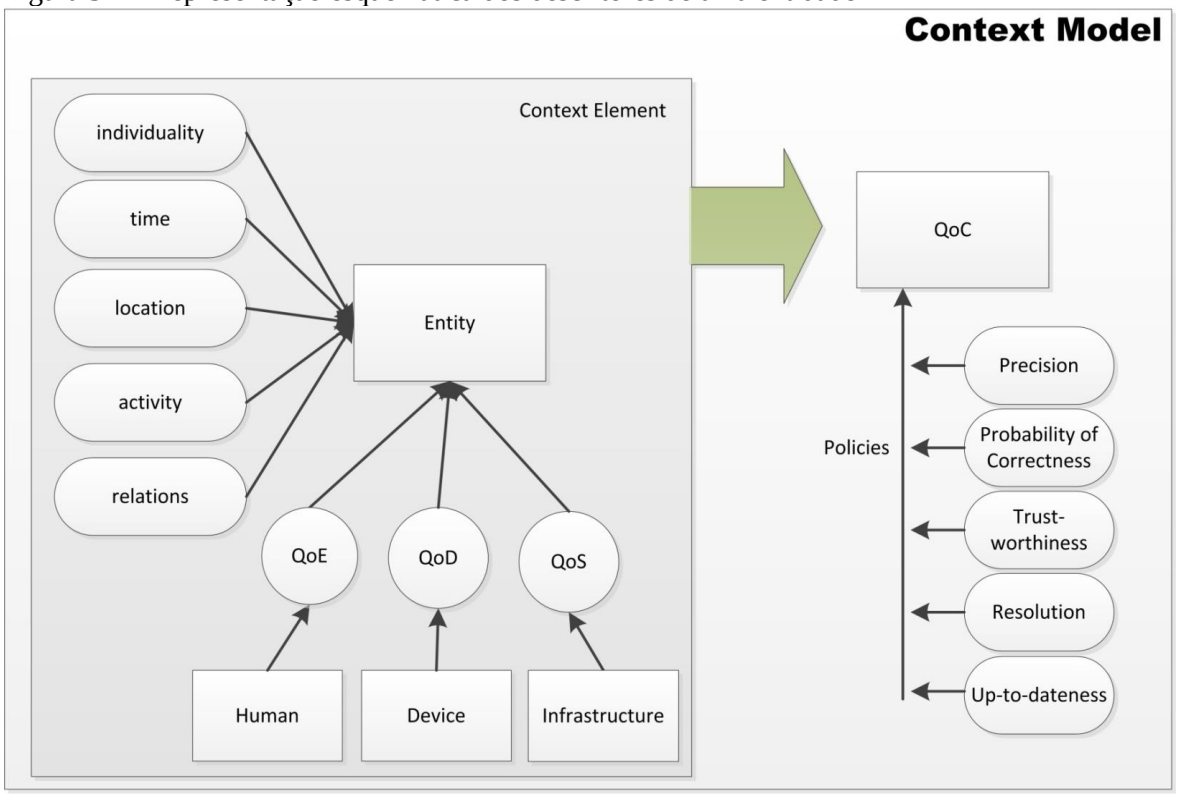
10. Probabilidade de Certeza – avaliação da probabilidade de uma informação estar correta;

11. Confiabilidade – avaliação do nível de confiabilidade de uma determinada fonte de informação;

12. Resolução – nível de granularidade de uma determinada informação, e;

13. Atualidade – avaliação da temporalidade de uma informação.

Figura 3-1 - Representação esquemática dos descritores de uma entidade



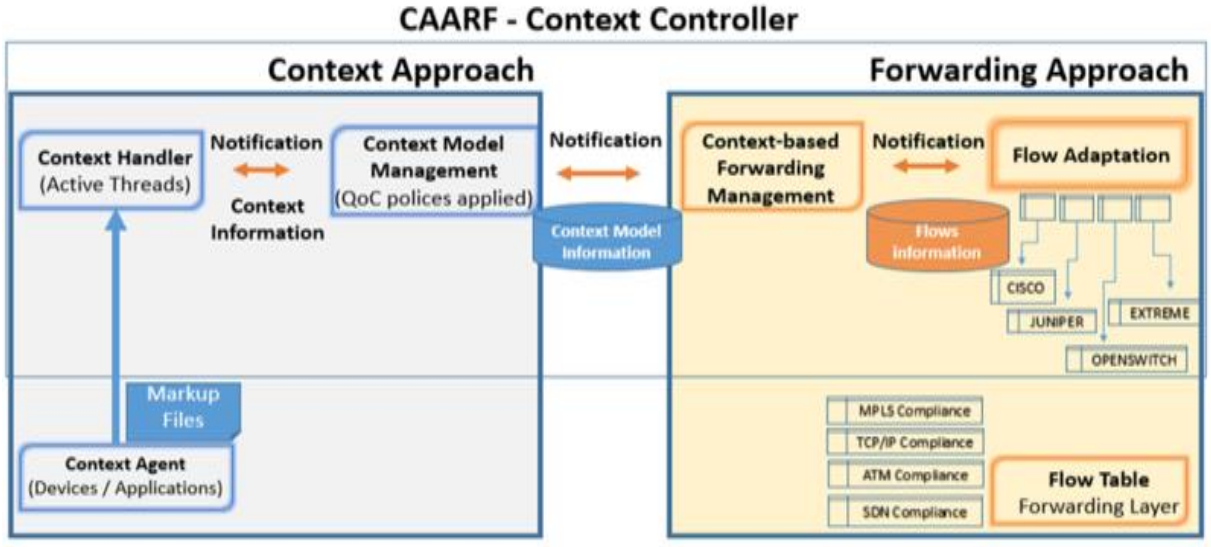
Fonte: Oliveira (2014).

O sistema CAARF permite a representação de dados e troca de mensagens através de linguagens de marcação, como no formato XML e a representação de dados através da implementação em banco de dados relacional (RDBMS) (OLIVEIRA, 2015), (MUAKAD, 2015). O modelo admite, contudo, formas alternativas de representação e formatos de trocas de mensagens e notificações, conforme apresentadas nas contribuições desta dissertação.

3.4 CONTEXT-AWARE ADAPTIVE ROUTING FRAMEWORK

A Figura 3-2 apresenta um representação esquemática do arcabouço, denotando os módulos principais. Cabe notar que utilizou-se a nomenclatura com termos da língua inglesa para denominar os módulos e componentes do sistema, a fim de manter a coerência com as denominações adotadas em trabalhos científicos publicados internacionalmente (OLIVEIRA et al., 2014).

Figura 3-2 - *Context Controller*, Módulos e Sub-módulos



Desta forma, o módulo de gestão de contexto foi nomeado de *Context Approach* (Abordagem de Contexto) e o de gestão de encaminhamento, também conhecido como *Forwarding Approach (*Abordagem de Encaminhamento). O sub-módulo responsável pela coleta dos dados de contexto por meio da interação com o *Middleware* e alimentador de dados na base de contexto é denominado *Context Handler* e o sub-módulo responsável pela validação de QoC e QoE, bem como por gerar as notificações ao módulo de Abordagem de Encaminhamento é denominado *Context Model Management* (Gerenciamento do Modelo de Contexto), abreviado para Gerenciamento de Contexto na sequência desta dissertação. Assim os termos em inglês são empregados quando são referidos os módulos e sub-módulos que remetem às figuras e diagramas que contêm nomenclaturas na lingua inglesa e na terminologia em língua portuguesa quando a sua função é descrita.

Após a descrição inicial do modelo de contexto e as definições referentes aos formatos adotados para a troca de informações entre os elementos da arquitetura, segue-se a descrição do fluxo de processamento da informação contextual que determina a atualização do sistema de encaminhamento a fim de incrementar a performance global da rede. A arquitetura do sistema proposto foi concebida com base em dois macro grupos funcionais:

- Abordagem de Contexto (*Context Approach*) – é responsável pela aquisição de dados, filtragem e qualificação da informação contextual, e;
- Abordagem de Encaminhamento (*Forwarding Approach*) - responsável pelo processamento das notificações encaminhadas pelo *Context Approach* e aplicação das regras de encaminhamento sobre a estrutura de redes.

- As principais funcionalidades da arquitetura proposta são:
  - Coletar e compartilhar a informação contextual obtida de dispositivos de rede;
  - Formatar e centralizar o armazenamento de informação contextual;
  - Realizar a validação de informação contextual baseado em políticas de qualidade de contexto e;
  - Prover notificações acerca da informação contextual capturada de forma a atualizar o roteamento para serviços sensíveis ao contexto.

  O arcabouço CAARF foi concebido baseado em uma integração de diferentes sub-módulos funcionais conforme explicitado a seguir:

  - ***Context Agent:*** é o agente de contexto, responsável por receber informação contextual de dispositivos de comunicação ativos, aplicações e usuários. Estas informações são encaminhadas ao módulo de gerenciamento de contexto, módulo macro *Context Approach*;
  - ***Context Handler****:* reponsável por receber a informação de contexto proveniente do agente de contexto (no *Middleware*), validar o formato, prepará-las e registrá-las na base de dados de contexto, no interior do módulo de gerenciamento de contexto;
  - ***Context Model Management****:* responsável por acessar a informação contextual, processar a qualidade do contexto (QoC), aplicar regras de QoE, e tornar a informação disponível ao módulo de gerenciamento de encaminhamento;
  - ***Context-based forwarding management****:* Pertence ao macro módulo do *Forwarding Approach* e é responsável por processar as regras de encaminhamento baseado nas regras de encaminhamento baseadas na informação contextual, e;
  - ***Flow adaptation****:* responsável pela notificação da informação atualizada do encaminhamento atuante sobre o dispositivo de roteamento e comutação.

  A seguir são elencadas as tarefas pertinentes a cada sub-módulo. Acerca do ***Context Approach:***

  - ***Context Handler***: responsável pelo tratamento das informações de contexto:
    - Coleta as informações do *Context Agent*;
    - Filtra a consistência dos dados coletados;

- Realiza a atualização/inserção das informações no banco de dados do *Context Model Management, e;*
- Notifica o *Context Model Management* sobre a existência de novas informações para processamento e validação do QoC.

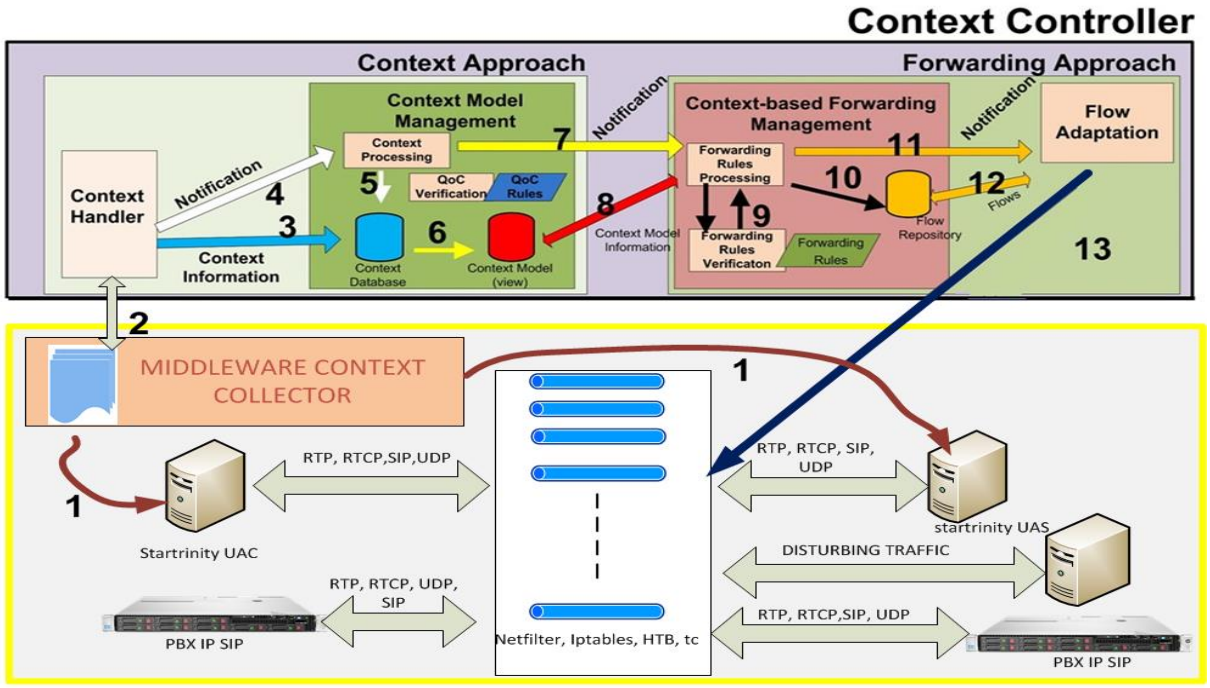
- ***Context Model Management***: responsável pelo gerenciamento do modelo de contexto:
  - Centraliza as informações de contexto;
  - Centraliza as informações de usuário e sessão;
  - Aplica as políticas de qualidade de contexto;
  - Realiza a validação de QoC;
  - Gera Logs de violação de QoC;
  - Realiza a validação de QoE, e;
  - Notifica o *Context-based Forwarding Management* acerca da existência de novas informações de processamento e necessidade de alteração de rotas.

3.5 ABORDAGEM OPERACIONAL

Conforme ilustrado na Figura 3-3, o funcionamento dos módulos é descrito através de uma abordagem operacional e ilustrado através da aplicação do sistema a um determinado cenário hipotético. Neste cenário um par de PBXs IP, compatíveis com o protocolo SIP, localizados em diferentes redes estabelecem um tronco de comunicação. O tráfego entre os dois PBXs atravessa um *gateway* comum, que suporta *traffic shaping*. Este é um caso particular, que é suportado pelo módulo de adaptação de encaminhamento (*forwarding module*). Conforme já mencionado, o módulo de adaptação de encaminhamento foi concebido com o propósito de suportar variadas tecnologias e protocolos, viabilizando o controle de tráfego nos casos já elencados no capítulo 3, como serviços de provedores baseados em MPLS, GMPLS, SPB, redes definidas por software (SDN), Openflow e também redes baseadas apenas em IP.

Neste tipo de cenário um sensor genérico pode executar a tarefa de coletar os dados referentes ao fluxo de chamadas SIP, analisando o fluxo RTP, a fim de obter os dados de QoS e QoE (contexto).

Figura 3-3 - Diagrama funcional e operacional aplicado a um cenário de Telefonia IP



Neste exemplo as violações de QoE ou QoS são avaliadas após o processamento de informações contextuais, que poderão eventualmente determinar uma notificação que motivará a adaptação de fluxo. A seguir são elecadas as etapas do processo, numeradas de 1 a 13:

1. ***Middleware Context agents*** periodicamente coletam a informação contextual a partir das aplicações e elementos da rede, como tempo, QoS, QoE e QoD entre outros parâmetros;
2. ***Context Middleware*** prepara informação contextual devidamente formatada para o **context handler,** que alscuta as mudanças;
3. ***Buffer management***, registra toda a informação recebida no ***context database*** (repositório localizado no ***context model management***);

4. ***Notification scheduler***, submódulo do ***context handler***, notifica o ***context model management*** sobre o recebimento de informação contextual;

5. ***Context Processing,*** submódulo do ***context model management***, recebe as notificações referentes a novas informações contextuais e recupera a informação do ***context database***, e as submete ao módulo de verificação de qualidade de contexto ***Quality of Context (QoC) verification***, que aplica as regras de QoC pré-definidas;

6. ***Context processing***, submódulo do ***context model management***, registra um novo contexto dentro do ***global context model***;

7. ***Context processing***, submódulo do ***context model management***, notifica o modulo do ***context-based forwarding management*** sobre modificação de contexto e violações de QoE;

8. ***Forwarding rules processing***, submódulo **context-based forwarding *management***, recebe notificações acerca de novo contexto e o resultado de consultas ao ***global context model*.**

9. ***Forwarding rules verification***, submódulo do ***context-based forwarding management***, realiza as consultas no ***global context model*** e aplica regras pré-definidas de encaminhamento;

10. ***Forwarding rules processing***, submódulo do ***context-based forwarding management***, registra os fluxos atualizados no ***flow repository***;

11. ***Forwarding rules processing***, submódulo do ***context-based forwarding management***, notifica o ***flow adaption*** acerca de uma atualização disponível para ser aplicada aos roteadores ou gerência de infraestrutura;

12. ***Flow adaptation*** verifica novas regras de fluxos descritos no repositório contido no ***context-based forwarding management***, e;

13. ***Flow adaptation*** aplica novas regras sobre a estrutura de roteamento.

3.6 VALIDAÇÃO DE QOC

Após a aquisição de novos dados de contexto apreendidos pelo ***Context Handler***, e a subsequente validação dos dados adquiridos e uma vez realizada a atualização da base de dados, é gerada uma notificação direcionada ao ***Context Model Management***, que então atua para realizar a validação de QoC. Esta validação consiste na aplicação das regras de QoC definidas no sistema. Estas regras são definidas como documentos registrados na base de

dados e são resolvidas em lógica de programação. O processo de validação de QoC é distinto da validação praticada pelo ***Context Handler***, que se restringe a descartar valores mal formados, notadamente com erros de tipagem ou que possam gerar tratamentos de exceção no sistema.

Já a validação de QoC rastreia informações incongruentes, discrepantes, inconsistentes ou contraditórias. A seguir são apresentados alguns exemplos de Regras de QoC (Tabela 3-1), aplicáveis neste sistema, e que servem de base para a constituição de novas regras. O sistema deve, portanto, permitir a edição contínua de novas regras.

Tabela 3-1 - Regras de QoC

Evento	Critério	Tratamento
Status do Processador do Endpoint do usuário muito baixo	Confrontar com a tabela de requisitos/capacidade de processamento da aplicação.	Se não conforme, descartar e gerar log.
Leitura da localidade inconsistente	Conformidade com a tabela de localidades/aplicações/usuários.	Se não conforme, descartar e gerar log.
Tempo coletado discrepante (validade temporal)	Se o tempo coletado está distanciado do relógio referencial do NTP superior à margem de tolerância.	Se acima da tolerância, descartar e gerar log.
É informado o MOSu (pelo usuário) e MOS (pela rede)	Calcula-se a margem de tolerância para a diferença.	Adota-se a média das duas medidas ou arbitra-se um dos dois.
Leitura das portas SIP e RTP fora do padrão	Verifica-se se a faixa está fora do padrão de admissibilidade.	Descartar a leitura e gerar log.
Resolução em pixels fora do padrão	Verificar a conformidade com a especificação da aplicação.	Descarte e log.
Leitura dos parâmetros de QoS (jitter, packet loss, latência, round-trip-delay e R-factor/MOS presentes simultaneamente	Verifica-se a discrepância nas leituras destes parâmetros.	Verifica-se a consistência. Se houver discrepância, opta-se pelos dados de QoS.
Ruído de fundo em decibels (variável ambiental discrepante)	Se usuário reportar MOS baixo, verificar se esta leitura deve ser descartada.	Se o QoE computado estiver muito baixo, os dados de contexto são descartados para efeito de QoE. É gerado um log.
Quantidade de chamadas simultâneas elevada	Verificar se a quantidade de chamadas excede o padrão estabelecido para aquele perfil de usuário.	Se a quantidade exceder, gera log de sistema e não gera notificação de melhoria de performance, ou utiliza regra específica definida para aquele usuário, e.g, nível de tolerância por um determinado

Evento	Critério	Tratamento
		período de tempo.
MOS/R-factor em conformidade com o SLA e usuário reporta MOSu baixo	Verificar o Codec utilizado e a presença de amostra de áudio.	Gerar um log e uma ação subsequente de alteração de *Codec*. Pode gerar um alarme para o gerente do sistema.
Codec de áudio informado fora do especificado	Verificar se o *Codec* é admissível para verificação de qualidade.	Se Codec é fora da especificação, os dados não são tratados e é gerado um log.
Codec de vídeo especificado fora do especificado	Verificar se o *Codec* é admissível para verificação de qualidade.	Se *Codec* é fora da especificação, os dados não são tratados e é gerado um log.
Banda disponível abaixo do requerido	Verificar se a banda é admissível para verificação de qualidade.	Se a banda estiver fora dos requisitos, descartar dados e gerar log.
Aplicação não cadastrada	Se aplicação não cadastrada, aplicar regra padrão.	Descarta e gera log.
Protocolo divergente	Se o protocolo informado, por exemplo, TCP, for diferente do esperado, e.g, UDP.	Descarta e gera log.
Sobrecarga de CPU	Verificar a regras padrão para a situação.	Gera log somente.
Erro (genérico) de Dispositivo	Verificar o tipo de erro e o respectivo comprometimento.	Gera log somente.
Erro de sistema	Verificar a regra aplicável para cada tipo de erro de sistema.	Gera log somente.
Erro de Rede	Informado erro de rede, como indisponibilidade de serviço.	O sistema notifica o Gestor do Encaminhamento sobre erro de rede, sem notificação de elevação de nível de serviço.
Codec Inválido e MOS_u informado	A critério do administrador do sistema, o MOS informado pelo usuário pode ter precedência sobre Codec Inválido.	O sistema pode notificar o Gestor do Encaminhamento para efetuar o *Best Effort* (melhor esforço).

3.7 VALIDAÇÃO DE QOE

Após realização da validação de QoC, e a devida qualificação dos dados de contexto, se estes forem considerados válidos, serão encaminhados para a validação de QoE. Esta validação consiste na aplicação das regras de QoE definidas no sistema e que devem considerar os parâmetros de Qualidade de Experiência definidos no sistema, para cada conjugação de

usuário e respectivas aplicações. A seguir são apresentados alguns exemplos de regras de QoE, aplicáveis neste trabalho, e que servem de base para a constituição de novas regras. O sistema deve, portanto, permitir a edição contínua de novas regras:

- Baseada em MOS – Neste caso, o sistema compara o MOS apurado pelo *Handler* com as métricas definidas no SLA do sistema. Em caso de violação de MOS, é gerada uma notificação para o Módulo de Encaminhamento;
- Baseada em R-factor – Se somente o R-Factor estiver disponível, é possível calcular o MOS a partir deste parâmetro, recaindo na regra de encaminhamento baseado em MOS;
- Inferida do QoS – através da aquisição dos parâmetros de QoS, se somente estes estiverem disponíveis, é possível calcular o MOS a partir dos dados de QoS, cujos cálculos estão detalhados no Anexo I;
- Entrada do Usuário – Se a aplicação estiver preparada para receber a informação de MOS do usuário, esta pode ser considerada, em função das regras definidas para aquela cojugação usuário/aplicação para efeito de notificação ao módulo de encaminhamento;
- Inferida na aplicação – além de *Experience Sampling Method (ESM)*, se a aplicação estiver preparada para inferir a experiência do usuário a partir da forma como ele está usando o aplicativo, é possível também alimentar a base de contexto com esta informação, como MOS da aplicação, e;
- Inferida de métricas de Qualidade de Vídeo (VQM Monitor) Video MOS – Medidas de Video MOS podem sinalizar o módulo de Encaminhamento acerca de uma violação de QoE de vídeo. Neste caso, o módulo de encaminhamento deverá conjugar as informações de codec de vídeo utilizado com as características de QoS a fim de qualificar a rota que compreende os enlaces apropriados aos requisitos de um dado codec. Por exemplo, no caso do H.264, se utilizar SVC a aplicação de vídeo será mais tolerante à perda de pacotes, dado que diferentes técnicas produzem diferentes resultados de QoE para um mesmo QoS.

3.8 EVENTOS DE GESTÃO DE CONTEXTO

Segundo Pressman (2006), em geral, os eventos ocorrem sempre que um sistema ou um ator trocam informações, sendo o evento o fato de trocar informações, e não as informações trocadas.

Para o CAARF, alguns eventos ocorrem e são tratados internamente em um mesmo componente do arcabouço, enquanto outros podem desencadear ações em outros componentes. A descrição completa dos eventos está contida no Apêndice E, que compreende todos os eventos. O escopo deste trabalho tem seu foco concentrado no agrupamento de funções referente a Abordagem de Contexto (*Context Approach*) e a seguir são destacados os principais eventos relevantes para o foco de visão abordados neste trabalho.

- **São eventos do *Context Agent*:**
  - **Ativação de um agente:**
    - Decorrente da inicialização de um Serviço/aplicação;
    - Inicializa uma identificação de fluxo: flow_id;
    - Envia as informações coletadas para o *Context Handler*.

  - **Coleta de informações:**
    - Coleta informações sobre a estação *(host)*, o usuário corrente, parâmetros do dispositivo e informações sobre as aplicações de rede que estejam realizando conexões remotas, isto é, relações com outros *hosts*;
    - Coleta informações sobre a aplicação, e;
    - Se existir uma interface de interação direta com o usuário, coleta informações fornecidas pelo usuário nesta interface.

  - **Interação do usuário:**
    - Considerando que existe uma interface de interação direta com o usuário no agente, do tipo *Experience Sampling Method (ESM)* e que esta foi habilitada;
    - Gera informações para o *Context Handler*.

- **São eventos do *Context Handler*:**

    - **Processamento do Contexto:**
        - Notifica ao *Context Model Management* a necessidade de processar as alterações do contexto existentes no *Context Database*.

    - **Atualização do *Context Handler*:**
        - O *Context Handler* é notificado pelo Agente de Contexto acerca de novos dados coletados do *probe*/sensor e que precisam ser tratados pelo *Context Handler*;
        - O *Context Handler* é notificado pelo *Context Model Management* sobre a liberação de documento de contexto, após processamento ou;
        - O *Context Handler* é notificado pelo *Forwarding Approach* sobre a liberação de documento de contexto, após processamento.

- **São eventos do *Context Model Management*:**

    - **Processamento do Contexto:**
        - Executa o sub-módulo *QoC Verification* do *Context Model Management* para checar se existem novas informações no repositório *Context Database* para aplicar as políticas do *QoC Rules*;
        - Executa o sub-módulo QoE, aplica as regras definidas para verificação de QoE e, caso ocorra uma violação, notifica novo contexto para o módulo de encaminhamento através da FIFO de notificações.

Dentre os dados que foram considerados para efeito de coleta neste trabalho, destacamos as seguintes informações de contexto coletáveis:

- Informações que individualizam a entidade:
    - Usuário;
    - Aplicações associadas ao usuário;
    - Nome de Registro SIP;
    - Endereço físico da interface de rede padrão do equipamento;
    - Endereço lógico da interface de rede padrão do equipamento.

- Informações que auxiliem na identificação temporal:
    - Hora local referente aos dados de coleta.
- Informações que auxiliem na localização:
    - Coordenadas GPS;
    - Endereço de rede do equipamento;
    - Localidade;
- Informações que descrevam as atividades em curso:
    - Identificação das relações ativas, e;
    - Relações ativas com conexão a serviços existentes.
- Informações que detalhem as relações (conexões remotas de rede):
    - Nomes informados pelas aplicações;
    - Endereços IPs dos equipamentos remotos;
    - Portas origem das conexões;
    - Portas remotas das conexões;
    - Faixa de portas utilizadas;
    - Protocolos de transporte utilizados nas conexões, e;
    - Protocolos de aplicação utilizados nas conexões.
- Informações sobre o QoE:
    - Identificação das relações que tem dados de QoE disponíveis;
    - Média dos escores de opinião calculados (*R-Factor*);
    - Média dos escores de opinião informados (MOS);
    - Nível de ruído no ambiente, e;
    - Identificação do Probe/coletador utilizado.
- Informações sobre QoD:
    - Uso ou não de dados GPS;
    - Endereços lógicos das interfaces de rede, e;
    - Codecs de áudio ou vídeo utilizados.
- Informações sobre QoS:
    - Identificação do Sensor ou Probe coletador de dados;
    - Média de vazão de cada interface;
    - Perda de pacotes das interfaces;
    - Média dos atrasos no caminho ida-e-volta de cada interface, e;
    - Média das variações de atraso (jitter) de cada interface.

3.9 CONCLUSÃO

Por meio dos modelos e do arcabouço aqui expostos e analisados é possível incorporar uma ampla gama de dados de contexto nas decisões de roteamento, permitindo elevar o grau de inteligência aplicável nas decisões de roteamento e uma aplicabilidade do arcabouço aqui apresentado a uma ampla diversidade de cenários e topologias de redes de dados.

Dentre os objetivos do arcabouço destacam-se a possibilidade de associar a aplicação, a criticidade da operação, o seu perfil de utilização e dados tradicionais de rede (QoS) nas decisões de roteamento, assim como a agregação da experiência pessoal do usuário.

O arcabouço permite abstrair por completo a infraestrutura de roteamento ou opcionalmente considerar estas estruturas através da utilização de adaptadores específicos na camada de encaminhamento.

Por outro lado, a concepção do *Middleware* considerada, como uma camada externa que emprega agentes de coleta especializados permite multiplicar a capacidade e escalabilidade da solução, simplificar a sua implementação e deixa perspectivas em aberto para futuros desenvolvimentos acerca do gerenciamento de estruturas de comunicação heterogêneas.

As regras acerca do tratamento de QoC e QoE foram exemplificadas e expostas consolidando as informações necessárias à modelagem da persistência, da lógica de decisão orientada a notificações e da arquitetura de implementação, que devem considerar, no âmbito do Gerenciador de Contexto, a aplicabilidade das regras definidas para o QoC, seguidas do QoE, e que são centrais para a Abordagem de Contexto (*Context Approach*). O próximo capítulo discorre sobre o mapeamento da arquitetura de encaminhamento baseado em contexto aqui descrita através de uma arquitetura de implementação.

4 IMPLEMENTAÇÃO: COLETA E GESTÃO DE CONTEXTO

4.1 INTRODUÇÃO

Este capítulo descreve a arquitetura do sistema de coleta e gestão de contexto, assim como da modelagem de persistência, seguida da descrição esquemática e funcional dos módulos de coleta (*Handler*) e gerenciamento de contexto (*Context Model Management*), além de parte essencial da camada de *Middleware* que foi incorporada a este trabalho, requisito essencial para a validação. Inicialmente são descritos os requisitos funcionais e não funcionais do sistema, assim como da interação do *Middleware* com o Controlador de Contexto, e os respectivos métodos de notificação. Também são descritas a arquitetura funcional e na parte final do capítulo a arquitetura do mecanismo de banco de dados e da linguagem empregada, acompanhada da esquematização da estrutura global da implementação e sua relação com o experimento subsequente. A validação da solução apresentada neste trabalho refere-se especificamente ao comportamento de um sistema de gerenciamento e controle baseado em contexto voltado para aplicações convergentes e a sua interface de notificações para com o Gerenciador de Encaminhamento.

4.2 REQUISITOS DO SISTEMA

Conforme Pressman, (2006), os requisitos funcionais definem as funções do sistema compreendidas por suas entradas e saídas. Desta forma, podemos entender que os requisitos funcionais são as necessidades apontadas pelo demandante da plataforma, i.e, seus *stakeholders* (DUDASH, 2015), ou seja, são as funções que o sistema deve a priori exercer, tais como cadastrar usuários, permitir a inserção de parâmetros de configuração do sistema, realizar cálculos específicos com base em parâmetros coletados, emitir notificações, gerar alarmes e logs, que são exemplos de requisitos funcionais, definidos durante a etapa de levantamento de requisitos do sistema junto ao demandante. Em Robertson e Robertson (2006) um requisito é definido como algo que o produto tem de fazer a fim de atender uma necessidade, ou seja, a qualidade do sistema está centrada na capacidade de atender estes requisitos, uma vez que representam o comportamento esperado pelo demandante.

Os requisitos não funcionais de um sistema dizem respeito as suas características gerais, como o comportamento esperado e suas facilidades, relacionando os componentes externos

necessários ao funcionamento de uma determinada solução. São características não funcionais a performance, usabilidade, plataformas e sistemas empregados, padrões, portabilidade, confiabilidade e segurança, em geral relacionados a fatores como facilidades, desempenho, escalabilidade e estabilidade.

4.2.1 Requisitos Funcionais

Especificamente, no caso do módulo de gerenciamento de contexto proposto neste trabalho, foram levantados os seguintes requisitos funcionais, compreendidos em cinco grupos:

1) Administração do Sistema:

- Gerenciamento de usuários: Cadastramento de aplicações, relação entre usuários e aplicações, requisitos de SLA, tais como aqueles definidos pelo administrador de sistema;
- Possibilidade de definir métricas de qualidade e satisfação por usuário/aplicação/localidade no âmbito do gerenciamento de contexto;
- O sistema deve permitir adicionar, remover e alterar campos na base de dados mantendo a integridade da estrutura do sistema sem requerer a modificação de métodos já existentes;
- O sistema deve ser extensível para a adição de novos campos e capaz de cadastrar e reconhecer dados de registro de usuário referentes ao protocolo de sessão SIP;
- O sistema deve permitir a alteração, inserção, supressão de campos de dados de contexto, e a introdução de novos campos diretamente no sistema de gerenciamento de banco de dados, ou por meio de consoles de sistema com interface GUI, por meio de interface Web;
- O sistema deve possibilitar a inserção de dados cadastrais referentes à qualidade de dispositivo (QoD);

2) O sistema deve prover funções de aquisição/coleta de dados de QoS e QoE:

- O sistema deve adquirir dados referentes à QoS e QoE a partir da leitura de dados de aquisição obtidos de ***probes*** (sensores) de terceiros;

- O sistema deve proporcionar um método para interagir com a camada de *Middleware* (responsável pela coleta junto a sensores);
- O sistema deve ser capaz de coletar dados de percepção de usuário;
- O sistema deve ser capaz de coletar dados temporais;
- O sistema deve possuir uma interface para a coleta de dados de localização apreendidos pelo *Middleware*;
- O sistema deve ser capaz de realizar cálculos e inferências em relação à conformidade de QoE, QoS e QoC:
- O sistema deve coletar parâmetros de QoS do usuário/aplicação a partir de dados coletados de variação de atraso (*jitter*), perda de pacotes, atraso fim-a-fim, entre outros;
- O sistema deve ser capaz de inferir o QoC a partir de regras pré-definidas na base de dados;
- O sistema deve ser capaz de permitir a edição de regras de QoC;
- O sistema deve permitir a inserção de novas regras de QoC preservando a estrutura do sistema, e;
- O sistema deve ser capaz de coletar dados de MOS e R-factor.

3) Logs e notificações:

- O sistema deve ser capaz de gerar notificações de violação de MOS/R-factor (QoE) para o *Gerenciador de Encaminhamento* através de uma fila de notificações para efetivar esta comunicação, isto é, sinalizar eventuais violações de QoE, e;
- O sistema deve ser capaz de gerar Logs de atividade e violações de QoC e QoD na base de dados de Log do *Context Management*.

4) O sistema deve ser capaz de iniciar e finalizar as sessões:

- O sistema deve inicializar as sessões internas a partir da sinalização de início de fluxo – flow_id, por parte do *Middleware*, e;

- Após um período de inatividade, os documentos de representação devem ser excluídos e as sessões finalizadas. O sistema dever ser capaz de realizar a exclusão de dados automaticamente, por meio de regras definidas pelo administrador de sistema.

4.2.2 Requisitos não funcionais

A seguir estão descritos os requisitos não funcionais do presente sistema.

- Usabilidade: o sistema deve oferecer uma documentação (tutorial) destinada aos futuros desenvolvedores de aplicações de gerenciamento orientado a contexto, descrevendo a forma de representação, suportada pelo banco de dados BSON (BSON, 2015), permitindo editar a base de dados de contexto diretamente, seja por meio de um editor de textos como um Notepad++, Wordpad ou similar. O sistema de representação de contexto deve poder ser alocado em serviços na Nuvem – banco de dados como serviço, oferecendo ao usuário formas de acessar a base de dados através de uma interface de uso geral *Graphical User Interface (GUI)*. Deve ser possível também acessar a base de contexto através de utilitários apropriados ao padrão de banco de dados empregado. A documentação fornecida deve incluir o roteiro necessário à implantação de novas adições e incrementos de funcionalidade, facilitado pelo uso de uma linguagem de alta legibilidade;

- Confiabilidade: a confiabilidade do sistema de representação é dada pelos recursos inerentes ao banco de dados que suporta:

    - Replicação e Recuperação de falhas, permitindo o *roll-back* do sistema ao estágio anterior mais seguro. O sistema deve ser escrito em linguagem de alto nível, possuindo cobertura de “try/exception”, protegendo as chamadas críticas do sistema contra erros de execução provenientes de dados mal formados, discrepantes, ou fora de escala;

    - Confiabilidade ditada pela capacidade do sistema tratar exceções e se recuperar de falhas, sem que haja perda de dados. Backup e restauração do banco de dados também se encaixam neste requisito;

    - Checagem de consistência é inerente ao sistema e diz respeito não somente à consistência interna do sistema, mas também dos dados adquiridos. Desta

forma, dados não conformes ou fora de escala e padrão são descartados, evitando a propagação de ações relacionadas a eventos falsos, e;

- A perda de dados poderá ser evitada através da ativação de recursos de replicação do banco de dados, opcionalmente alocados.

- Segurança: acesso ao banco de dados e gerência do sistema protegidos por usuário e senha. Em última instância, a comunicação com o banco dados poderá ocorrer via túnel SSH ou SSL. *Middlewares* externos deverão ser autenticados de forma segura para poderem iniciar a comunicação com a Abordagem de Contexto.

- Interoperabilidade: representação de mensagens compatíveis entre os módulos e que perpassem através das representações em banco de dados, fim-a-fim;

- Escalabilidade**:** o banco de dados deve ser expansível horizontalmente através do particionamento de grandes bases de dados em partes menores, mais rapidamente acessíveis e facilmente gerenciáveis;

- Modularidade: os módulos e sub-módulos devem possuir funcionamento independente e poder interagir através de um sistema comum de troca de mensagens.

- Portabilidade: Os módulos devem ser portáveis entre diferentes plataformas de sistemas operacionais, tais como Linux, Windows e MAC OS;

- Características de desempenho:

  - Tolerância a intervalos temporais de violação de conformidade de desempenho admissíveis, devendo ser definíveis pelo administrador;

  - Representação de Dados orientada a documentos, comportando a representação em forma de coleções (*collections*) e cujo formatação possa ser mantida fim-a-fim, por meio de documentos, evitando fases intermediárias de *parsing* conferindo melhor desempenho;

  - O sistema deve possibilitar usar o histórico de uso para solicitar uma faixa de banda de comunicação preventivamente;

  - É desejável que o tempo de reação do sistema, considerado a partir da aquisição dos dados de coleta e a eventual notificação seja inferior a 8 segundos, meta estabelecida para este trabalho. Este valor foi obtido a partir de

> uma meta estabelecida de 99.9% de disponibilidade para um sistema VoIP, considerando 86,4 segundos de tribulação diária, e que a quantidade máxima de ocorrências de violação de conformidades de qualidade seja inferior a 10/dia.

4.3 MODELAGEM DA ARQUITETURA

O sistema implementado é composto por módulos distribuídos que atuam em paralelo e de forma autônoma, executando diversas funções de sistema. São constituintes principais da arquitetura os seguintes módulos:

- *Gerência do Controlador de Contexto*
- *Middleware* Collector
- Interface *Middleware*/Coletor
- *Context Handler* (Coletor do *Handler*)
- *Context Manager*
    - Validação de QoC
    - Validação de QoE
- Notificação ao *Gestor de Encaminhamento*

Esses módulos passam a ser apresentados a partir da próxima seção compreendendo as respectivas descrições e correspondem à visão geral da arquitetura ilustrada na Figura 4-1 e os fluxos de operações relacionados, ilustrados na Figura 4-2.

Figura 4-1 - Visão geral da arquitetura do sistema

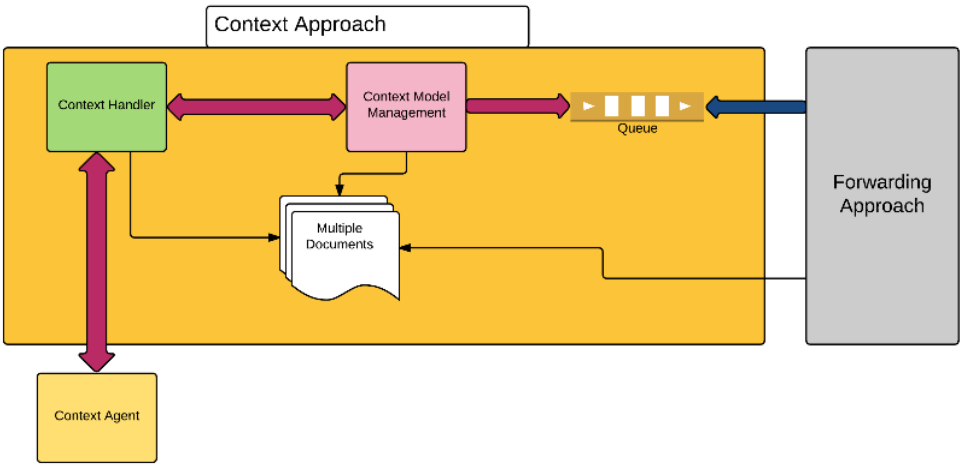
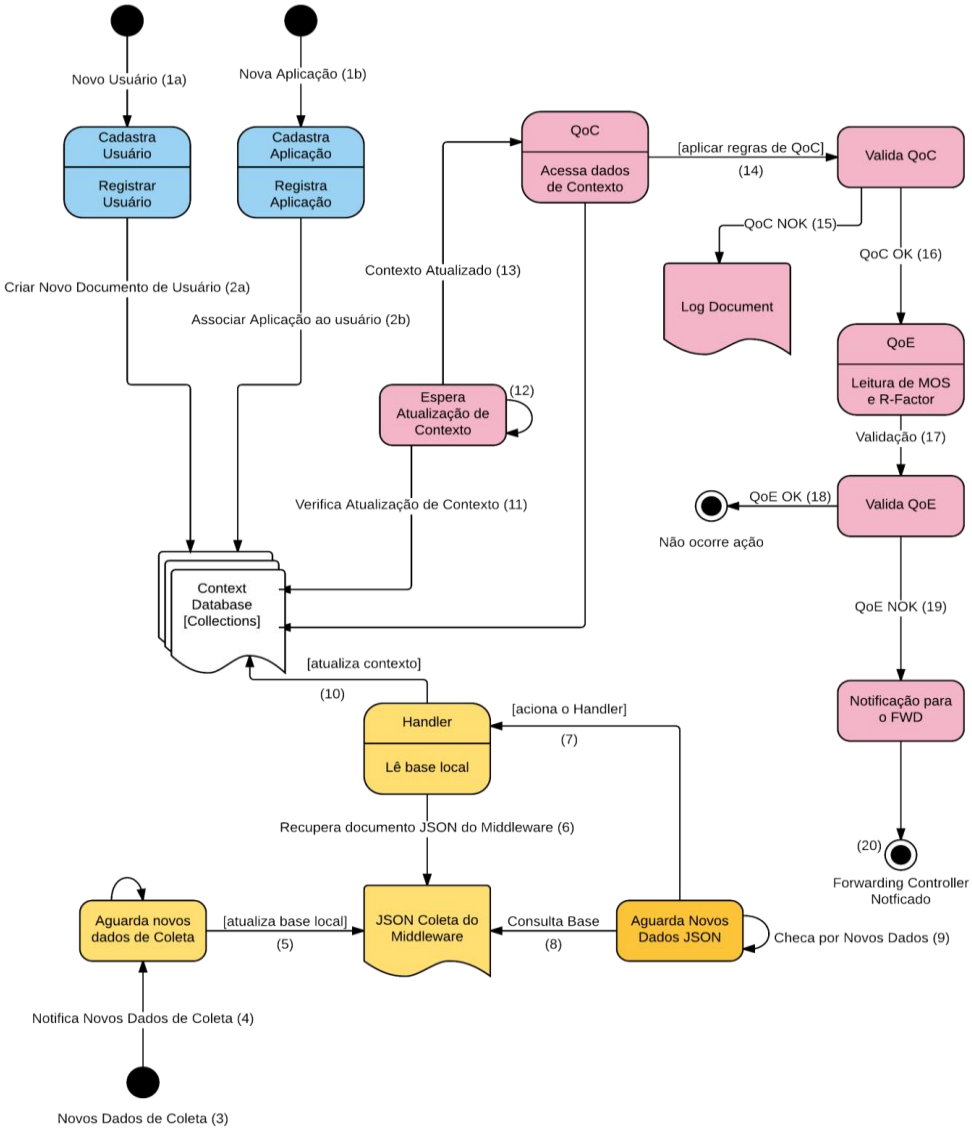


Figura 4-2 - Arquitetura do sistema: fluxo operacional



4.3.1 Gerenciamento do Controlador de Contexto (Administrativo)

Um gerenciamento de sistema para o contexto foi implementado a fim de permitir a inserção adição e alteração de usuários e aplicações a eles associados. O fluxo de operações do sistema, que é operado pelo administrador corresponde às operações (1a, 1b, 2a, 2b) indicadas no diagrama da Figura 4-2. O documento objeto de manipulação associado a este conjunto de operações reside no *Context Database* como uma coleção “User-Profile”, onde estão registrados e definidos todos os dados de usuário, estabelecer as metas de SLAs e SLSs para usuários/aplicações, bem como dados de localização, dispositivos e aplicações cadastradas.

4.3.2 Gerenciamento do Controlador de Contexto (Sistema)

Uma sessão no *Context Management* é iniciada quando ocorre uma notificação de início de fluxo pelo *Middleware* (7). Na inexistência de fluxo prévio, é iniciada/criada uma nova sessão. O *flow_id* é então informado pelo Middleware, que pode ser baseado, por exemplo, no *obj_id* criado na base local BSON do Middleware. A sessão só tem início se o usuário e dados de endereçamento IP são reconhecidos pelo sistema. Desta forma, nesta abordagem, a sessão relacionada com o gerenciamento do contexto, presente no documento de contexto de um dado usuário, é associada com o início de outra sessão por parte da aplicação (SIP, por exemplo) iniciada por esta, que por sua vez estará associada a um serviço como o de telefonia IP, ou um serviço de vídeo-conferência.

A sessão (SIP) que determina o início de uma aplicação é distinta da sessão interna do *Context Controller*, embora esta somente possa ser iniciada após a inicialização de um serviço (aplicação). Estes parâmetros de sessão, em conjunto com a sinalização de início de fluxo por parte do Agente de *Middleware* determinarão o início da sessão interna do *Context Controller*.

4.3.3 Agente Coletor (Middleware Agent Collector):

Embora não seja parte integrante da arquitetura interna do *Context Controller*, sua implementação é essencial para a execução da validação e é um elemento essencial ao funcionamento do cenário dos experimentos, permitindo viabilizar o ciclo de experimentos.

Desta forma, o módulo de coleta do *Middleware* foi necessariamente considerado neste trabalho e implementado de forma simplificada, a fim de propiciar o fornecimento dos dados coletáveis necessários, essenciais à simulação.

Em termos gerais o *Middleware* deve ser entendido como o mediador entre o mundo externo heterogêneo, caracterizado por múltiplas fontes de informação, e o *Context Controller*. A função do *Middleware* é interagir com as diversas ferramentas e dispositivos existentes, como dispositivos de usuário, ferramentas de monitoramento SNMP, *probes* e sensores, e os sistemas de gerenciamento de infraestrutura de redes.

Na presente implementação, o *Middleware* implementado possui os seguintes componentes:

- Coleta: coleta os dados de log da ferramenta de monitoramento, filtrando estes dados (3,4);
- Prepara os dados coletados no formato JSON (5);
- Atualiza a base de dados local de transferência. Esta base pode residir localmente no *Middleware* ou em qualquer domínio de armazenamento em nuvem (5).

4.3.4 Interface entre o módulo de coleta do *Middleware* e o Coletor do *Handler*

O módulo de coleta do *Middleware* constantemente atualiza a base de dados local do *Middleware*, com dados de rede não formatados adquiridos dos sensores (3) e notifica módulo local do *Middleware* acerca de novos dados para coleta concertidos para JSON (4) e atualiza a base local (5). O módulo de interface do *Handler* com o *Middleware* verifica continuamente (8) e (9), se existem novos dados formatados disponibilizados pelo *Middleware.* Se a base de contexto estiver liberada para a escrita de um novo ciclo de coleta, então a base de dados do *Middleware* é consultada (8) e se houver novos dados de coleta dos sensores, o *Handler* é acionado (7) para que possa recuperar estes dados (6) e atualizar a base de contexto (10)

4.3.5 Coletor *(Handler)*

O *Handler* é o módulo responsável por interagir com os Agentes de *Middleware*. O *Handler* reconhece os Agentes de *Middleware* distribuídos pela Rede Internet, devendo cada agente de *Middleware* credenciar-se no sistema a fim de ser reconhecido. O *Handler* realiza a leitura da base de dados JSON (6) de cada Agente de *Middleware* e constantemente verifica se ocorreu uma atualização desta base de informações.

A ocorrência da base de atualizações provoca a ação do *Handler* no sentido de copiar estas informações para a base de contexto principal (10) e sinalizar a mudança de contexto referente a um dado/usuário/aplicação.

O *Middleware* gera um número de flow_id, que irá identificar o fluxo a ser controlado pelo *Context Controller* ao passo que o *Handler* sinaliza na base de contexto que existe uma sessão ativa para tratar aquele Contexto ativo. Enquanto a sessão for válida, o *Context Controller Management* continuará monitorando aquele fluxo. A validade de uma sessão passa ser determinada pela validade temporal da última atualização. Desta forma, fluxos cujo *timestamp*

estejam acima da validade definida pelas políticas definidas pelo administrador do sistema ensejarão o encerramento da sessão.

4.3.6 Gerenciador de Contexto (*Context Manager*): Validação de QoC

O módulo de validação de QoC constantemente monitora a base de dados de contexto a fim de encontrar alterações de contexto associadas a um fluxo e sessão válidos (11,12). Ao encontrar uma alteração do contexto (13) , a função de validação é acionada, a partir do acesso aos dados de contexto, e os dados referentes ao contexto são submetidos ao escrutínio de regras de QoC, conforme definido na seção 3.6 (Regras de QoC).

A função principal do módulo de QoC é validar a qualidade dos dados coletados (14), realizando checagens de consistência, a fim de eliminar discrepâncias, dados que estejam mal formatados, ou assinalando a não conformidade de leitura, descartando dados imprecisos, inválidos, fora de escala, não cadastrados, ou provenientes de dispositivos não conformes com a aplicação (15). Uma vez realizada a checagem, confrontada com as regras de QoC, o módulo de validação de QoE é acionado (16).

Um erro de QoC interrompe o tratamento do fluxo e leva à geração de um log de erro de QoC na base de dados de Contexto (15). O tratamento deste erro será efetivado pela gerência do sistema, que pode computá-lo para fins estatísticos, gerar alarmes de sistema, ou provocar alertas que em última instância podem levar a uma notificação em alto nível do usuário.

4.3.7 Gerenciador de Contexto (Context Manager): Validação de QoE

A presente implementação é direcionada à validação orientada ao QoE, dado que esta é uma métrica que possui correlação direta com a medição de qualidade de soluções convergentes de voz e voz/vídeo em tempo real e interativas.

Os parâmetros mais importantes de validação de QoE são o MOS e R-factor.. A escolha destes dois parâmetros como critério para a tomada de decisões possui também uma motivação de ordem prática, dado que muitas das ferramentas disponíveis para a comunidade de gerência de operações de rede é capaz atualmente de computar estes parâmetros diretamente, o que simplifica e otimiza o processo de apuração desta métrica.

Além disto, é possível agregar a medição realizada pelas ferramentas padrão de mercado com aquelas que são embutidas em aplicações, chamadas de ESM (*Experience Sampling Methods*), que consistem na inserção de plugins de inferência de qualidade de experiência embutidas em aplicativos, tais como softwares de vídeo-conferência, onde tais plug-ins avaliam a experiência do usuário com base na forma como o software está sendo utilizado (perfil de uso).

Alternativamente, o próprio usuário deve ser capaz de emitir uma opinião a respeito da sua experiência em várias situações compreendendo experiências de uso para determinadas aplicações, conduzindo à obtenção do QoE de usuário, definido neste trabalho como parâmetro MOS_u ou *User Perception*.

Estes dados de QoE estarão, desta forma, representados no documento JSON de usuário/aplicação/fluxo. O módulo de QoE realiza a correlação (17) destes dados com o SLA definido para aquela aplicação e, se ocorrer uma violação de MOS, uma notificação é gerada para o Gerenciador de Encaminhamento (19), e de outra forma não ocorre ação de notificação e o documento de contexto é liberado (18). Nesta implementação a notificação ocorre por meio de uma FIFO implementada a partir da capacidade nativa do banco de dados de suportar este tipo de estrutura.

O Gerenciador de Encaminhamento monitora a fila de notificação e, com base nas informações de contexto e mapa de rotas tomará a decisão de escolher uma nova rota que seja capaz de comportar a exigência de QoE referente àquele fluxo.

4.3.8 Notificação ao Gestor de Encaminhamento

Após a Validação de QoE, caso ocorra um QoE não conforme (19), é acionada a rotina que realizar o enfileiramento (FIFO) da notifiação compreendendo o número do *flow_id.* O módulo de Encaminhamento, que ciclicamente monitora a FIFO de notificação recupera esta informação e com esta chave (*flow_id*) possui acesso aos dados de contexto referenciados e poderá então realizar os cálculos de rota necessários a fim de efetivar a recuperação do QoE.

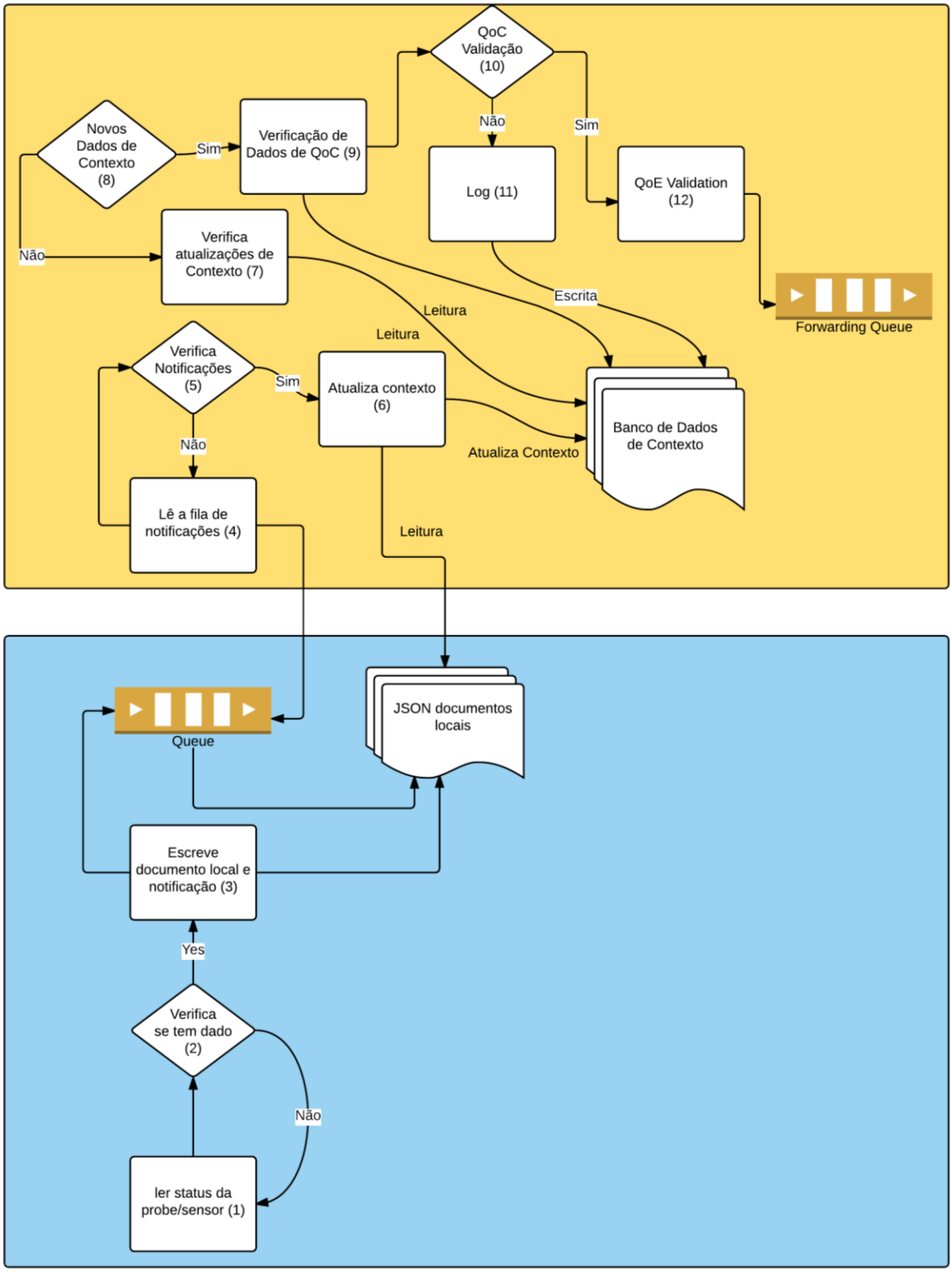
4.4 DESCRIÇÃO FUNCIONAL RELACIONADA AO EXPERIMENTO

O diagrama da Figura 4-3 representa o fluxo das etapas e o ciclo de funcionamento do sistema, contemplando o fluxo de coleta, decisões, atualizações e notificações, e os mecanismos de sinalização traduzidos em código Python, dispostos no Apêndice C, que compreende um tutorial de instalação.

Descrição dos Fluxos e Módulos:

- Leitura do Probe/Sensor pelo Agente de Middleware (1) e (2)
- Formatação local de dados JSON (3)
- Atualização da fila de notificações Middleware/Handler (3)
- Preenchimento dos dados de sessão SIP e flow_id (3)
- Verificação da existência de Notificação pelo Handler (4)
- Atualização do Contexto Geral (5) e (6)
- Verificação pelo Context Management sobre novos dados de Contexto (7), (8), (9)
- Validação do QoC (10) e (11)
- Validação do QoE (12)
- Gera uma fila de Notificação para o Forwarding Controller (12)

Figura 4-3 - Fluxo e ciclo de funcionamento do sistema



Neste diagrama o agente de *Middleware* é o componente externo da solução, cuja função é extrair e inserir dados nos sistemas objeto de monitoramento e validação. O *Middleware* é executado em uma estacão de trabalho externa e deve interagir com outras ferramentas e dispositivos a fim de coletar os dados necessários para o processamento do *Context Controller*. São estes dados coletados:

- Network
- QoS
- QoE
- Dispositivo
- Localização
- Tempo

O *Middleware* pode ser implementado em qualquer linguagem e ser capaz de ler/perscrutar os arquivos de dados gerados por ferramentas externas. Estas ferramentas são em geral *probes* e sensores, que podem ser específicos ou de terceiros.

O *Middleware* deve ser capaz de ler arquivos de logs gerados por estes probes e sensores de fabricantes diversos, de forma a potencializar a capacidade de coleta evitando a necessidade de desenvolver ferramentas de sensoriamento e leitura desde o início. Em termos gerais, a vantagem da abordagem de coleta através do *Middleware* é reduzir o grau de complexidade do *Context Handler*. Para atingir este objetivo, o *Middleware* deve ser capaz de formatar os dados de coleta em JSON e manter estes dados em uma base local.

Neste trabalho, cujo foco é o *Context Handler*, a fim de empreender as notificações entre o Agente de *Middleware* e o *Gerenciador de Contexto*, optou-se pela abordagem de notificações baseada em *flag* de *update*, que pode ser transformada em FIFO, dado que esta é uma facilidade nativa do banco de dados empregado nesta implementação.

Desta forma, o Agente de *Middleware* deve ser capaz de acessar e converter os dados coletados de sensores/probes, e por este motivo deve ser desenvolvido para cada tipo de *sensor*/*probe*, ou ainda embutir seu próprio *sensor/probe*, como, por exemplo, no caso requerido da leitura de dados de georeferência, de parâmetros ambientais de um dado dispositivo móvel, ou ainda ser capaz de interagir com ferramentas que gerem estes dados

diretamente em JSON ou XML. O parsing de XML, nestes casos, deve ser realizado pelo *Middleware*.

No diagrama funcional, ilustrado na Figura 4-2, o *Middleware* realiza a aquisição dos dados e os prepara em uma base de dados local. Da mesma forma, é gerada uma notificação em uma fila local que pode ser acessada pelo Controlador de Contexto. Estas informações são representadas no banco de dados BSON e o acesso a estas bases é protegido por meio de credenciais de acesso.

4.5 MODELAGEM DA PERSISTÊNCIA

A persistência de dados em um sistema é compreendida pela forma como o estado desse sistema é mantido ou conservado durante o seu ciclo de vida, como por exemplo, através do armazenamento não volátil. Neste trabalho foram adotadas a forma de representação JSON para descrição de mensagens e seu correspondente formato BSON, através do banco de dados MongoDB para implementar a persistência dos dados.

Elencamos a seguir os fatores para esta escolha:

- Simplicidade: a representação JSON é direta e simples, e pode ser realizada diretamente através de um editor de texto simples, como o Notepad++. A seguir é apresentado um exemplo de representação JSON, cabendo notar que as representações adotadas neste trabalho seguem uma estrutura aderente ao formato representado na Figura 4-4;

Figura 4-4 - Exemplo de esquema JSON

```
{
        “title": "Example Schema",
        "type": "object",
        "properties": {
                "firstName": {
                        "type": "string"
                },
                "lastName": {
                        "type": "string"
                },
                "age": {
                        "description": "Age in years",
                        "type": "integer",
                        "minimum": 0
                }
        },
        "required": ["firstName", "lastName"]
}
```

Fonte: Json-Schema (2015).

- Compatibilidade: a representação JSON é reconhecida diretamente em código de programa por diversas linguagens, sem a necessidade de realizar “parsing”, como no caso do XML. Por ser suportada nativamente por linguagens como Java e Python, torna possível realizar a transmissão de mensagens fim-a-fim sem necessidade de passos de construção e reconstrução ou a chamada de funções adicionais para se alterar, recuperar, modificar, excluir e inserir valores;
- Flexibilidade/Legibilidade: estes são dois fatores que influenciam diretamente na desempenho do programador e consequentemente, na manutenção do sistema. O banco de dados MongoDB expressa a representação JSON binária. i.e, BSON (BSON, 2015), o que permite traduzir expressões em JSON diretamente em banco de dados;
- Potencial de Representação: a representação JSON/BSON permite expressar as informações do sistema por meio de documentos com um grande grau de liberdade. Esta é uma forma bastante flexível e apropriada de modelagem da persistência, pois permite descrever dados de forma livre, mas sem prejuízo da documentação, sem necessidade de manter esquemas de representação rigorosamente fixos. Em síntese, é possível criar a representação de dados como um documento em um editor de texto simples e realizar a inserção através de uma linha única de script em Python, ou outra linguagem.

Esta representação privilegia o entendimento por parte da equipe de programadores e permite uma rápida compreensão da representação, auxiliada pelo emprego de ferramentas de visualização gratuitas disponíveis no mercado, como o MongoChef (MONGOCHEF, 2015), que permite ler a estrutura do banco diretamente por meio de uma interface GUI.

A Figura 4-5 apresenta a representação esquematizada adotada no documento de cadastro de usuário/aplicação/dispositivo/aplicação, neste projeto, ao nível gerencial. Um único documento registra todos os dados estáticos de cadastro referente a um dado usuário.

Esta é a representação em que o administrador do sistema, no módulo de gerência, aplica as definições de SLA, registro de localidades válidas, lista de dispositivos gerenciados, endereços de rede válidos e todos os dados de cadastro. Podem ser criados tantos documentos e subdocumentos quanto necessários, e cada documento de cada usuário distinto pode variar em quantidade de campos e subcampos/subdocumentos.

Figura 4-5 - Diagrama da representação de documentos e sub-documentos de cadastro de usuário no sistema, sub-documentos e arrays de documentos

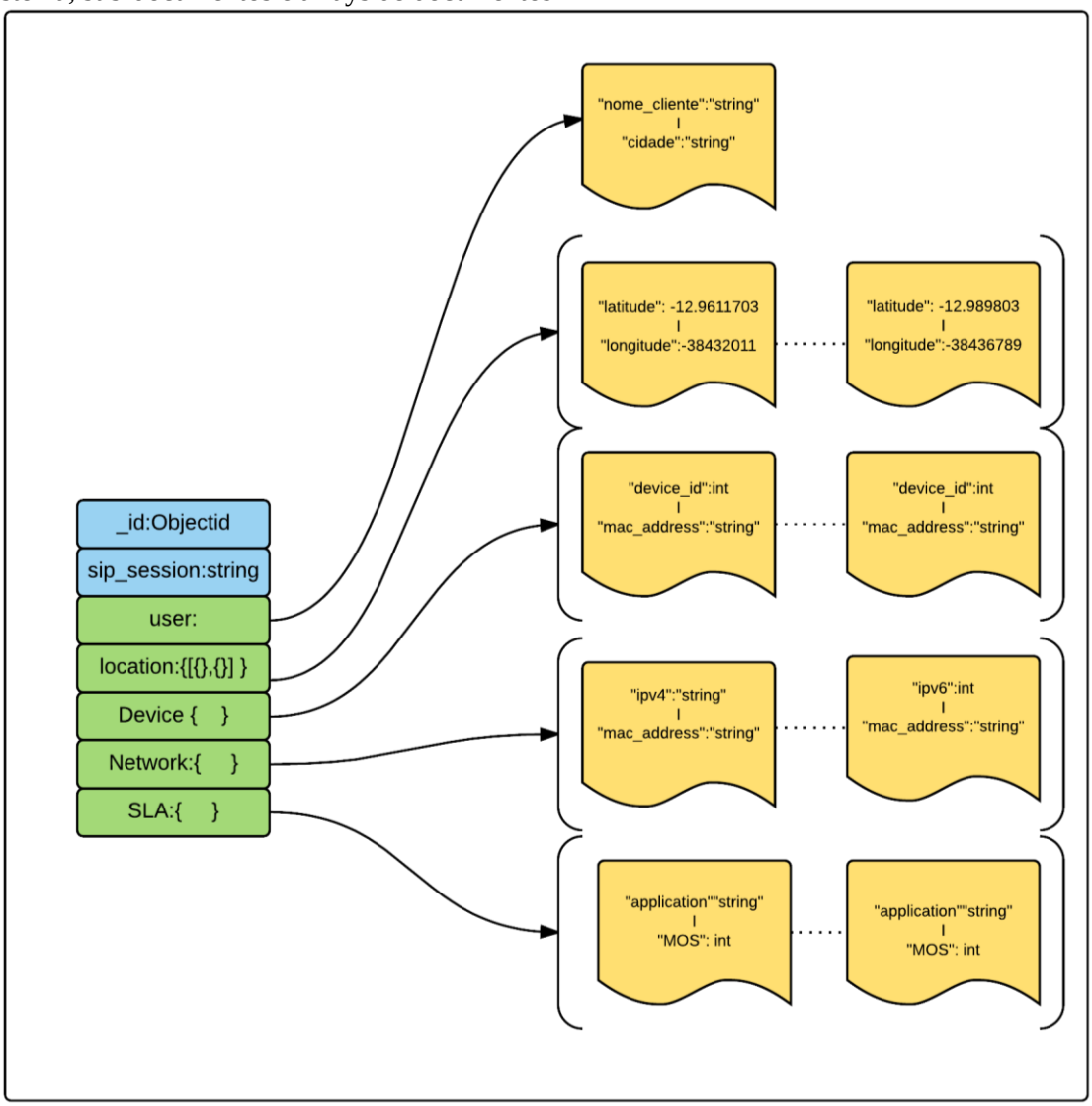


Figura 4-6 - Representações em MongoDB de Usuário, registro SIP, dispositivos e Localização

USER

```
"user" : {
      "nome_do_cliente" : "string",
     "razão_social" : "string",
      "CNPJ" : "string",
      "endereço" : "string",
      "cep" : "cep",
      "pais" : "brasil",
      "cidade" : "salvador",
      "bairro" : "pituba",
      "user_id" : NumberInt(5)
  },
```

SIP REGISTRAR

```
"sip_register" : {
      "sip_user" : "sip:telco@ip10.com.br",
      "rtp_port_ini" : "16000",
      "rtp_port_fim" : "16500"
   },
```

LOCATION

```
"Location" : {
     [ "latitude" : -12.9611703,
      "longitude" : -38.432011],
     [ "latitude": -12.9848484,
       "longitude": -38.456777]
   },
```

DEVICE

```
"Device" : [
     {
        "device_id" : NumberInt(177),
        "device_name" : "string",
        "mac_address" : "cc:a4:62:dd:be:50",
        "mobile" : "true",
        "connection" : "802.1ac",
        "os" : "android",
        "has_gps" : "false",
        "gps_precision" : "none",
        "process_cores" : "two",
        "screen_resolution" : "string"
     },
     {
        "device_id" : NumberInt(155),
        "device_name" : "string",
        "mac_address" : "9e:8c:63:58:96:65",
        "mobile" : "true",
        "connection" : "802.1g",
        "os" : "ios",
        "has_gps" : "false",
        "gps_precision" : "none",
        "process_cores" : "two",
        "screen_resolution" : "string"
     }
  ],
```

Figura 4-7 - Representação complementar de cadastro para dados de rede e SLA

NETWORK

```
"Network" : {
      "ip_v4" : [
         "192.168.1.101",
         "75.82.0.37"
      ],
      "ip_v6" : [
         "FE80::0202:B3FF:FE1E:8329"
      ]
   },
```

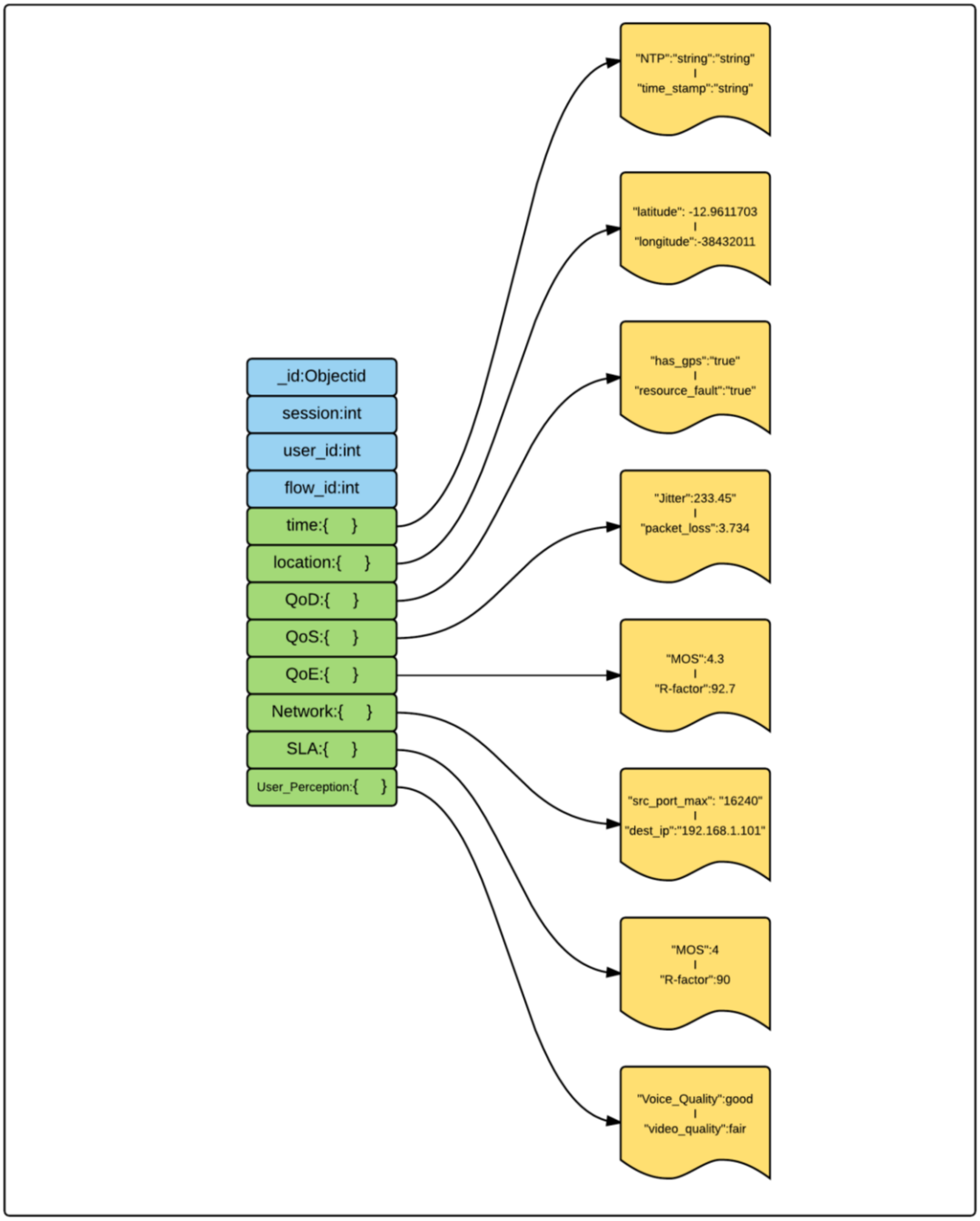
SLA_APP

```
"SLA_APP" : [
   {
      "application_name" : "sip_pbx_call",
      "MOS_SLA" : NumberInt(4),
      "max_calls" : NumberInt(30),
      "Requirements" : [
         "QoE",
         "QoS",
         "Network",

         "sip_session"
      ],
      "audio_codecs" : [
         "G.729",
         "G.711"
      ]
   },
   {
      "application_name" : "sip_conf",
      "MOS_SLA" : NumberInt(4),
      "max_calls" : NumberInt(5),
      "Requirements" : [
         "QoE",
         "QoS",
         "Network",
         "sip_session"
      ],
      "video_codecs" : [
         "H.263",
         "H.264"
      ]
   }
]
```

As Figuras 4-6 e 4-7 ilustram os grupos de documentos e sub-documentos por categoria referentes à representação de dados para um dado perfil de usuário, i.e, respectivamente

dados de identificação de um usuário, registro SIP (quando for o caso), ou outro tipo de registro, localidades válidas, dispositivos cadastrados, como smartphones, por exemplo, localidades de rede e o SLA por aplicação, contendo, por exemplo, a lista de Codecs válidos. É favorecida a expressividade da representação, no sentido de torná-la autoexplicativa, uma das metas dos idealizadores deste tipo de arquitetura de banco de dados e que se procurou potencializar neste trabalho.

Figura 4-8 - Diagrama da Representação dos dados de Contexto para um dado usuário



A Figura 4-8 apresenta o esquema de representação de dados de contexto em tempo real, isto é, aqueles adquiridos pelo *Handler*. Cabe notar que todos os documentos e subdocumentos apresentados na Figura são contidos em um único documento.

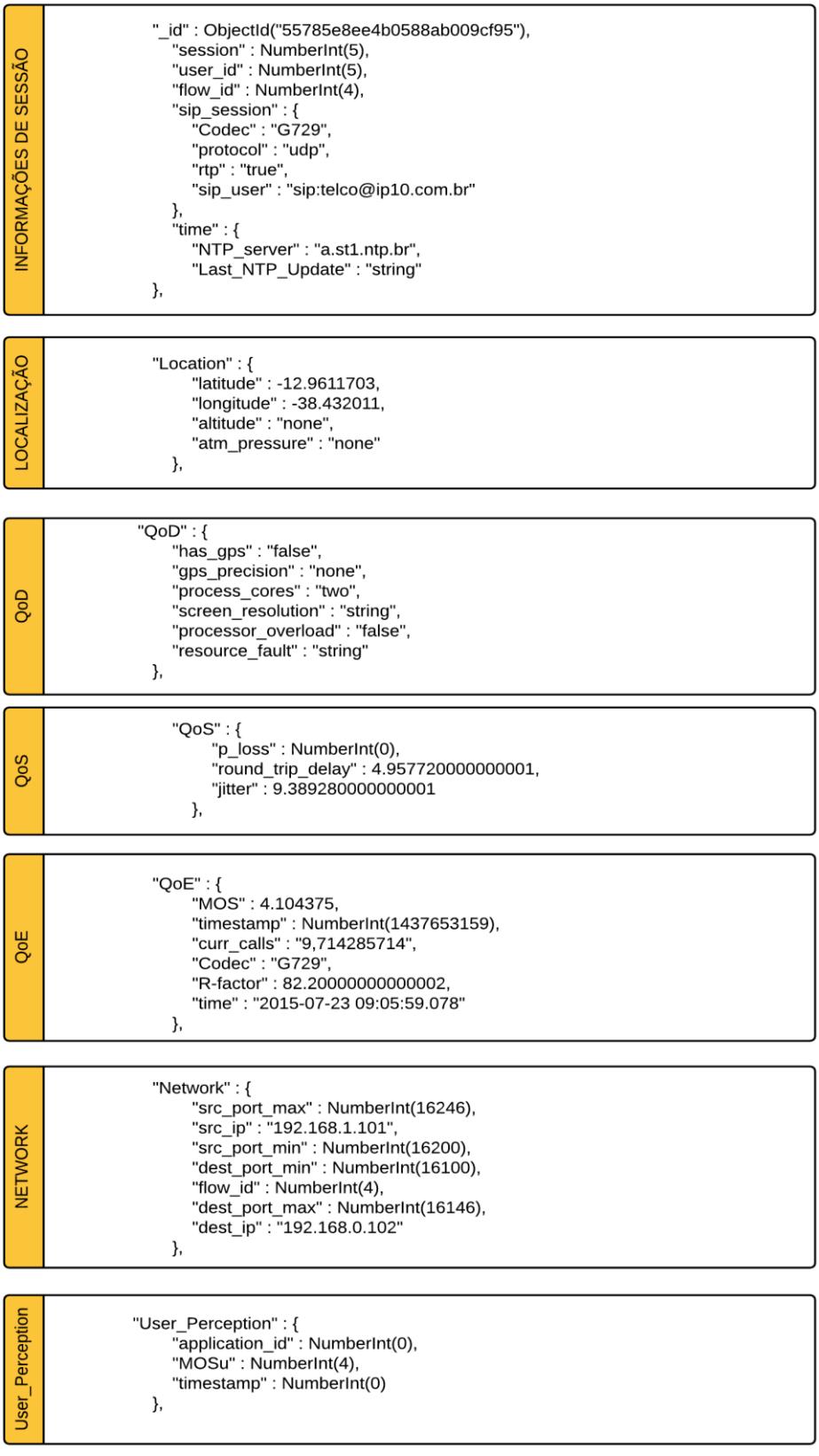
A Figura 4-9 apresenta o exemplo de um documento preenchido da representação em tempo real, agrupados, referentes à representação dos dados de contexto adquiridos pelo *Handler* na sua interação com o *Middleware*.

Dessa forma, a base dados é preenchida com informações de rede, portas utilizadas, da aplicação ativa, da sessão referente àquela aplicação – SIP neste exemplo, a sessão do sistema e o *flow_id*, criado pelo Middleware, o identificador do fluxo corrente, além dos dados de QoE, QoS e dispositivo. Essas informações compõem o documento de contexto que é confrontado com as regras de SLA definidas no documento de cadastro.

Cabe notar que os dados de MOS de usuário, isto é, a medida da percepção de satisfação de usuário, estão presentes no grupo de parâmetros "User Perception". A representação aqui mostrada permite conjugar diferentes perspectivas de avaliação de qualidade.

O MOS pode ser adquirido de outras fontes, como por exemplo, por meio de um "plug-in" no aplicativo, e que pode adquirir a percepção do usuário diretamente, seja pela forma pela qual este usuário interaja com a sua aplicação, seja pela informação direta, isto é, pelo "apertar de um botão" de qualificação de qualidade. Esta possibilidade será apresentada experimentalmente no capítulo 5.

Figura 4-9 - Grupos de Dados de Contexto capturados em tempo real



No Figura 4-10 são apresentadas as representações de regras de QoC selecionadas. Dentre estas são escolhidas para exemplificar esta representação as regras de violação de *timestamp* geral, i.e, quando o novo contexto adquirido foi invalidado por caducidade, e cuja validade temporal é definida pelo administrador do sistema; por meio das definições de desvio; Codec de áudio e vídeo admissíveis; quantidade máxima de ligações permitidas; localização informada não conforme, e; desvio entre a percepção do usuário e o MOS medido.

Figura 4-10 - Representação de Regras de QoC extraída da console local do MongoChef, com base no Mongolab

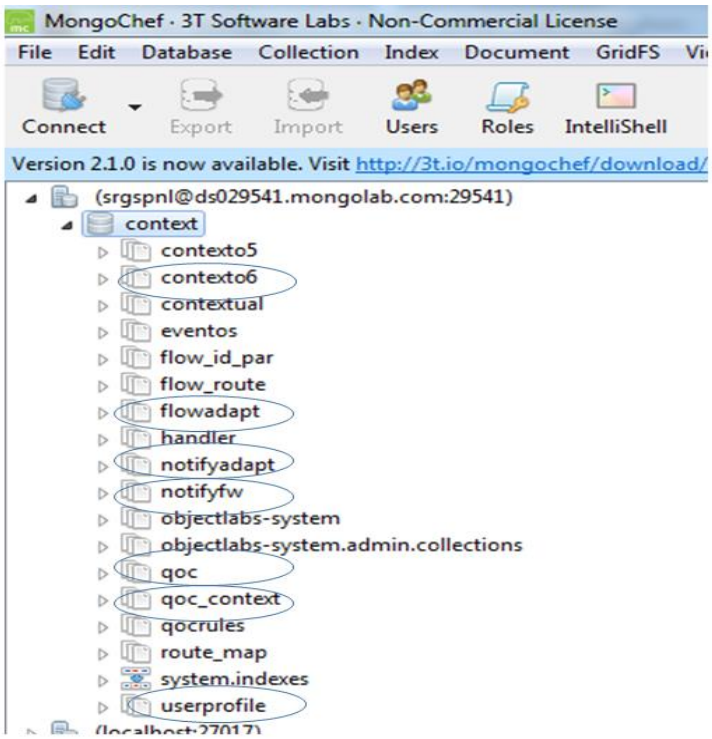

```
{
    "_id" : ObjectId("55c53153736e393aa4b5dbfc"),
    "code" : NumberInt(10),
    "description" : "timestamp violation",
    "deviation" : NumberInt(60),
    "action" : "drop"
}
{
    "_id" : ObjectId("55c53153736e393aa4b5dbfd"),
    "code" : NumberInt(40),
    "description" : "location not valid",
    "longitude" : 0.01,
    "latitude" : 0.01,
    "action" : "alarm"
}
{
    "_id" : ObjectId("55c53159736e39bb70c1f1ce"),
    "code" : NumberInt(100),
    "description" : "User perception mismatch",
    "deviation" : 0.7,
    "action" : "alarm"
}
{
    "_id" : ObjectId("55cf9fce699d8e0344d9b2a2"),
    "code" : NumberInt(60),
    "description" : "video codec invalid",
    "action" : "drop"
}
{
    "_id" : ObjectId("55d4d355e4b095eeb88a5f46"),
    "code" : NumberInt(90),
    "audio-codec" : [
        "G729",
        "G711A"
    ],
    "description" : "audio codec invalid",
    "action" : "drop"
}
{
    "_id" : ObjectId("55e03a4ae4b01b38147dd0e4"),
    "code" : NumberInt(20),
    "description" : "Max concurrente calls",
    "deviation" : 0.1,
    "action" : "alarm and drop"
}
{
    "_id" : ObjectId("55e03abae4b01b38147dd0f0"),
                                        "code" : NumberInt(50),
    "description" : "por range mismatch",
    "deviation" : NumberInt(0),
    "action" : "alarm and drop"
}
```


Nos experimentos apresentados no capítulo 5 serão elencados alguns fatores para fins de atividades de log e selecionado um fator de codec para exemplificar a invalidação da continuidade de um dado processamento. Na Figura 4-10 são descritos alguns destes fatores, como tolerâncias, os valores admissíveis, o código de erro e a mensagem associada.

4.5.1 Tabela de Coleções (*Collections*): banco de dados contexto

A Figura 4-11 apresenta a captura da tela de visualização através do cliente de acesso ao banco no Mongolab da base de dados de contexto e das coleções empregadas pelos módulos *Handler* e *Context Management*. As demais *collections*, i.e, aquelas não destacadas na tela apresentada na Figura 4-10 destinam-se ao suporte do experimento, armazenamento de *traces* ou backup.

Figura 4-11 - Visualização da tela do MongoChef, base de dados e coleções



Desta forma, em relação às coleções assinaladas na Figura 4-11 a Tabela 4-1 apresenta um descritivo das coleções utilizadas e suas representatividades:

Tabela 4-1 – Descrição das coleções empregadas nas representações de dados de contexto e filas

contexto6	É a coleção (*collection*) principal, contempla os dados de contexto coletados pelo Handler em tempo real.
userprofile	*Collection* compreendendo os dados das aplicações, usuários, endereçamentos físicos e lógicos, definições de SLA, permissões e capacidades. Esta coleção é acessada pela Gerência de Administração.
flowadapt	Collection referente aos parâmetros de encaminhamento do Forwarding Approach
notifyfw	Fila de notificação entre o Context Approach e o Forwarding Approach. Por esta fila o Forwarding reconhece qual o fluxo que deverá ser tratado.
qoc	Collection que armazena as violações de QoC detectadas pelo módulo de QoC
qoc_context	Collection que registra as Regras de Contexto

Uma descrição detalhada destas coleções, apontadas nestas seção, pode ser visualizada no Apêndice correspondente às representações completas relacionadas e utilizadas neste trabalho.

4.6 FERRAMENTAS E LINGUAGENS UTILIZADAS

Foram empregadas na realização desta implementação as seguintes linguagens, ferramentas e motor de banco de dados:

- Linguagem Python
    - Versão 2.7.9 da linguagem;

  - driver de conexão entre o Python e o MongoDB, pymongo;
  - Um micro-arcabouço web para Python: bottle.py, e;
  - Um conjunto de bibliotecas detalhado no Apêndice C.

- MongoDB na Nuvem: MongoDB como serviço de banco de dados e nuvem, MongoLab (MONGOLAB, 2015). Versão baseada em livre utilização. O MongoLab impõe apenas uma restrição de 500MB (versão Sandbox);
- Ferramenta Startrinity (STARTRINITY, 2015),utilizada como gerador de ligações e como sensoriamento;
- O *gateway* foi implementado com uma versão Debian do Linux e as bibliotecas empregadas tornam possível a utilização do *Netfil*ter;
- O Editor de texto utilizado para edição de códigos foi o Notepad++, e;
- Ambiente de codificação e prototipação: Pycharm 4.5.

Uma descrição completa das linguagens, banco de dados, sistemas operacionais, instalação, importação e procedimentos relacionados pode ser encontrada nos Apêndices C e D referentes aos códigos de programação e representações no banco de dados.

4.7 DISCUSSÃO E LIÇÕES APRENDIDAS

O desafio inicial enfrentado nesta implementação foi a escolha do sistema de representação de dados aplicado a cenários de alta diversidade e heterogeneidade, motivando o estudo e pesquisa de ferramentas de representação, assim como das opções de linguagem, a fim de levar a cabo a tarefa de implementação de uma forma mais eficiente. Este estudo resultou no levantamento de formas de representação de informações e desenvolvimento de código que terminaram por influenciar substancialmente na modelagem e concepção do projeto.

Estes estudos revelaram que a manipulação do banco de dados por meio de coleções e documentos, distintamente da abordagem tradicional, considerando os cenários a serem manipulados, revelou-se promissora. Por outro lado, esta abordagem exigiu um tempo de aprendizado adicional a fim de se adquirir uma maior versatilidade no emprego desta ferramenta de banco de dados.

Este tempo adicional dispendido na capacitação resultou em um ganho substancial na expressividade e, em termos gerais, na simplificação da tarefa de desenvolvimento, documentação, compreensão e elaboração das representações necessárias.

O uso da notação JSON em substituição ao XML simplificou a implementação e permitiu realização do encaminhamento de mensagens em um formato uniforme, fim-a-fim, entre todos os módulos A simplicidade aparente destas representações visava também tornar efetiva a continuidade deste trabalho rumo a outros projetos derivados, sobretudo por parte de profissionais que não sejam tão versados em linguagens de programação clássicas.

A comunicação com a Gestão de Encaminhamento foi implementada através de uma FIFO de notificações, com referenciamento ao *flow_id*, tendo sido decidido encaminhar notificações do Gerenciador de Contexto para o Gestor de Encaminhamento somente na ocorrência de uma violação de regra de degradação de QoE. Esta opção permite filtrar um excesso de envio de notificações ao sistema de encaminhamento, que passa então a processar o rumo do encaminhamento a partir de cada notificação enviada considerando todas as notificações como verdadeiro positivo.

A decisão de implementar esta filtragem prévia foi discutida entre os implementadores do Gestor de Contexto e do Gestor de Encaminhamento e decidida em favor do processamento prévio nesta abordagem. A alternativa consideraria enviar notificações relativas a cada troca de contexto para o sistema de encaminhamento, o que obrigaria o Gestor de Encaminhamento a realizar um processamento para cada troca de contexto. Contudo, considerando que o Gestor de Contexto obrigatoriamente processa as violações de QoC, consequentemente a sobrecarga para processar as violações de QoE passa a ser reduzido se efetivado no mesmo passo de validação de QoC.

Acrescentamos a estas discussões a questão referente à crescente importância que o acesso à comunidade de desenvolvimento, presente na Internet, e a interação com a mesma, pode acrescer em termos de ganho de produtividade. No caso do banco de dados, o MongoDB University (MONGODB UNIVERSITY, 2015) oferece uma grade de treinamentos gratuitos e exemplos de implementação extremamente valiosos. O mesmo se dá com a comunidade de programadores Python. Estes recursos, e os serviços online disponíveis para abrigar as plataformas devem ser considerados como componente importante de projetos de maior envergadura. Desta forma, foi aprendido que a estreita interação com a comunidade de desenvolvimento é indispensável para alavancar projetos com a maior eficiência e rapidez.

4.8 CONCLUSÃO

A partir da definição do modelo de representação baseado em documentos e *collections* notou-se um grande ganho na obtenção da expressividade combinada com uma melhor interface e menor verbosidade na descrição do Contexto.

Embora o foco deste trabalho seja a Coleta e Gestão no âmbito do Controlador de Contexto, foi verificada a necessidade de aprofundamento acerca da implementação da camada de *Middleware*, essencial para implementar a validação completa e gerar os dados de coleta necessários, pois a obtenção de dados efetivos e reais revelou-se um requisito essencial para a validação do sistema global. Para realizar o experimento em ciclo realimentado e contínuo constatou-se ser necessário um coletor ativo junto ao gerador de chamadas.

O estudo dos parâmetros de QoE e sua adequação ao sistema durante o processo de modelagem levou à conclusão de que uma maior efetividade seria obtida se o tratamento do QoE ocorresse no submódulo de gerenciamento de contexto.

Foi necessário desenvolver uma parcela mínima do *Middleware*, utilizando os conceitos apresentados neste capítulo, o que possibilitou a interação em ambiente funcional do Gerenciador de Contexto em um cenário cíclico, abordado no capítulo 5, de Validação e Testes, no qual as implementações levadas a cabo e descritas neste capítulo foram objeto de validação.

Também foi verificado que a validação do QoE deve ocorrer concomitantemente como a validação de QoC. A ausência de um *Middleware* completo, entretanto, limitou as possibilidades de validação no âmbito do QoD.

Uma validação completa da validação de QoD depende de uma implementação mais completa deste aspecto no *Middleware*, por meio de um sistema de coleta mais completo, embora sua representação esteja contida e prevista nos documentos BSON aqui elaborados. Ainda assim, pelo menos um aspecto desta abordagem foi validado no Capítulo 5. Por meio da utilização do banco de dados NoSQL orientados a documentos a etapa de definição de campos e documentos de representação de dados de contexto foi acelerada, e o objetivo de tornar mais eficiente a representação de uma diversidade complexa de dados foi atingido.

5 VALIDAÇÃO E TESTES

5.1 INTRODUÇÃO

Este capítulo descreve os experimentos aplicados a variados cenários visando a validação da implementação da arquitetura proposta e a demonstração dos conceitos apresentados nos Capítulos 3 e 4. Os experimentos foram realizados em ciclos fechados e realimentados compreendendo a integração efetiva entre os submódulos de Abordagem de Contexto (*Context Approach)* e Abordagem de Encaminhamento (*Forwarding Approach)*, acrescido de um agente de coleta, compreendendo ligações telefônicas IP com protocolo SIP, e baseados em dados coletados em tempo real.

Os experimentos realizados promovem a integração entre todos os módulos, i.e, os módulos de Coleta (*Handler),* de Gerenciamento de Contexto (*Context Management*) e o Sistema de Encaminhamento, os quais interagem com os dispositivos do cenário de experimentação e são objeto de captura de dados por parte de um agente coletor de *Middleware,* implementado especificamente para este cenário.

Nas seções seguintes serão descritos o cenário geral de estudo, as ferramentas empregadas para emular as funções de realização de chamadas de telefonia IP, os sensores utilizados, os elementos integrantes dos cenários objeto de estudo e suas funções, e posteriormente 8 estudos de caso.

Nestes estudos buscou-se dividir a abordagem da experimentação em duas partes. A primeira parte compreende os cenários iniciais de 1 a 4, listados na Figura 5-1, que buscaram validar a capacidade de restauração de qualidade de Experiência (QoE) do sistema. Na segunda parte, englobando os cenários de 5-8,, listados na Figura 5-2, buscou-se demonstrar e validar a efetividade da aplicação de regras de QoE,e QoD bem como validar os mecanismos de qualificação por meio do QoC.

Também se procurou demonstrar o funcionamento efetivo em um ciclo fechado e realimentado, compreendendo a interação efetiva entre os dois módulos principais (Abordagem de Contexto e Abordagem de Encaminhamento) constituintes do Controlador de Contexto, e o funcionamento de uma parcela funcional de um agente de *Middleware.*

Figura 5-1 - Cenários de 1 a 4: Recuperação de MOS

Cenário	Chamadas por Rodada	Progressão de Chamadas Simultâneas	Codec	Controle	Pertubação
1	25	Sim	G.711	IP	Não
2	25	Sim	G.729	IP	Não
3	25	Sim	G.729	IP	Sim (UDP)
4	50	Sim	G.711	Portas	Sim (UDP)

Figura 5-2 - Cenários de 5 a 8: validações de QoC

Cenário	Objetivo	Validação	Regra Conjugada
5	Medir tempo de reação do sistema a partir da aquisição de novo contexto	Não	Não
6	Validar regra de QoD	Codec G.723	Não
7	Validar regra de QoD e MOS de usuário	Codec G.723 e MOS de usuário	Sim
8	Validar regra de QoD, MOS de usuário e temporal	Codec G.723, MOS de usuário e Temporal	Sim

5.2 PLATAFORMA DE EXPERIMENTAÇÃO

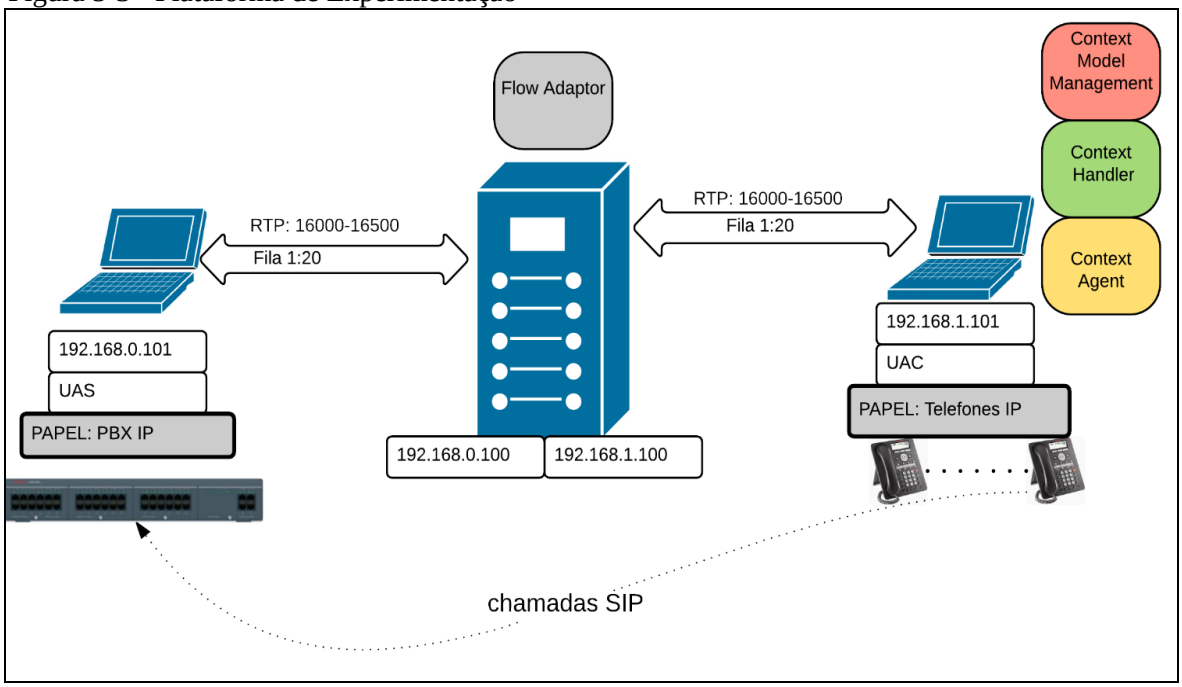
O cenário considerado para o presente estudo de caso é representado por um sistema de emulação de chamadas de voz sobre IP e de um PBX IP emulado, aderente ao protocolo de sinalização SIP, escalável, e compreendendo um fluxo variável de chamadas entre o terminal (*host*) origem – onde se localizam os clientes chamadores, denominados UAC (*User Agent Client*), e o *host* destino, denominado UAS (*User Agent Server*).

Nos cenários descritos, o UAS representa o papel do sistema PBX IP, que é o atendedor das chamadas telefônicas VoIP/SIP. O sistema atendedor e chamador encontram-se localizados em redes distintas, e todo o fluxo de chamadas deve atravessar um sistema de gerenciamento de filas, implementado em um *gateway* cuja banda de largura disponível é variável e controlada pelo Netfilter (NETFILTER, 2015). Essas filas simulam as rotas disponíveis, cuja métrica é a capacidade de banda alocável.

Neste cenários serão criadas diversas situações caracterizadas por diferentes quantidades de chamadas simultâneas, i.e, diferentes cargas entre o *host* origem e o *host* destino, com diferentes volumes de tráfego VOIP, empregando o protocolo SIP de sessão, RTP/RTCP na camada de aplicação e controle dos pacotes de voz e UDP como transporte.

Serão desta forma inseridos dois tipos de pertubação variáveis, a primeira delas é o volume de chamadas, considerando canais de comunicação exclusivos para VoIP, e o segundo caracterizado pela inserção de um fluxo de tráfego concorrente e peturbador a fim de causar interferência na qualidade. A Figura 5-3 ilustra a plataforma de experimentação, em que serão empregados principalmente os protocolos SIP sobre UDP e RTP sobre UDP, além de variados codecs.

Figura 5-3 - Plataforma de Experimentação



Nessa plataforma de experimentação os dispositivos IP realizam múltiplas chamadas de voz, progressivas e variáveis, em ambiente de telefonia IP, com protocolo SIP, para um Server (PBX) com duração de 20 segundos cada. Podem ser visualizados na Figura 5-3 os módulos implementados em Python correspondentes aos sub-módulos do CAARF.

O esquema de realização e atendimento de chamadas compreende a sinalização SIP referente à solicitação, estabelecimento e encerramento de uma chamada de telefonia IP entre um

Cliente (UAC) e um Servidor (UAS). Toda chamada é inicializada com um INVITE e finalizada com um BYE/200 OK, de acordo com a sinalização SIP, normalmente na faixa de portas 5060-5070. O fluxo de pacotes de voz sobre dados ocorre sobre RTP em portas distintas, em geral na faixa 16000-16XXX.

5.3 SISTEMA DE EMULAÇÃO E MEDIÇÃO

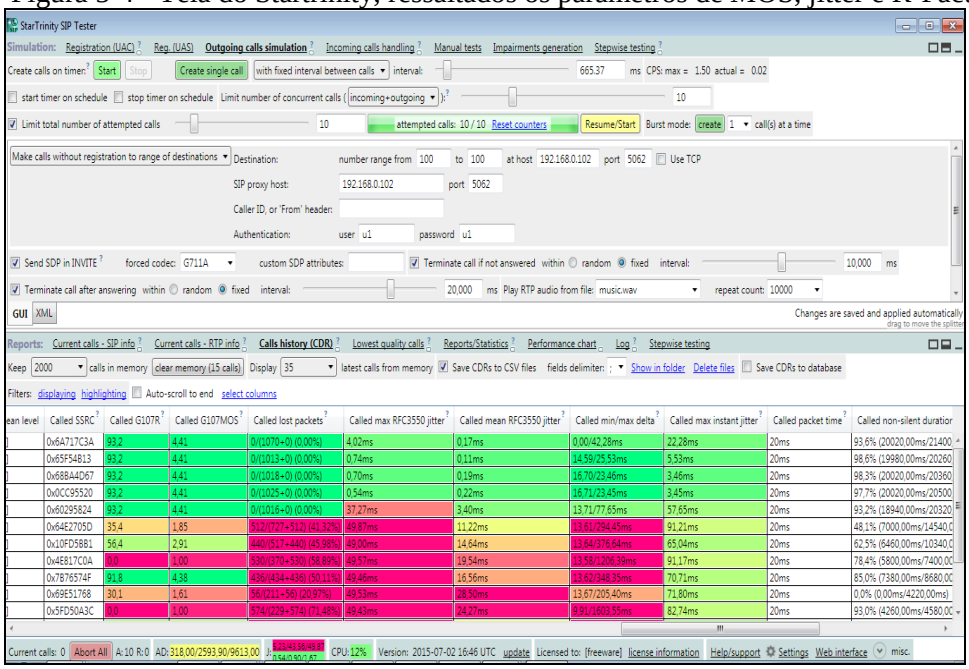
O papel de emulador de chamadas IP e respectivo atendimento é realizado pelo sistema de emulação e medição *Startrinity* (STARTRINITY, 2015), uma ferramenta de simulação de chamadas e atendimento por parte do PBX que exerce o papel de chamador (UAC) e atendedor (UAS), aderente ao protocolo SIP, e que incorpora um *sniffer* (analisador de tráfego) de chamadas específico para VoIP.

Este sistema permite programar a quantidade de chamadas simultâneas, definir diferentes Codecs e também fornece um meio para apurar os parâmetros relacionados às chamadas fim-a-fim, acompanhar o andamento das chamadas *online*, desde o início à sua conclusão e todos os dados referentes à apuração de cada chamada, contemplando dados de QoS e QoE.

Em síntese, este sistema se comporta como um *sniffer* de chamadas e executa as funções de chamada e atendimento exatamente como no funcionamento de um PBX IP padrão. Uma vantagem adicional é permitir simular chamadas com conteúdo de tráfego (*payload*) realístico.

Este tipo de sistema de avaliação de disponibilidade nos canais de tráfego é geralmente empregado por diversos provedores de serviços de telefonia IP e corporações para aferir e ajustar os parâmetros de rede para tráfego IP de voz durante a realização de chamadas em tempo real ou em *offline* como ferramenta de Engenharia de Tráfego. Existem diversos sistemas de gerenciamento que permitem realizar as simulações e coleta de dados, dentre os quais destacamos, por exemplo, o PRTG (PRTG, 2015), OPManager (OPMANAGER, 2015) e SolarWinds (SOLARWINDS, 2015).

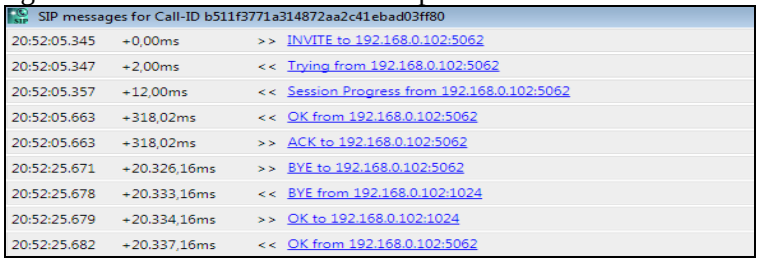
Figura 5-4 - Tela do Startrinity, ressaltados os parâmetros de MOS, jitter e R-Factor



A Figura 5-4 apresenta a apuração e tempo real da qualidade de ligações efetuadas, e os parâmetros MOS, R-Factor apresentados em tempo real, bem como os parâmetros de QoS como *delay*, *jitter* e *RTT*. O *Startrinity* gera um arquivo CDR (*Call Detailed Record*), e tempo real, inserindo dados em um arquivo CSV, o qual é constantemente lido pelo Agente de *Middleware*.

Por sua vez, o *Handler* adquire estes dados e realiza a filtragem, calcula as médias necessárias e filtra os dados mal formatados ou inválidos. Neste momento, são descartadas as ligações não completadas, como aquelas que são terminadas com código 487 (*Request Timeout*) ou 503 (*Service Unavailable*). Estes dados coletados alimentam por sua vez a base de dados de contexto.

Figura 5-5 - O Sniffer de chamadas permite obter o detalhamento de cada ligação



A Figura 5-5 ilustra a sequência de uma chamada bem sucedida, que tipicamente será aquela que fornecerá dados válidos para coleta e que alimentará o sistema na base de contexto. São chamadas que finalizam com o código SIP 200 OK e possuem duração de 20 segundos.

5.4 SISTEMA DE COLETA

A concepção do *Controlador de Contexto* prevê tarefas distribuídas e, especificamente para aquela referente ao sistema de coleta de dados dos sensores, essas tarefas são realizadas pelos agentes de coleta integrantes do *Middleware*.

Para a sua implementação, necessária para todo o experimento, a coleta de dados realizada pelo *Middleware* é composta por meio de rotinas escritas em Python que lêem o arquivo de *Log* de chamadas (CDR) apurado pelo *Startrinity*, tanto em *offline* como em tempo real, fazem a crítica dos dados, filtram as chamadas válidas e fazem a sua tradução e inserção na base de dados de contexto.

Estas rotinas implementam a parte do *Agente de Contexto* e exercem também a parte do papel do *Handler*, que é o módulo responsável por efetivar a inserção dos dados de contexto apurados na base de dados de contexto no MongoDB. Desta forma, um sistema de geração/simulação de chamadas em ambiente real foi configurado, assim como um método de coleta, apuração e preenchimento da base de dados de contexto aplicado a este estudo de caso. Uma máquina Linux Debian, denominada *gateway*, equipada com duas interfaces de rede exerce o papel de gerenciador de filas de encaminhamento, implementada pelo Netfilter.

5.5 CARACTERIZAÇÃO DO EXPERIMENTO

O cenário apresentado na Figura 5-5 corresponde ao estudo de caso que será objeto de validação por parte do *Controlador de Contexto*. O *Context Handler*, juntamente com a rotina de extração de dados do *Startrinity* realiza a filtragem dos dados, e valida aquelas chamadas bem sucedidas (200 OK) – somente, e descarta dados não conformes ou inválidos (mal formados ou fora do padrão esperado), alimentando a base de dados de contexto com as informações referentes àquele fluxo de chamadas.

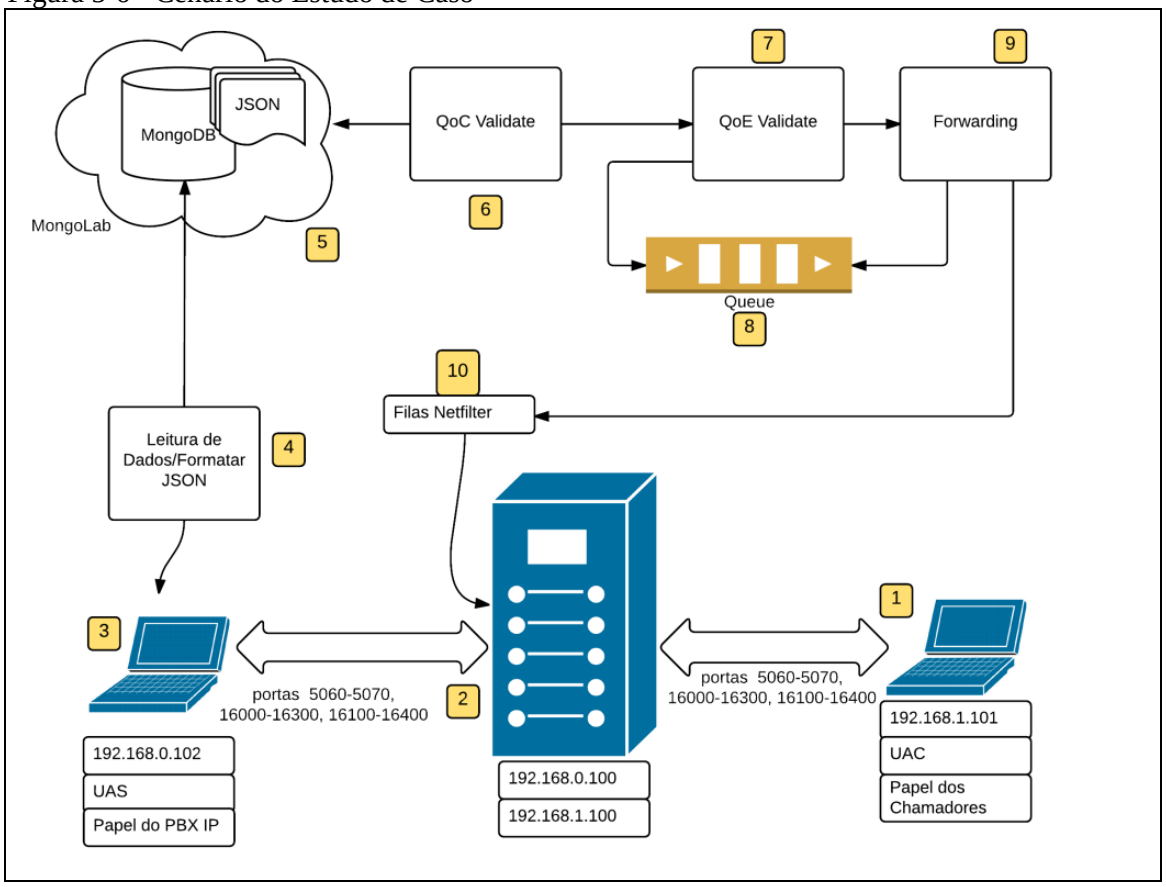
O sub-módulo *Gerenciador de Contexto* irá validar os dados apurados e confrontá-los com as regras de QoC. Especificamente para este experimento serão validados na sequência do experimento os seguintes fatores de congruência:

- *Timestamp;*
- Chamadas concorrentes;
- Faixa de portas RTP em uso (*port range*);
- Aplicacão não registrada (implementado apenas);
- R-factor versus MOS;
- Percepção de usuário, fator humano (*User Perception*);
- Codec válido/inválido;
- Localização (implementado apenas), e;
- QoS versus MOS/R-Factor (QoE).

Na interpretação dada para este estudo de caso, a *Abordagem de Contexto* que engloba o *Handler e Gerenciamento de Contexto* irá validar o QoE. Este será o "*trigger*" do experimento, que desencadeará a notificação ao módulo de *Encaminhamento*. O parâmetro escolhido para esta validação foi o MOS, o qual é apurado em tempo real durante a leitura do arquivo de CDR gerado pelo *Startrinity* quando executa a função de sensor (*probe*).

Como ilustrado na Figura 5-6, o *host* (2) 192.168.0.100/192.168.1.100 exerce o papel de gateway entre o UAS (3) e UAC (1), implementando um sistema de filas através do *Netfilter* (10). Em última instância, comandos *Iptables* (10) alteram a escala destas filas. O *Context Management* (6) realiza a validação de QoC (6), com base nos dados coletados (4) e armazenados na base de contexto (5) e também faz a validação do MOS (7), que neste cenário é o próprio QoE. Ocorrendo a violação de MOS, uma notificação é gerada para o módulo de *Encaminhamento* – "*queue*" (8) no diagrama da figura, que é uma FIFO de notificações. O módulo de *Adaptação de Fluxos* (9) é então acionado para determinar uma nova fila no *Netfilter (1)* cujas métricas correspondam aos requisitos necessários para acomodar as novas condições apresentadas na base de Contexto. Esta determinação é um comando *iptables* enviado por uma rotina que, neste experimento, fica em escuta no host que aloja o *Netfilter*.

Figura 5-6 - Cenário do Estudo de Caso



Nestes experimentos o critério de escolha de fila será a quantidade de banda necessária para minimamente e com alguma margem, sustentar um fluxo de chamadas dada a quantidade de chamadas apuradas (parâmetro *concurrent calls*). Este cálculo é realizado também com base no *Codec* empregado. Serão realizadas passagens experimentais com o *Codec* G.711 e G.729, nos experimentos iniciais, para apurar as diferenças de uso e sensibilidade relacionados a diferentes métodos de compressão de voz. Nos últimos experimentos será empregado o *Codec* G.723 a fim de introduzir uma variável relacionada ao dispositivo (QoD) aos cenários.

5.6 DINÂMICA DO EXPERIMENTO

A fim de facilitar a compreensão e análise dos dados obtidos, optou-se inicialmente por realizar fluxos de chamadas controlados, variando-se a quantidade de chamadas simultâneas, a cada passo correspondendo um bloco de 25 chamadas SIP.

Para estes primeiros cenários foram criadas nas rotinas pontos de parada possibilitando ao experimentador a visão precisa dos efeitos e ações realizados sobre cada variável, em rodadas sequenciais, tais como a ocorrência de notificação, se os dados estão coerentes, e a captura de tela do banco de dados a cada rodada. Um fator a ser levado em consideração é o consumo de CPU, evitando-se o consumo excessivo e as possíveis distorções que tal consumo poderia exercer sobre os resultados obtidos, i.e, perdas de pacotes derivadas do processamento elevado, e não propriamente do congestionamento do tráfego.

O tráfego de controle, de sessão SIP, foi alocado para uma fila exclusiva, de tal forma que fosse minimizado o não estabelecimento de chamadas, dado que o que se pretende medir é o efeito que o cenário exerce sobre o *payload* da ligação, mais especificamente, o tráfego de mídia que é transportado sobre RTP.

Assim sendo, a cada 25 chamadas realizadas, efetiva-se a apuração e validação do MOS referente ao intervalo de apuração. O módulo de *Encaminhamento* é informado no caso de violação do MOS definido no SLA associado ao perfil de usuário/aplicação. O módulo de *Encaminhamento*, neste caso específico, calcula a banda necessária com base em "concurrent calls", isto é, chamadas concorrentes, e no Codec empregado naquele momento, determinando a utilização de nova fila para aquele fluxo de ligações.

Em seguida, é realizada uma nova rodada de 25 chamadas e o processo é retomado, ciclicamente. A cada nova violação é gerada uma notificação, que é tratada pelo módulo de *Encaminhamento* e em seguida aplicado o respectivo comando *iptables* sobre o *Netfilter*. No cenário 4 os blocos e chamadas são ampliados e o funcionamento é cíclico.

5.7 CENÁRIOS DE EXPERIMENTAÇÃO

Nos cenários a seguir são apresentadas as telas obtidas a cada rodada, destacando-se os campos de MOS e R-factor, bem como chamadas concorrentes. O banco de dados está localizado na nuvem, em uma plataforma de hospedagem denominada MongoLab. Desta forma, todos os módulos Python podem ser executados de forma independente, em qualquer local, na mesma máquina, ou em redes distintas, sendo a única exigência o acesso à internet.

O *Controlador de Contexto* pode ser executado em uma determinada localidade, assim como o *módulo de Encaminhamento* e o *Middleware* em outras. A conexão comum é o Banco de

Dados de Contexto e as filas de notificação, que são implementadas por meio de filas FIFO no próprio MongoDB.

A visualização do banco MongoDB pode ser realizada diretamente no MongoLab, ou por meio de um software de console de banco MongoDB, neste caso, o MongoChef. Este software cliente permite visualizar o estado do banco e suas respectivas variáveis a qualquer tempo, a partir de qualquer localidade. Ao final de cada passo o "*refresh view*" permite verificar a mudança de estado das variáveis.

Nos cenários de 1 a 4 o enfoque do experimento terá portanto, como foco, a avaliação das regras de QoE. A Tabela 5-1 apresenta a aplicação das Regras de QoC comuns aos cenários de 1 a 4. Nestes primeiros quatro cenários a validação de QoC resultará em conformidade plena. O cenário 5 apresenta a medição dos tempos de resposta e os cenários de 6 a 8 apresentam situações onde ocorrem violações de QoC, conjugações de regras, e o respectivo tratamento, bem como a aplicação de regras alternativas baseadas no MOS de usuário. i.e, a introdução do fator Humano no cenário de experimentação.

Tabela 5-1 - Aplicação das Regras de QoC, comum aos cenários de 1 a 4

Parâmetro de Contexto	**Regra**	**Conformidade de QoC**
Timestamp	Validade temporal: 120 segundos	Inferior a 6 segundos
Codec	São válidos G.729 e G.711	G.729 ou G.711, admissíveis
Faixa de portas RTP	16000-16500	Variável < 16500
Protocolo de transporte	UDP	UDP
MOS	Entre 1 e 5	admissível
R-Factor	Entre 0 e 100	admissível
MOS & R-Factor	Correlacionamento	compatível
Chamadas Concorrentes	Inferior a 30	Maior valor em todos os cenários: 20 => conformidade

5.7.1 Cenário 1

A Tabela 5-2 apresenta os parâmetros aplicados aos testes e associados aos respectivos parâmetros de configuração iniciais referentes ao tamanho de filas, as chamadas realizadas simultaneamente, os resultados apurados medidos em MOS e a ação tomada, isto é, se ocorreu uma notificação para o módulo de gestão de encaminhamento ou não. Cabe notar que o consumo de banda por chamadas IP de telefonia são relativamente baixos.

Na última rodada verificou-se que com apenas 1200Mbps é possível suportar 18 chamadas simultâneas com o Codec G.711. A Tabela 5-2, portanto, apresenta as sequências referentes às rodadas executadas, o tamanho das filas, o MOS apurado e a ação tomada em termos de notificação emitida para o módulo de Encaminhamento (*Forwarding Approach*) para a manutenção e alteração de filas aplicadas a fluxos.

Tabela 5-2 - Resultados obtidos em cada passagem

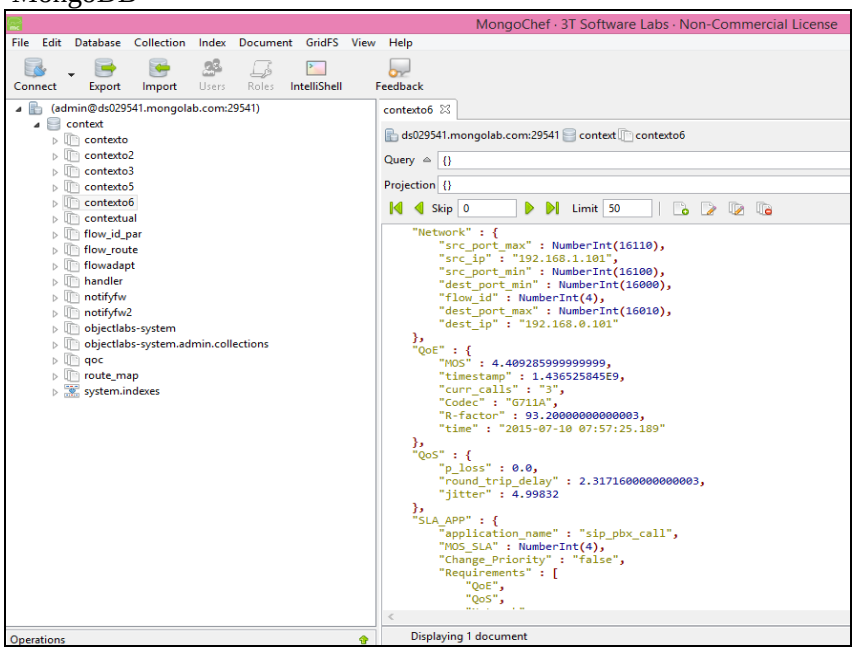
Tamanho da Fila em Kbps	**Chamadas SIP Simultâneas**	**MOS Médio Apurado**	**Ação de Notificação ao Módulo de Encaminhamento**
256	3	4,409	Não Ocorreu
256	6	2,04	Violação de QoE
700	6	4,31	Não Ocorreu
700	12	2,63	Violação de QoE
1200	12	4,409	Não Ocorreu
1200	18	3,538	Violação de QoE

Na sequência de Figuras desde a 5-8 até 5-13 são ilustrados os seguintes parâmetros:

- *Timestamp*: log temporal da coleta em segundos;
- *MOS*: medida do MOS online;
- *curr_calls*: quantidade de chamadas médias máxima apurada;
- *codec*: é o codec apurado;
- *R-Factor*: é o valor de QoE apurado como R-Factor;
- *time*: valor do tempo, data e hora;
- *ploss*: parâmeto de QoS apurado, perda de pacotes;

- *round_trip_delay*: atraso no caminho ida-e-volta;
- *jitter*: variação do atraso;
- *application_name*: nome da aplicação, e;
- *MOS_SLA*: o MOS estabelecido no sistema para a aplicação cadastrada.

Figura 5-7 - Tela do MongoChef, software cliente de visualização de banco de dados MongoDB



A seguir é apresentado o estado da base de contexto ao final de cada rodada, sequencialmente, conforme descrito na legenda relacionada e extraída ao final da cada rodada a partir da console (MongoChef) apresentada na Figura 5-7.

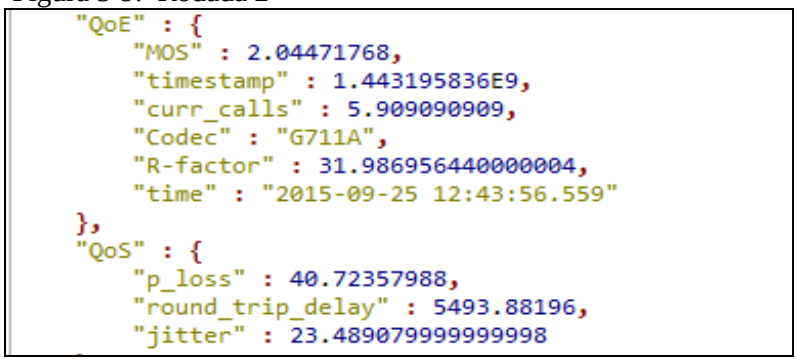
Figura 5-8 - Rodada 1


```
"QoE" : {
    "MOS" : 4.409286,
    "timestamp" : 1.443195299E9,
    "curr_calls" : 3.0,
    "Codec" : "G711A",
    "R-factor" : 93.20000000000003,
    "time" : "2015-09-25 12:34:59.854"
},
"QoS" : {
    "p_loss" : 0.0,
    "round_trip_delay" : 17.70988,
    "jitter" : 13.667560000000002
```


Nesta primeira rodada constata-se na Figura 5-8 que o MOS apurado foi de 4.40, portanto acima do valor definido no sistema para a aplicação “sip_pbx_call”, que é um valor inteiro

equivalente a MOS=4.0. Não é gerada a notificação e são realizadas no máximo 3 chamadas simultâneas. A fila utilizada, definida no Netfilter é aquela definida inicialmente, de 256Kbps, que foi capaz de comportar 3 chamadas simultâneas.

Figura 5-9. Rodada 2



```
"QoE" : {
    "MOS" : 2.04471768,
    "timestamp" : 1.443195836E9,
    "curr_calls" : 5.909090909,
    "Codec" : "G711A",
    "R-factor" : 31.986956440000004,
    "time" : "2015-09-25 12:43:56.559"
},
"QoS" : {
    "p_loss" : 40.72357988,
    "round_trip_delay" : 5493.88196,
    "jitter" : 23.489079999999998
```

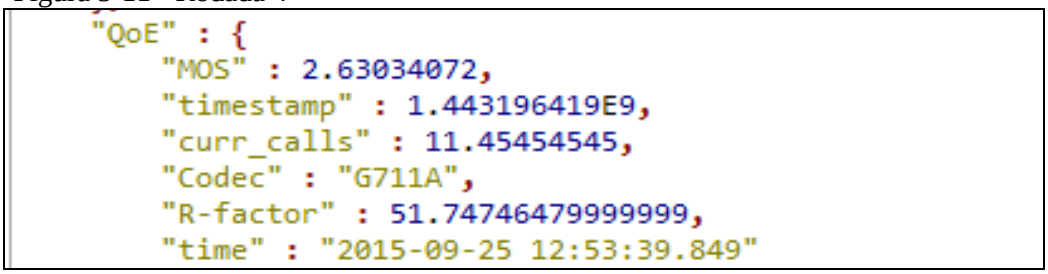

Na rodada 2, com 6 chamadas simultâneas, valor que pode ser observado no parâmetro "curr_calls" e mantendo a mesma fila de 256Kbps, o parâmetro MOS apresentou degradação e registrou um valor de 2,04. Este valor irá gerar uma violação de QoE e uma notificação ao módulo de encaminhamento para que efetivar a troca de fila.

Figura 5-10. Rodada 3

```
        "src_port_max" : NumberInt(16022),
        "src_ip" : "192.168.1.101",
        "src_port_min" : NumberInt(16000),
        "dest_port_min" : NumberInt(16100),
        "flow_id" : NumberInt(4),
        "dest_port_max" : NumberInt(16122),
        "dest_ip" : "192.168.0.102"
    },
    "QoE" : {
        "MOS" : 4.312470559999999,
        "timestamp" : 1.443196061E9,
        "curr_calls" : 5.916666667,
        "Codec" : "G711A",
        "R-factor" : 91.01012560000001,
        "time" : "2015-09-25 12:47:41.459"
    },
    "QoS" : {
        "p_loss" : 0.6330368,
        "round_trip_delay" : 172.52155999999988,
        "jitter" : 17.582119999999996
    },
    "SLA_APP" : {
        "application_name" : "sip_pbx_call",
        "MOS_SLA" : NumberInt(4),
```


Na terceira rodada, após a notificação ao módulo de encaminhamento e subsequente troca de fila, decorrente da aplicação de um comando iptables, pode ser notado que o MOS mantêm-se na faixa de 4.31. Como este valor é superior ao estabelecido pelo sistema de 4.0. A fila é mantida (700Kbps) e não é gerada uma notificação. Nesta captura de tela é apresentada uma visão expandida com outras variáveis, como por exemplo portas origem e destino, endereços ip origem e destino.

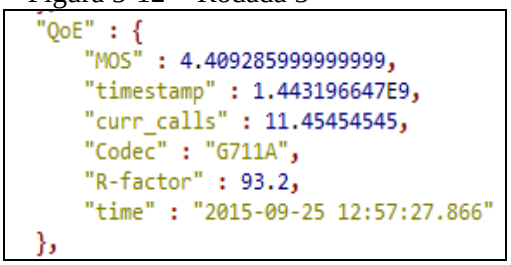
Figura 5-11 - Rodada 4



```
"QoE" : {
    "MOS" : 2.63034072,
    "timestamp" : 1.443196419E9,
    "curr_calls" : 11.45454545,
    "Codec" : "G711A",
    "R-factor" : 51.74746479999999,
    "time" : "2015-09-25 12:53:39.849"
```


Na quarta rodada, ao se tentar estabelecer 12 chamadas simultâneas obtêm-se um MOS apurado de 2.63, portanto, abaixo do mínimo estabelecido. Aqui é gerada uma notificação para o controlador de encaminhamento, solicitando uma revisão fila, já que a banda de 700Kbps não comporta mais esta quantidade de ligações.

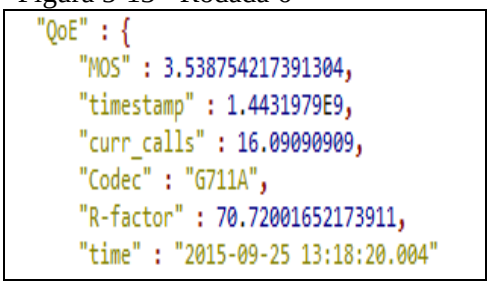
Figura 5-12 - Rodada 5



```
"QoE" : {
    "MOS" : 4.409285999999999,
    "timestamp" : 1.443196647E9,
    "curr_calls" : 11.45454545,
    "Codec" : "G711A",
    "R-factor" : 93.2,
    "time" : "2015-09-25 12:57:27.866"
},
```


Conforme pode ser verificado na Figura 5-12, as chamadas geradas sobre a fila de 1200Kbps mantêm uma medida de MOS equivalente a 4.40, e portanto, não é gerada uma notificação.

Figura 5-13 - Rodada 6



```
"QoE" : {
    "MOS" : 3.538754217391304,
    "timestamp" : 1.4431979E9,
    "curr_calls" : 16.09090909,
    "Codec" : "G711A",
    "R-factor" : 70.72001652173911,
    "time" : "2015-09-25 13:18:20.004"
```

Na rodada final, ao aplicar uma quantidade de chamadas equivalente a 18 chamadas simultâneas em uma fila de 1200Kbps, foi apurada uma degradação do MOS, cuja medição registrou o valor de 3.538, inferior ao valor limítrofe, e portanto gera notificação.

Este cenário foi executado a fim de permitir a visualização da coleta dos dados, a modificação dos dados de contexto e a geração de notificações (por meio de filas de notificação no MongoDB). O módulo de *Encaminhamento*, após o cálculo de adequação, deve escolher dentre as filas disponíveis no *Netfilter* do *gateway*, aquela que irá comportar a quantidade de chamadas simultâneas demandada. As escolhas realizadas estão representadas na coluna 1 da tabela 5-2.

Ao final da execução do cenário 1 foi observado que o sistema foi capaz de recuperar a qualidade de experiência (MOS) após notificação enviada ao controlador de encaminhamento. Neste momento o controlador escolheu a fila mais apropriada sempre que demandado, entre uma rodada e outra, com base na banda de comunicação necessária para acomodar as ligações concorrentes para o codec em uso.

5.7.2 Cenário 2

No Cenário 2 o Codec é alterado para G.729, cujo consumo efetivo de largura de banda é cerca de 31.2kbps, de acordo com a Tabela 5-3. O Codec G.729 apresenta um consumo de banda obtido por meio de compressão bem inferior ao consumo de banda do Codec G.711, motivo pelo qual é bastante empregado em situações onde a banda seja estreita, notadamente na WAN. Em redes corporativas e internas, onde a banda disponível pode ser bem mais elevada, adota-se preferencialmente o Codec G.711, que é menos suscetível a variações de parâmetros de QoS, como perda de pacotes, *jitter* e *Round Trip Delay*.

O cenário 2 comporta uma quantidade de ligações simultâneas superior ao do cenário 1, cabendo observar que a mudança de Codec é captada pelo *Controlador de Contexto* através dos dados coletados pelo *Context Handler*.

Tabela 5-3 - Dados do Experimento para Codec G.729 e avaliação das diferenças

Codec	Consumo de banda por Codec
G.711	87.2kbps
G.729	32.2kbps

Esta maior sensibilidade reflete-se na especificação G.107, que atribui ao uso do Codec G.729 um fator de *impairment*, ou seja, degradação, que é refletida em uma pontuação de partida inferior de MOS em relação ao Codec G.711 (ITU, 2015), mesmo quando a qualidade do link é a ideal em ambos os casos. Em outras palavras, o ponto de partida do fator de MOS para o Codec G.279 é menor do que o G.711.

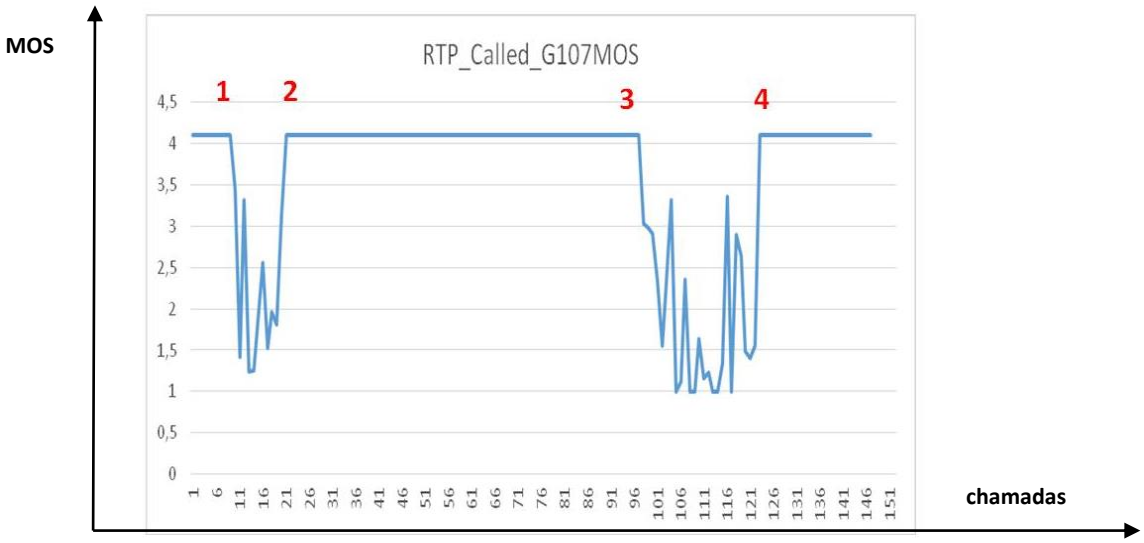
Neste experimento foram realizadas 6 rodadas sucessivas aplicando variações referentes aos parâmetros de quantidade de chamadas simultâneas e tamanho de largura de banda disponibilizada pelas filas do Netfilter, de acordo com a Tabela 5-4.

Tabela 5-4 - Quantidade de banda x Chamadas simultâneas

Banda(Kbps)	Chamadas Simultâneas	Notificações ao Módulo de Encaminhamento
128	5	Violação de QoE [Sim]
300	5	Não
300	8	Não
300	12	Violação de QOE [Sim]
700	12	Não
700	18	Não

Quando submetidas ao estrangulamento de canal, as chamadas G.729 apresentam uma rápida e abrupta degradação. Analogamente ao Cenário 1, estas alterações são captadas pelo Controlador de Contexto e resultam na notificação ao Módulo de Encaminhamento, que neste cenário requer mudanças de fila. O gráfico apresentado na Figura 5-14 foi apurado a partir dos dados coletados em função das filas utilizadas.

Figura 5-14 - Comportamento do sistema e sua recuperação a partir da mudança de fila para uma banda mais larga



O processamento de QoE ocorre somente após a validação dos parâmetros de contexto selecionados para este experimento, isto é, o QoC, elencados na Tabela 5-1, comum aos experimentos dos cenários de 1 a 4.

Em todos os cenários o "eixo x" corresponde à sequência de chamadas realizadas, e o "eixo y" corresponde aos parâmetros de QoE (MOS e R-Factor) e QoS apurados. Em relação ao primeiro cenário, o principal parâmetro alterado foi portanto o Codec G.729, que consome uma banda máxima média em torno de 30Kbps. Na Figura 5-14, pode ser visualizado que na primeira passagem, na tentativa de introduzir 5 ligações concorrentes em uma fila cuja banda é de apenas 128Kbps, logo começam a ocorrer perda de pacotes, levando a uma degradação abrupta do MOS, no instante 1. A sequência dos eventos, referenciada à Figura 5-14 é então apresentada:

1. O *Context Handler* recolhe os dados do coletados pelo agente de contexto;

2. O Context Handler sinaliza a existência de uma nova coleta de dados para o Gerenciador de Contexto;

3. O sub-módulo de apuração de QoC do Gerenciador de Contexto aplica as regras de qualidade de contexto, conforme a Tabela 1, que neste cenário está em conformidade plena;

4. O submódulo de validação QoE é chamado, onde uma violação de MOS, ou seja, um valor inferior ao limite estabelecido no perfil de usuário (MOS=4) é verificada a partir da coleta de dados realizada entre os instantes 1 e 2;

5. O Gerenciador de Contexto emite uma notificação, preenchendo a fila de notificações com o *flow_id* direcionada ao Módulo de Encaminhamento, e;

6. O Módulo de Encaminhamento, por meio de um novo cálculo de banda necessária para acomodar o fluxo de chamadas pesquisa uma nova fila, neste caso, de 300Kbps, cujo efeito de recuperação é observado no instante 2.

Deve também ser notado que no caso do MOS com Codec G.729 o MOS máximo possível é 4.1, em contraposição ao MOS igual a 4.4 com Codec G.711. Esta perda é decorrente da utilização de um Codec que introduz perda de qualidade, embora implique em ganho de banda.

Com o Codec G.729 a quantidade de ligações simultâneas possíveis em relação ao G.711 supera este por mais do que o dobro, sobre um mesmo canal, o que pode ser visualizado ao compararmos os gráficos deste cenário com os gráficos do Cenário 1, i.e, aquela que utiliza Codec G.711. O módulo de QoC, neste cenário, apura todos os dados coletados selecionados, conforme a Tabela 5-1, como válidos. Os dados de contexto apurados serão utilizados posteriormente pelo módulo de encaminhamento para calcular a banda necessária para uma nova fila.

Este cálculo é realizado através do cômputo da quantidade de ligações multiplicada pelo consumo de banda de uma chamada G729, o que resulta na ação de escolha de uma fila de 300Kbps, a primeira disponível na tabela de filas, cujo efeito é observado no instante 2 (Figura 5-14).

Desta forma, é com base na notificação emitida pelo módulo de Gestão de Contexto, e dos dados coletados na base de contexto, que tem como referência de evento a queda do MOS abaixo do mínimo especificado - uma violação de QoE, que o módulo de Encaminhamento faz os cálculos necessários para definir um novo caminho.

A execução dos fluxos de ligação segue em ordem crescente de ligações, com 8, 12 e 18 ligações simultâneas. Com 18 ligações simultâneas, o *Context Handler* apura uma nova queda de MOS, e no instante 3, e é gerada uma nova notificação do Gerenciador de Contexto para o

módulo de encaminhamento, que irá designar uma nova fila de 700Kbps para o fluxo de ligações, no instante 4.

Quadro 5-1 - Estrutura da Fila de Notificação e comandos de FIFO in e FIFO out

```
{
   "_id" : ObjectId("55a2b429e4b0b251e71411d8"),
   "notification" : "true",
   "session" : "true",
   "flowidqueue" : [
      NumberInt(4),
      NumberInt(6)
   ]
}
```

FIFO IN ➔ q.update({'_id':id},{'$push':{'flowidqueue':flow_id}},upsert=True)

q.update({'_id':id},{'$pop':{'flowidqueue':-1}}, upsert=True) ➔FIFO OUT

O Quadro 5-1 ilustra o mecanismo de enfileiramento e notificações entre o *Gerenciador de Contexto* e o *Gerenciador de Encaminhamento.* A fila "flowidqueue" registra os fluxos que deve ser tratados pelo *Gerenciador de Encaminhamento,* de acordo com a sua ordem de entrada. Neste exemplo, o fluxo de número 4 é o primeiro a ser inserido e será o primeiro a ser tratado. O comando "$pop:-1" realiza a inversão da pilha em FIFO. A sequência de eventos de sistema é então descrita:

1. O *Gerenciador de Encaminhamento,* recupera na FIFO o número de fluxo a ser tratado;
2. O Gerenciador de Encaminhamento recupera as informações de contexto referentes àquele fluxo e procura pela fila que é capaz de acomodar aquele fluxo;
3. Com base nesta informação de fila do *Netfilter* o Adaptador de Fluxo emite o comando itables, e;
4. A capacidade de transitar dados de VoIP com qualidade de MOS máxima para este Codec, ou seja, 4.1, é recuperada.

5.7.3 Cenário 3

Neste cenário introduzimos um elemento de perturbação. Os cenários 1 e 2 refletem as situações onde o canal de voz possui uma banda exclusiva e são demonstrados os casos onde

o fator de pertubação é a ocupação excessiva da fila por um excesso de ligações simultâneas. A diferença entre o cenário 1 e o cenário 2 residiu na diferença dos Codecs e na pontuação de partida do MOS apurado.

Neste cenário foi introduzida uma fonte de tráfego de perturbação, por meio da injeção de pacotes UDP concorrentes, de acordo com a sequência de testes, conforme disposto a seguir. A cadência dos testes segue a ordem e a sequência, de acordo com a seguinte nomenclatura:

<ip_origem> -----direção → <ip_destino> <chamadas simultâneas>/<chamadas por rodada> <tamanho da fila>

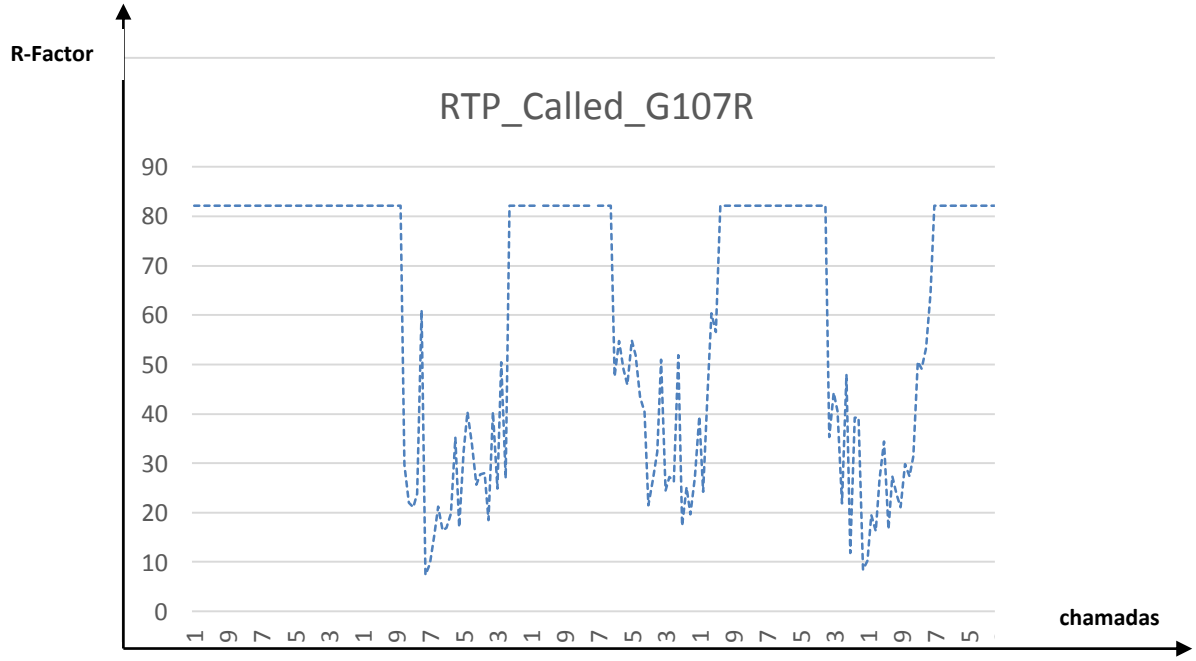
Figura 5-15 - Marcações (X) representam os períodos onde o MOS encontra-se dentro da conformidade, ao passo que marcações (X) denotam degradação

```
192.168.1.101  ------> rtp/udp 192.168.0.102  5 calls/25

queue 300kbps ( X )

192.168.1.101  ------> rtp/udp 192.168.0.102  5 calls/25

192.168.0.101  ------> udp 192.168.1.101 [12,3,50]

queue 300kbps (X)

192.168.1.101  ------> rtp/udp 192.168.0.102  5 calls/25

192.168.0.101  ------> udp 192.168.1.101 [12,6,50]

queue 300kbps (X)

192.168.0.101   -----> udp   192.168.1.101  [12,6,50]

192.168.1.101  ------> rtp/udp 192.168.0.102  5 calls/25

queue 512kbps ( X )

192.168.0.101   -----> udp   192.168.1.101  [12,10,50]

192.168.1.101  ------> rtp/udp 192.168.0.102  8 calls/25

queue 512kbps ( X )

192.168.0.101   -----> udp   192.168.1.101  [12,10,50]

192.168.1.101  ------> rtp/udp 192.168.0.102  8 calls/25

queue 700kbps ( X )

192.168.0.101   -----> udp   192.168.1.101  [12,15,50]

192.168.1.101  ------> rtp/udp 192.168.0.102  12 calls/25

queue 700Kbps (X )

192.168.0.101   -----> udp   192.168.1.101  [12,15,50]

192.168.1.101  ------> rtp/udp 192.168.0.102  12 calls/25

queue 1024Kbps  ( X )
```

A injeção de tráfego UDP – tráfego perturbador e concorrente, é calibrada de acordo com o padrão [X,Y,Z], onde X é o período em milissegundos das rajadas, Y é a quantidade de rajadas transmitidas neste período, e Z é o tamanho em bytes dos pacotes enviados.

A sequência acima descrita é realizada e os dados apurados pelo *Context Handler* embasam a tomada de decisões do Módulo de *Encaminhamento*, conforme demonstrado nos gráficos de decaimento de MOS e as retomadas da conformidade registrados a seguir. Cabe ressaltar que o controle de filas efetuado neste experimento é realizado com base somente no endereçamento IP. Não é realizada a separação de tráfegos UDP de injeção que ocorre pelas portas na faixa 13XXX e o tráfego RTP que ocorre nas portas na faixa 16XXX. O gráfico ilustrado na Figuras 5-17 está associado às tomadas realizadas com MOS no nível máximo, assinalados com a marca **(X)**, ao passo que, com MOS degradado, estes são assinalados com a marcação **(X).**

Figura 5-16 - Medições realizadas com o parâmetro R-Factor, que é estreitamente relacionado com o parâmetro MOS



A Figura 5-16 apresenta a leitura realizada com o parâmetro R-Factor, a partir do qual o MOS pode ser computado.

Figura 5-17 - Medições realizadas com o parâmetro MOS e respectivas filas

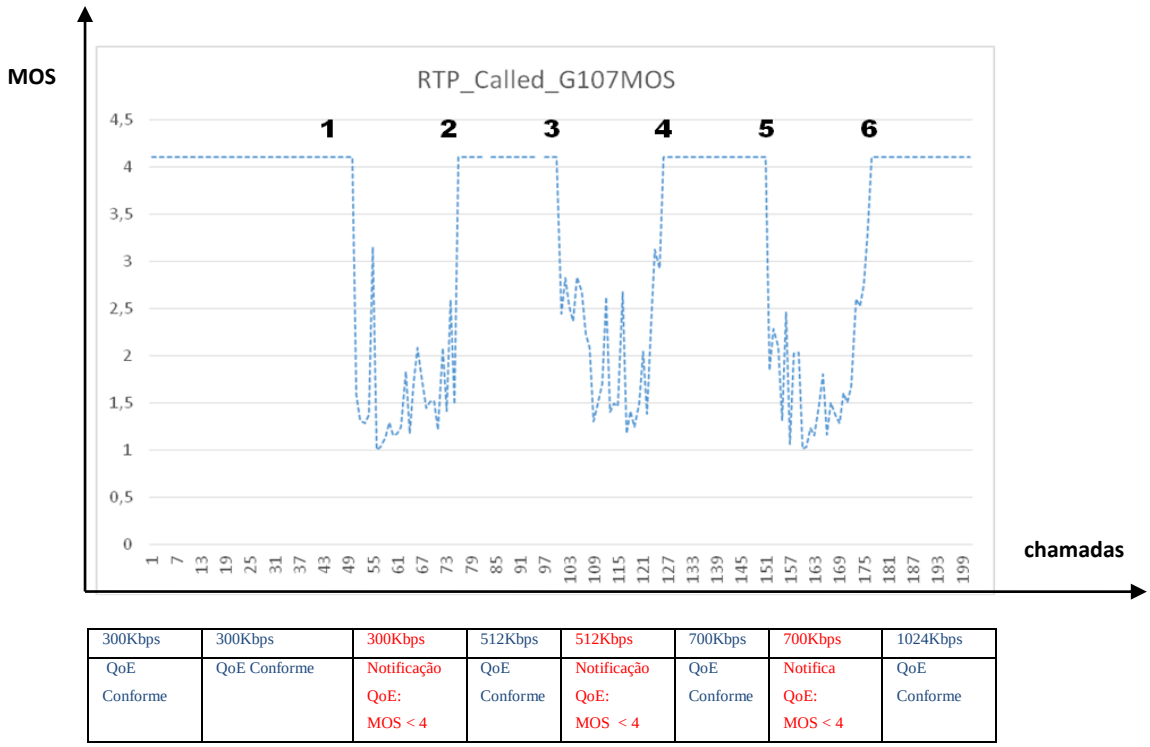


Tabela 5-5 - Aplicação das Regras de QoC

Parâmetro de Contexto	**Regra**	**Conformidade Apurada**
Timestamp	Validade temporal: 120 segundos	Inferior a 6 segundos
Codec	São válidos G.729 e G.711	G.729, admissível
Faixa de portas RTP	16000-16500	Porta máxima = 16148, admissível
Protocolo de transporte	UDP	UDP
MOS	Entre 1 e 5	admissível
R-Factor	Entre 0 e 100	admissível
MOS & R-Factor	Correlacionamento	Computado, comparado e equivalente.
Chamadas Concorrentes	Inferior a 30	Maior valor = 12; em conformidade

A validação dos parâmetros de QoC, a exemplo dos Cenários 1, 2 e 3, não produz eventos de violação de QoC, dado que para estes cenários as variáveis de contexto estão compreendidas dentro dos padrões de admissibilidade. Isto significa na prática que ao passar pela avaliação de QoC, o sub-módulo de validação de QoE será sempre acionado. A Tabela 5-5 apresenta os dados de contexto submetidos à validação de QoC para este cenário.

Pode ser notada na comparação entre os gráficos de R-factor (Figura 5-15) e MOS (Figura 5-16) a estreita vinculação entre estes parâmetros. O MOS é derivado do R-Factor, conforme verificado no Capítulo 2, e os gráficos acima demonstram esta correlação. Como nos casos anteriores, pode também ser verificado que o sistema sempre tende à recuperação quando ocorre a degradação de MOS, ou R-Factor. Na Figura 5-17 verificamos que nos instantes 1, 3 e 5 ocorre uma degradação do MOS. Com relação à Figura 5-17, ocorrem as seguintes e mesmas sequências de eventos nos intervalos (1-2), (3-4), (5-6):

1. O *Context Handler* recolhe os dados do coletados pelo agente de contexto;

2. O *Context Handler* sinaliza a existência de uma nova coleta de dados para o Gerenciador de Contexto;

3. O submódulo de apuração de QoC do Gerenciador de Contexto aplica as regras de qualidade de contexto conforme a Tabela 5-5, que neste cenário está em conformidade plena;

4. O submódulo de cálculo de QoE é chamado, onde uma violação de MOS é verificada , entre os instantes (1-2), (3-4), (5-6), ou seja, nos vales do gráfico;

5. O Gerenciador de Contexto então emite uma notificação, preenchendo a fila de notificações com o *flow_id* ao Módulo de Encaminhamento, e;

6. O Módulo de Encaminhamento, por meio de um novo cálculo de banda necessária aponta para as respectivas filas e instantes associados: 512Kbps (2), 700Kbps (4), e 1024Kbps (6).

Já entre os intervalos (0-1), (2-3), (4-5),(6):

1. O *Context Handler* adquire os dados coletados pelo agente de contexto;

2. O *Context Handler* sinaliza a existência de uma nova coleta de dados para o Gerenciador de Contexto;

3. O submódulo de apuração de QoC do Gerenciador de Contexto aplica as regras de qualidade de contexto conforme a Tabela 5-5, que neste cenário está em conformidade plena;

4. O submódulo de cálculo de QoE é chamado, onde o MOS é constatado como conforme, nos instantes (0-1), (2-3), (4-5), (6,-), ou seja, nos topos do gráfico, e;

5. O Gerenciador de Contexto então notifica e libera o Context Handler para adquirir novos dados a partir do Agente de Coleta do *Middleware*.

O gerenciador de contexto neste cenário irá computar como válidos todos os parâmetros apurados, como *timestamp*, codec utilizado, faixa de utilização das portas RTP, conforme Tabela 5-5, e na sequência do processamento é detectada a violação de MOS. A notificação enviada ao módulo de encaminhamento contém o valor do *flow_id*. Com base no *flow_id* o módulo de Encaminhamento é capaz de encontrar os dados de contexto referentes àquele fluxo e calcular o tamanho da fila necessária. Uma vez alocada uma nova fila, nos instantes 2, 4 e 6 pode ser observada a recuperação do sistema.

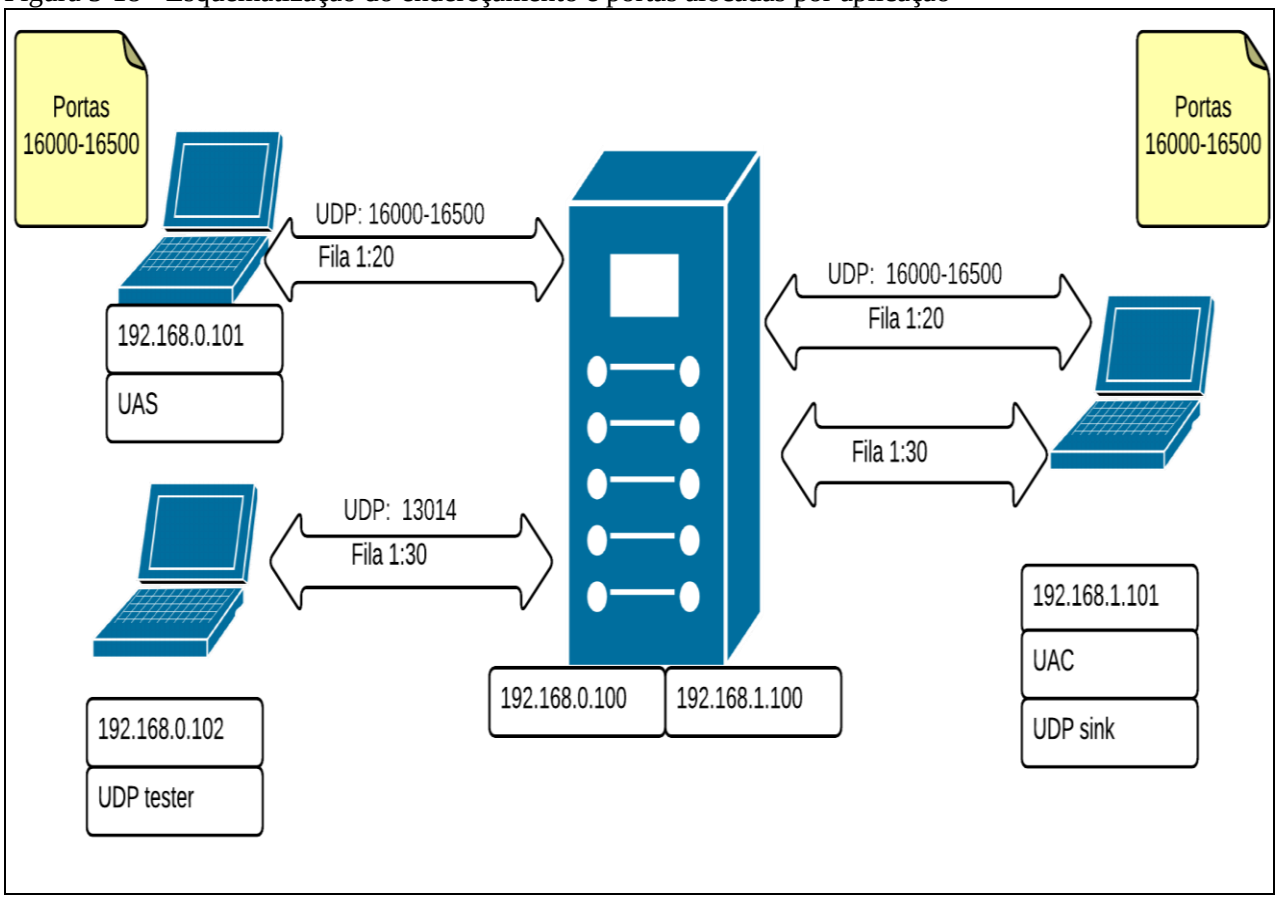
Cabe ressaltar que neste exemplo o fluxo de RTP e UDP pertubador concorrem na mesma fila. Como o cálculo de banda leva em conta apenas a quantidade de banda necessária para o fluxo RTP apenas, foi verificada a necessidade de implementar uma alteração no sistema de encaminhamento, aplicada aos próximos cenários, contemplando a separação dos fluxos por porta, e não apenas pelo endereço IP origem/IP destino. Esta alteração permite assegurar a separação por serviços, levando em conta as portas utilizadas. Na prática, neste cenário, o sistema de encaminhamento adotou a próxima fila disponível para acomodar o tráfego, sempre que ocorreu a violação de MOS, o que não é o ideal. Em decorrência desta constatação adotou-se o controle por portas na sequência dos testes.

5.7.4 Cenário 4

No cenário 4 as regras aplicadas ao gerenciamento de QoE são efetivadas através do controle de portas ao invés do endereçamento IP, como mostrado na Figura 5-18. Esta alteração permite determinar a separação das filas por serviços, considerando que cada serviço utiliza portas específicas. O *Módulo de Encaminhamento* recebe, tal como nos cenários anteriores, as notificações emitidas pelo *Controlador de Contexto*. As decisões acerca de adaptação do fluxo de chamadas são efetivadas, neste experimento, com base nos seguintes parâmetros adquiridos pelo *Controlador de Contexto*, agora considerando as portas utilizadas pela aplicação:

- Quantidade de chamadas concorrentes apuradas para um dado fluxo;
- Codec utilizado: G711, e;
- Portas utilizadas.

Figura 5-18 - Esquematização do endereçamento e portas alocadas por aplicação



Os parâmetros apurados pelo *Controlador de Contexto* são confrontados com a base de rotas disponíveis, que no caso do Módulo de *Encaminhamento* neste experimento, é implementado pelas filas do Netfilter. Assim, ao final do processamento dos dados de contexto, em cada rodada, um comando *iptables* correspondente a uma nova realocação de fila que é emitido por uma rotina que exerce o papel do *Adaptador de Fluxos* (no módulo de encaminhamento). As rodadas aplicadas neste caso seguiram a sequência descrita na Tabela 5-6:

Tabela 5-6 - Mapa de filas, classes e ações tomadas em função da quantidade de chamadas simultâneas realizadas em cada rodada.

Tamanho da fila (Kbps)	Contexto: Chamadas simultâneas	Ação: Notificação ao Módulo de Encaminhamento/ (causa)→	Adaptação: Fila no Netfilter (ação)
300	3	Não	1:20
300	6	Sim/(QoE baixo) →	1:20 (mudar de fila)
700	6	Não	1:40
700	12	Sim/(QoE baixo) →	1:40 (mudar de fila)
1400	12	Não	1:50
1400	20	Sim/(QoE baixo) →	1:50 (mudar de fila)
2400	20	Não	1:60

Este cenário compreendeu a realização de uma sequência de ligações realizadas em blocos de 50 chamadas. A Tabela 5-6 apresenta a sequência aplicada ao experimento, e as respectivas filas, quantidade de chamadas simultâneas e classes/filas aplicadas em cada rodada. Os módulos referentes ao *Handler, Gerenciador de Contexto, Gerenciador de Encaminhamento* e *Adaptador de Fluxos* foram executados em paralelo em ciclo fechado e realimentado, funcionando constantemente e agiram sobre o *Netfilter* em tempo real.

Os gráficos das Figuras 5-19 e 5-20 apresentam respectivamente os resultados apurados para o MOS e R-factor, denotando a estreita relação entre estes parâmetros. O sistema buscou a recuperação após uma apuração dos dados apreendidos pelo *Startrinity*, que são lidos pelo *Agente de Contexto no Middleware* e que são tratados pelo *Handler.*

Após um período de apuração e cálculo de médias do MOS, sempre que constatada a queda abaixo do limiar estabelecido, ocorreu uma notificação para o módulo de *Encaminhamento* que resulta em uma ação do *Adaptador de Fluxos*, que neste caso gera um comando *iptables* no sentido de atribuir uma nova fila para aquele fluxo. A esta ação sempre ocorre a recuperação subsequente do MOS/R-Factor, como pode ser observado nos gráficos das Figuras 5-19 e 5-20.

Figura 5-19 - Gráfico de apuração dos valores de MOS, decaimento e recuperação

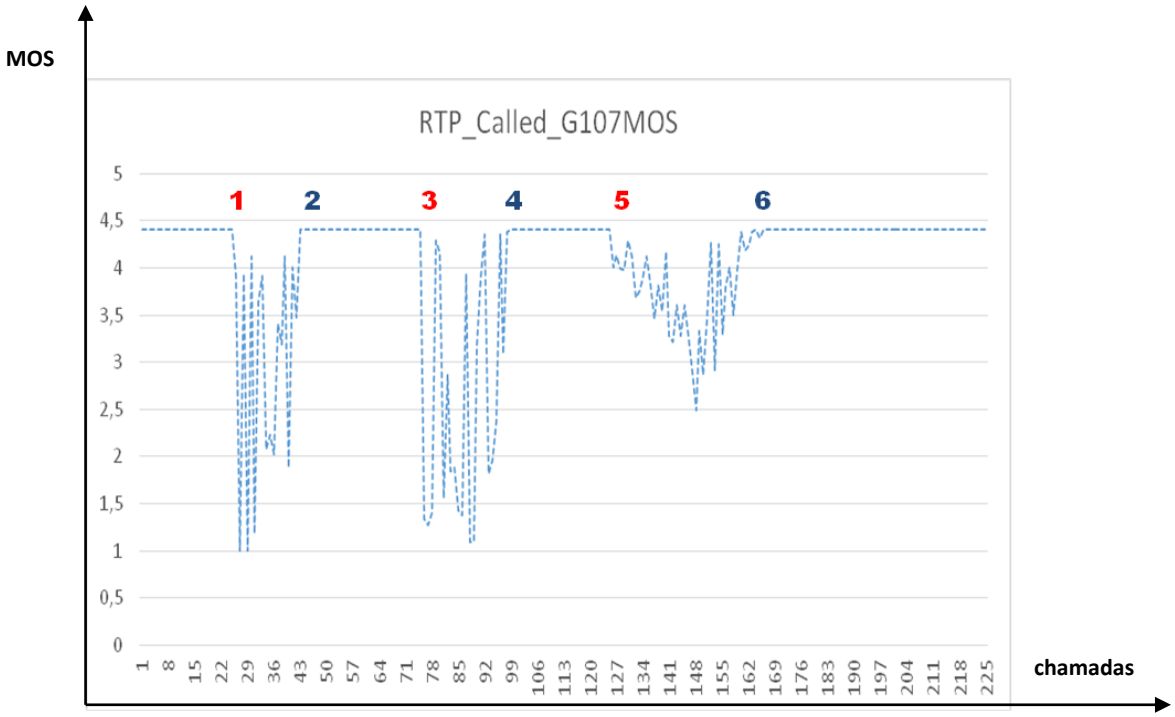
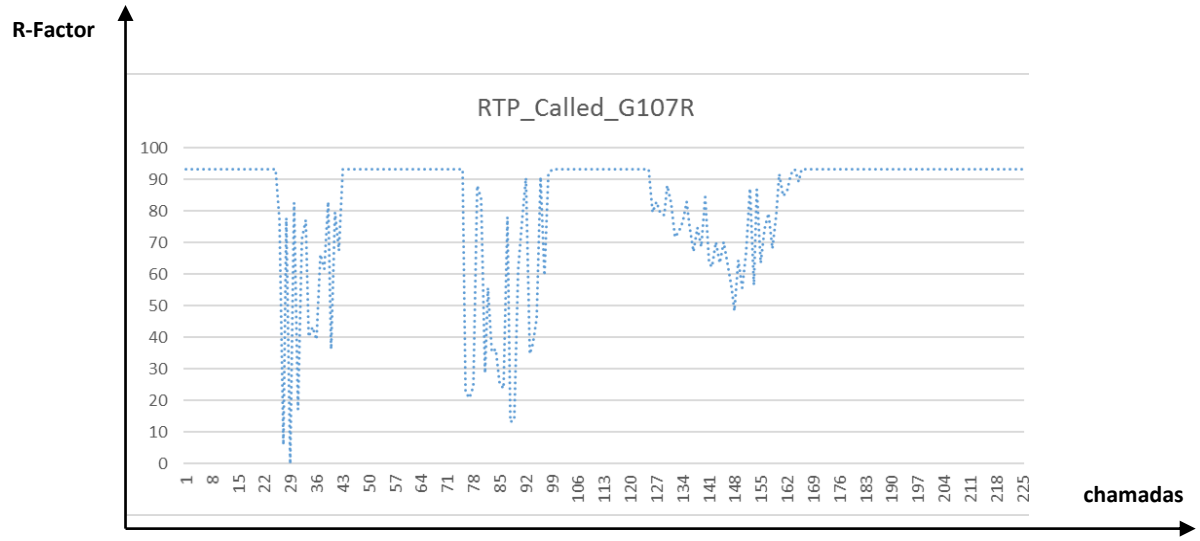


Figura 5-20 - Gráfico de apuração dos valores de R-Factor, decaimento e recuperação



Desta forma com relação ao gráfico da Figura 5-19, nos instantes (1-2), (3-4) e (5-6) ocorrem as violações de MOS. O gerenciador de contexto apura os dados de contexto, que são todos válidos neste cenário específico e a violação de QoE eventualmente apurada gera uma notificação ao *Módulo de Encaminhamento* logo em seguida ao evento de violação. O *Módulo de Encaminhamento* recebe as notificações e realiza o cálculo de filas com base no Codec utilizado e na quantidade de chamadas concorrentes apurada para aquele fluxo, conforme cálculos a seguir efetuados pelo *Gerenciador de Encaminhamento.*

Em (1-2) ao receber a notificação referente ao evento de violação de QoE o *Módulo de Encaminhamento* computa a necessidade de uma banda de 600Kbps para acomodar aquele fluxo. A próxima fila disponível para acomodar este fluxo é a 1:40, que corresponde a 700Kbps. Uma vez alocada a nova fila, ocorre em (2) a recuperação do MOS desejado. Esta sequência segue a mesma lógica nas transições que ocorrem nos seguintes eventos (3-4) e (5-6).

A sequência de eventos referente a este cenário, em função das aplicações de regras de QoE é numerada para efeito de descrição, de 1 a 8, conforme Tabela 5-7 e cuja sequência de eventos para todo o experimento é então descrita na Tabela 5.8:

Tabela 5-7 - Numeração descritiva dos eventos de sistema referentes ao cenário 4

Apura Novo Contexto: *Handler* interage com o *Middleware* a fim de adquirir novos dados coletados pelo Agente de *Middleware.*

Retém os dados de Contexto: Marca um flag na base de dados para prevenir sobrescrição enquanto ocorre o tratamento, após escrita na base de dados de contexto.

1. Notificação ao sub-módulo *Gerenciamento de Contexto* sobre Aquisição Novos Dados de Contexto: Dados foram adquiridos junto ao Agente de Coleta do *Middleware* com sucesso
2. Validação de Regras de QoC: [timestamp, RTP ports, UDP, Codec, Max Calls,MOS, R-Factor]
3. Validação de QoE
4. Liberação de Dados de Contexto pelo *Gerenciador de Contexto*: QoE está conforme
5. Notificação de violação de QoE ao *Módulo de Encaminhamento*: QoE não está conforme
6. Liberação Dados de Contexto pelo *Módulo de Encaminhamento*: Adaptação de Fluxo concluída

A Tabela 5-8 apresenta, por rodada executada, a sequência de eventos transcorrida (de acordo com a Tabela 5.7). Pode ser observado que, em determinadas rodadas são executadas a sequência de ações [1,2,3,4,5,6]. Isto ocorre quando o QoE está em conformidade com o esperado e, portanto, não ocorre notificação do *Gerenciamento de Contexto* para o *Gerenciamento de Encaminhamento*. Já na situação em que o QoE não está conforme, a sequência executada é [1,2,3,4,5,7,8]. Nos dois casos, ao final do ciclo, ocorre a notificação

ao *Handler* para que prossiga com novas aquisições de dados a partir do Agente de *Middleware*.

Tabela 5-8 - Sequência dos Eventos referentes ao cenário 4

Rodada 1	1	2	3	4	5	6	Libera aquisição	
Rodada 2	1	2	3	4	5	QoE não conforme	7	8
Rodada 3	1	2	3	4	5	6	Libera aquisição	
Rodada 4	1	2	3	4	5	QoE não conforme	7	8
Rodada 5	1	2	3	4	5	6	Libera aquisição	
Rodada 6	1	2	3	4	5	QoE não coforme	7	8
Rodada 7	1	2	3	4	5	6	Libera aquisição	

As Figuras 5-21, 5-22 e 5-23 apresentam respectivamente os fatores apurados de perda de pacotes (*Lost Packets*), atraso fim-a-fim (*RTT Round Trip Delay*) e variações de atraso (*jitter*). Estes gráficos mostram a correlação entre dados de QoS (ploss, RTT e jitter) com o MOS, isto é, a queda do MOS está sempre associada à elevação destes parâmetros de QoS. O que se pode verificar é que a apuração de um único parâmetro (MOS), traduz o estado de percepção do usuário, compreendendo uma métrica única para o sistema, dado que também leva em consideração o tipo de Codec que é utilizado.

As Figuras de 5-19 até 5-23 também ilustram a correlação que existe entre QoS e MOS. Nos casos estudados, pode ser observado que a degradação do MOS esteve sempre associada ao aumento de fatores de QoS como *jitter*, RTT e perda de pacotes. Embora estes não sejam os únicos fatores associados a MOS, a degradação de QoS é frequentemente associada diretamente, como pode ser visualizada nos gráficos, onde os “vales” das figuras de MOS correspondem aos “picos” das figuras de QoS.

Figura 5-21 - Gráfico de apuração dos valores de perda de pacotes entre rodadas

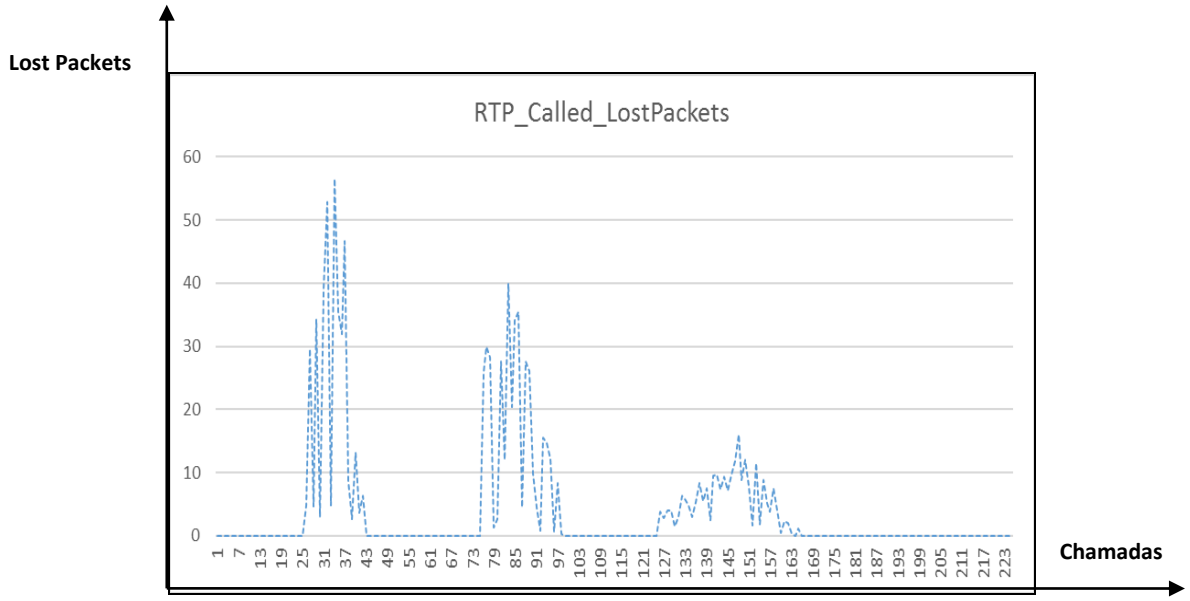


Figura 5-22 - Gráfico de apuração de RTT (atraso fim-a-fim) médios entre rodadas

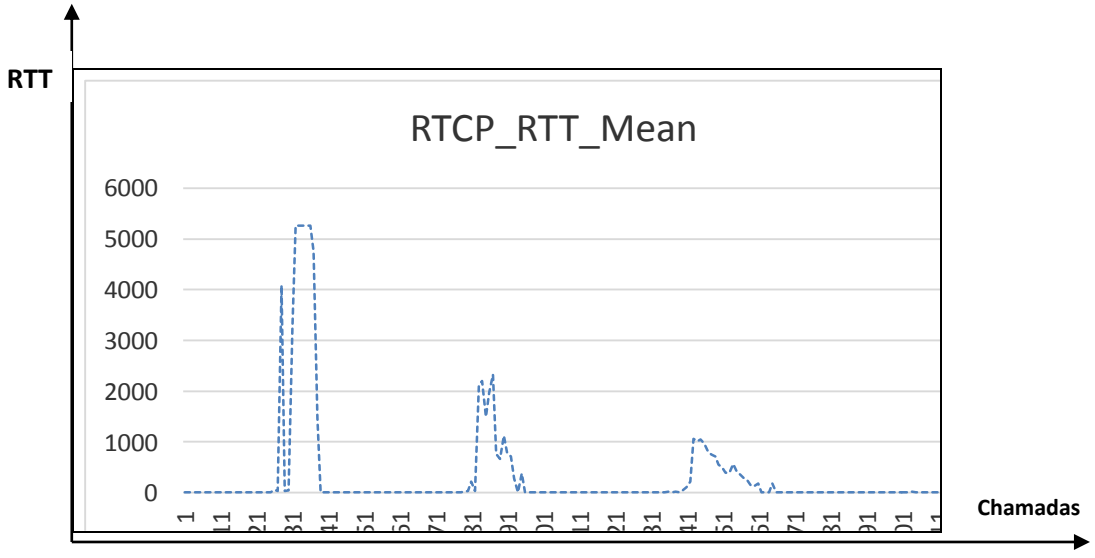
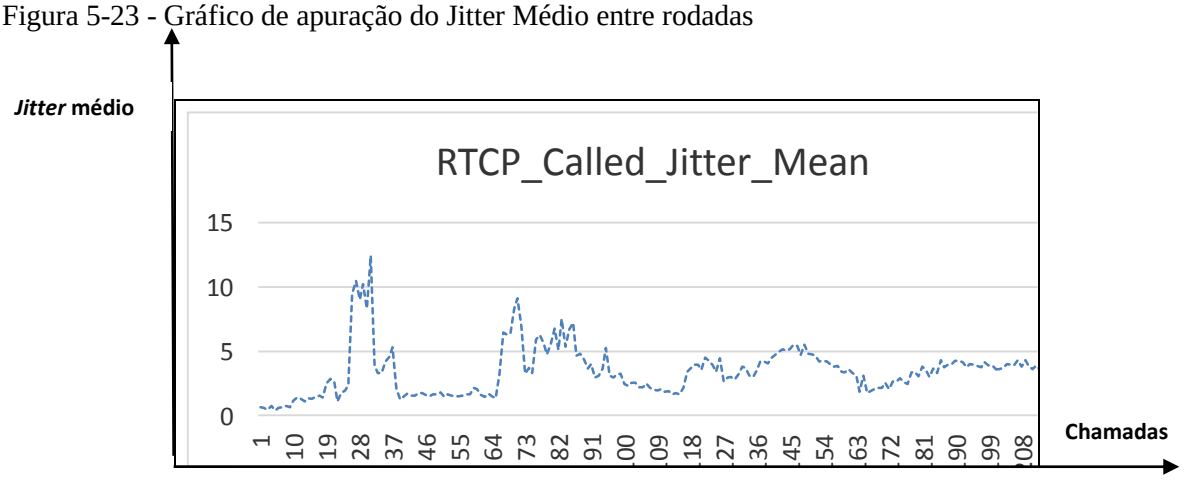


Figura 5-23 - Gráfico de apuração do Jitter Médio entre rodadas



A influência do Codec na apuração do MOS foi estudada até o momento, porém sua influência será destacada nos experimentos subsequentes. Como o parâmetro MOS é derivado do R-Factor através de uma equação que os correlaciona (R-Factor, 2015], descrita no Anexo I, pode ser visualizado que o perfil do gráfico traçado para um parâmetro está associado ao outro. Da mesma forma, podemos verificar que perda de pacotes, atrasos de ida e volta elevados e *jitter* elevado estão associados a uma degradação de MOS.

A Tabela 5-9 apresenta a aplicação de regras de QoC aplicáveis ao cenário 4. Da mesma forma que os três cenários anteriores, todos os parâmetros temporais, de QoS, QoE demais fatores de conformidade com a especificação no perfil de usuário/aplicação são obedecidos. Não havendo violação de QoC, todas as rodadas recaem no processamento de QoE, que é processado logo em seguida à validação de QoC.

Tabela 5-9 - Aplicação das Regras de QoC no Cenário 4

Parâmetro de Contexto	Regra	Conformidade Apurada
Timestamp	Validade temporal: 120 segundos	Inferior a 6 segundos: válido
Codec	São válidos G.729 e G.711	G.711 – é codec válido
Faixa de portas RTP	16000-16500	Maior valor alcançado: 16148: conforme
Protocolo de transporte	UDP	UDP: válido
MOS	Entre 1 e 5	Todas as leituras de MOS na faixa admissível
R-Factor	Entre 0 e 100	Todas as leituras de R-Factor na faixa admissível
MOS & R-Factor	Correlacionamento	Todos os Cálculos compatíveis
Chamadas Concorrentes	Inferior a 30	Maior valor em todos os cenários: 20, inferior a 30.

5.7.5 Cenário 5

No cenário 5 o objetivo é realizar a medição do tempo de reação do sistema após a aquisição dos dados de coleta. O gráfico ilustrado na Figura 5-24 apresenta o perfil de uma situação típica referente a uma degradação de MOS, levando ao evento queda de QoE.

O que se deseja medir é o tempo total decorrido entre:

- apresentação dos dados na base de contexto;
- cálculo de violação de QoE e QoC, que ocorre nos sub-módulos de verificação de QoC e QoE;
- envio da notificação ao módulo de *Encaminhamento, e;*
- ação tomada pelo *Adaptador de Fluxos* com a emissão do comando iptables, que ocorre na finalização da adaptação do roteamento.

Em outras palavras, o que se deseja saber é quanto tempo decorre entre a apresentação dos dados de coleta na base de contexto e a efetivação de uma ação por troca de fila.

Este tempo corresponde àquele que decorre no processamento interno do Controlador *de Contexto*, ou seja, quanto tempo o *Controlador de Contexto* consome para tratar o fluxo a partir da apresentação de novos dados de contexto.

Para computar o tempo decorrido foram inseridos pontos de coleta de *timestamp* associados a eventos específicos nos módulos implementados, conforme pode ser visualizado na Figura 5-24. A Figura 5-23 apresenta o cenário avaliado para realizar as medições temporais, ou seja, um cenário onde ocorreu a adaptação do fluxo.

Figura 5-24 - Gráfico de decaimento e recuperação de MOS - tomada de tempo

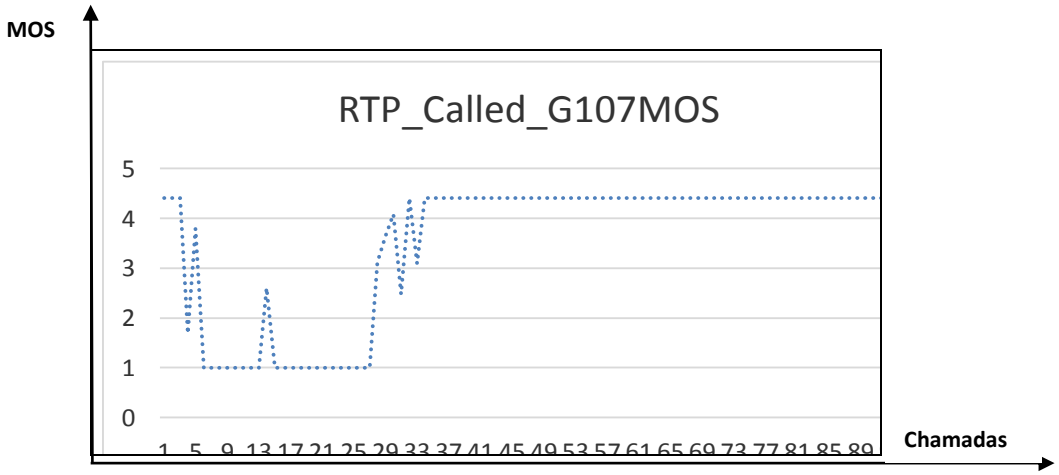
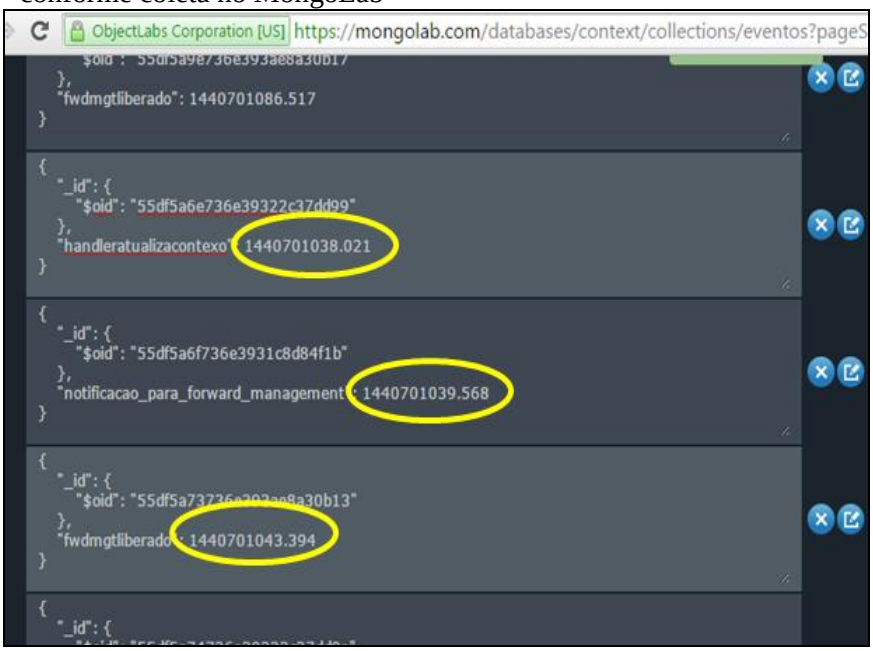


Figura 5-25 - Log de *timestamps* utilizado para cálculo de tempo de reação do sistema, conforme coleta no MongoLab



Como pode ser observado na Figura 5-25, referente aos logs de timestamp registrados na base de contexto (MongoLab), foram realizadas tomadas de tempo (*timestamp*), nos pontos descritos, de tal forma que as tomadas de tempo foram realizadas em:

- "handleratualizacontexo": timestamp = 1440701038.021, que é o ponto em que é sinalizada a atualização do documento de contexto, ou seja, que novos dados de contexto foram inseridos na base de contexto;
- "notificacao_para_forward_management": timestamp = 1440701039.568, é o período de tempo entre a sinalização de que existem novos dados de contexto para serem analisados e a atuação do Gerenciador de Contexto ao realizar os cálculos e validações de QoC e QoE, até a emissão de uma notificação para o Gerenciador de Encaminhamento, e;
- "fwdmgtliberado": timestamp =1440701043.394, é o tempo que assinala a alteração da fila no *Netfilter*.

O tempo total calculado entre a atualização pelo *Handler* e a efetivação da troca de fila neste caso foi de 5.373 segundos. Em termos gerais, foi possível concluir que o tempo entre a sinalização de novos dados de contexto e a efetiva ação do *Adaptador de Fluxos* é da grandeza equivalente a 5 segundos.

5.7.6 Cenário 6

Nos cenários anteriores, foram utilizados os Codecs G.729 e G.711. Conforme discutido no Capítulo 2, pode ser visualizado que o Codec utilizado é um parâmetro que influencia diretamente o ponto de partida inicial da apuração de MOS.

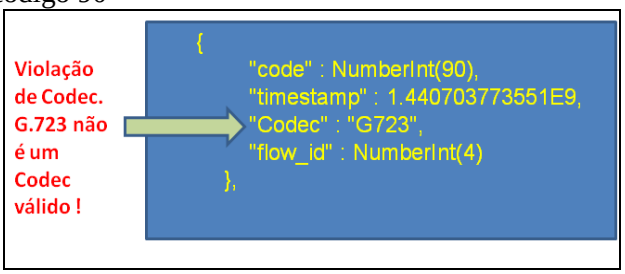
No caso do Codec G.723, este ponto de partida localiza-se ligeiramente abaixo do valor ‘4’, situando-se em ‘3.95’. Nos cenários empregados nestes experimentos, foi atribuído um parâmetro mínimo de MOS equivalente a 4. Desta forma, neste cenário hipotético, o administrador de sistema deveria excluir o Codec G.723 como Codec válido, por este não ser capaz de fornecer o valor requerido pelo sistema.

Na prática, o Codec G.723 é de fato menos empregado em comparação ao G.729 nos serviços oferecidos na WAN, dando-se preferência ao G.711 na LAN, por estar frequentemente

associado às queixas do usuário quanto à qualidade de ligações. Uma compressão mais elevada aplicada a Codecs tende a suprimir informação, ou seja, trabalha-se com aproximações que podem resultar em desconforto auditivo quando for restituída a conversação. Outro fenômeno observado é que Codecs com taxas de compressão mais elevadas tendem a ser mais suscetíveis a pequenas perdas de pacotes quando comparados a Codecs com menor compressão.

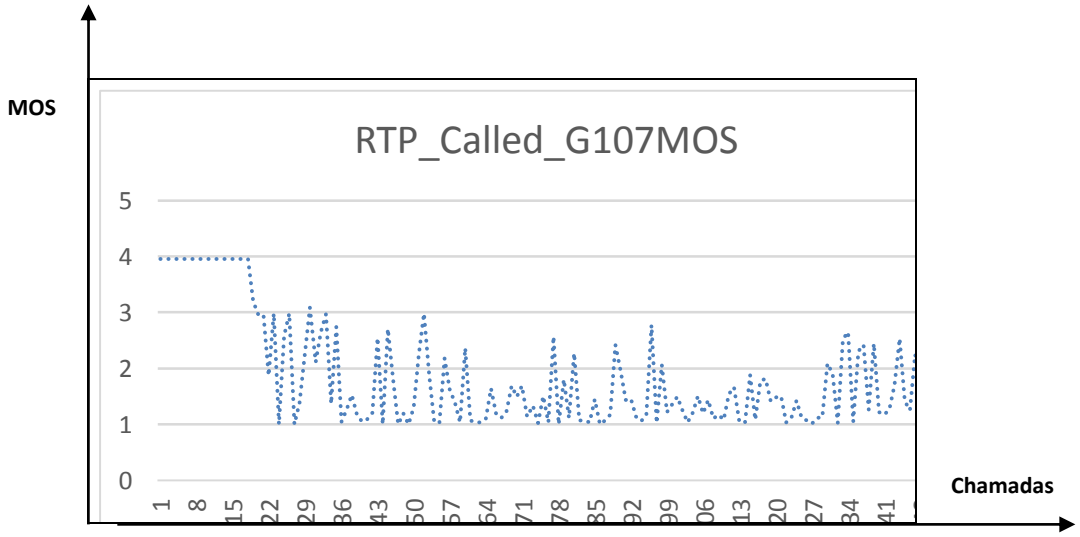
Neste cenário, o Codec G.723 representa o papel de uma violação de qualidade de dispositivo, isto é, considera-se que o G.723 é um Codec inválido, e por isto nenhuma ação deve ser *a priori* realizada quando é detectada a utilização deste Codec. A representação desta situação de violação, que também é um violação da qualidade de dispositivo (QoD) é visualizada na Figura 5-26:

Figura 5-26 - Violação de Codec, i.e, um codec não válido gera um log de código 90



O sistema implementado, portanto, é capaz de atualizar *Logs* de violação de QoC. No caso específico do Codec G.723, ao mesmo é atribuído um código (90) e a ação é o descarte. Dessa forma, quando for detectada a utilização por parte do usuário de um Codec G.723, nenhuma ação é tomada e o sistema não busca a recuperação do MOS, como pode ser observado na Figura 5-27.

Figura 5-27 - Ação de descarte, G.723 inválido, é realizado sem recuperação do MOS



Como pode ser observado na Figura 5-27, ocorreu uma violação de MOS, mas mesmo assim o sistema de gerenciamento de contexto não realiza a notificação para o *Módulo de Encaminhamento*, e somente é gerado um *log* sobre a violação de QoC. O tratamento de QoC detectou uma violação de dispositivo, dada pelo codec G.723 e foi gerado um log a este respeito. O sistema então descarta o tratamento de QoE, que não chega a ser avaliado, e o documento de contexto é dado como tratado, habilitando o *Handler* para adquirir novos dados de contexto. A Tabela 5-10 apresenta a sequência referente a evento de QoC (violação de MOS), a regra aplicada (admissão), a ação tomada (descartar), e a consequência, que é não tratamento, ao qual se segue a liberação do Handler para realizar uma nova aquisição de dados:

Tabela 5-10 - Evento, Regra, Ação e Consequência aplicados à apuração de Codec inválido

Evento de QoC: detectado Codec G.723 inválido.
Regra: somente são admitidos os Codecs G.729 e G.711 para este perfil de usuário/aplicação.
Ação: Log da notificação, com código 90, e não gera notificação (drop).
Consequência: o sistema não atua para recuperar o MOS (Figura 5-26), e libera o *Handler* para realizar novas apurações de dados.

5.7.7 Cenário 7

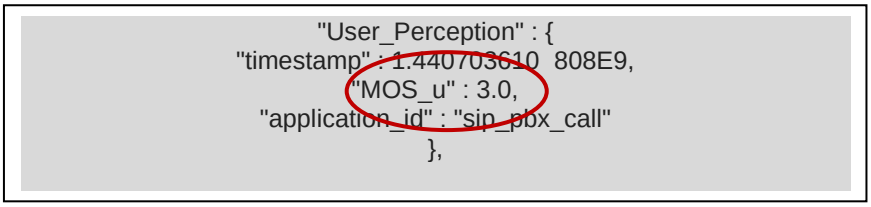
Neste caso, a utilização do Codec G.723 novamente gera um evento de violação de QoC, dado pelo uso de um Codec não cadastrado, e cujo limite teórico dado pelo MOS é 3.95, como visto no cenário anterior.

Novamente, o sistema não espera a informação deste Codec, que não está cadastrado na base de dados de usuário, o que representa uma violação de QoC. Adicionalmente, pela visão do QoD, é impraticável oferecer um MOS 4.0 conforme definido pelo administrador do sistema para um Codec G.723, denotando uma incapacidade do dispositivo de fornecer a qualidade demandada. Por outro lado, é considerado neste cenário, hipoteticamente, que o sistema admite que o administrador do mesmo possa definir a precedência da opinião do usuário (*MOS_u*) sobre as definições do sistema.

Neste caso, é aplicada uma regra conjugada e o sistema procura elevar o QoE a despeito da violação de Codec se houver uma intervenção direta do usuário. Esta intervenção é constatada quando o usuário fornece um MOS de usuário, que está presente no campo *User Perception*.

No campo da collection da Figura 5-28, para o *timestamp*: 1.440703610 808E9, ocorre a sinalização de *MOS_u* : 3.0, instante em que passa a prevalecer a opinião do usuário, e o sistema notifica o *Módulo de Encaminhamento* para procurar uma nova rota. Neste exemplo, foi determinante a percepção do usuário para que ocorresse a readaptação do fluxo.

Figura 5-28 - Campo de percepção de usuário na base de contexto com QoE associado ao valor 3 (insatisfeito)



Os dados de contexto na Figura 5-28 são representados nos campos da base de contexto do usuário **“User_Perception”:{“MOS_u”:3.0}** e, quando presentes, sinalizam ao módulo de resolução de QoC que o MOS de usuário deve ser considerado prioritariamente. Nesta elaboração de regras, que podem ser editadas, o MOS de usuário, que pode ser informado a partir do próprio aplicativo do usuário, é precedente à violação de QoC. Desta forma, é possível definir regras hierárquicas. O sistema irá trabalhar em regime de “*best effort*”

detendo o conhecimento de que o máximo MOS teórico possível neste caso é 3.95 , não sendo possível atingir a meta de MOS = 4, resultando na observação da recuperação verificada na Figura 5-29.

Figura 5-29 - Recuperação de MOS em resposta ao parâmetro MOS informado diretamente pelo usuário para um Codec G.723

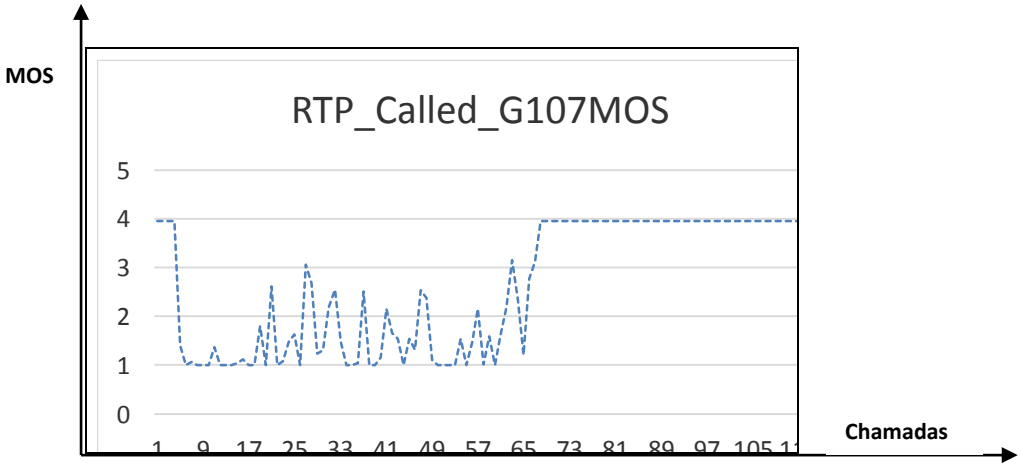
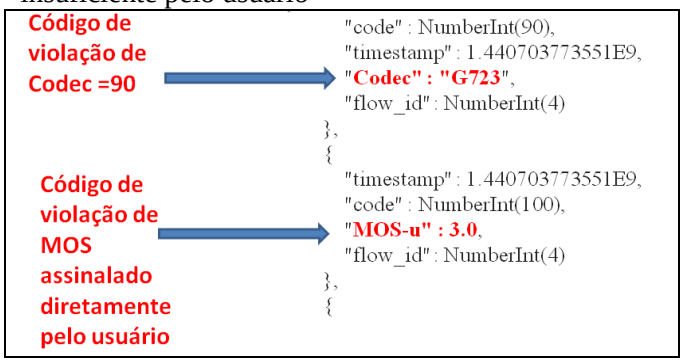


Figura 5-30 - Representação de violação de Codec e informe de MOS insuficiente pelo usuário



A Figura 5-30 apresenta a marcação código 90 no *Log* de *timestamp*, significando que ocorreu uma violação de QoD, isto é, de Codec. Da mesma forma, o usuário também assinalou uma violação de MOS de usuário, informando um MOS equivalente a 3.0, correspondente à marcação de código 100 no *Log* de *timestamp* (1.440703773551E9).

A conjugação destas duas situações leva o sistema, neste caso, a aplicar a regra de QoC que determina que o MOS informado pelo usuário pervalece sobre a violação de Codec. Em outras palavras:

- **Regra:** caso seja determinado no SLA daquele usuário/aplicação, e se for habilitada a possibilidade de o usuário notificar diretamente uma violação de MOS (por insatisfação com o serviço);
- **Evento:** o usuário informa sua percepção de QoE como baixa, e;.
- **Ação:** o sistema de gerenciamento de contexto qualifica esta violação como válida e em seguida notifica o *Módulo de Encaminhamento* para que procure uma nova fila para acomodar o presente fluxo.

Esta situação hipotética está associada às regras previamente definidas para cada cadastro de usuário/aplicação. Este cenário deve ser comparado à situação observada no cenário 6, onde a violação de Codec não é acompanhada de notificação de MOS de usuário, ou seja, o módulo de gerenciamento de contexto não gera a notificação para o *Gestor de Encaminhamento* e por este motivo não ocorre recuperação de MOS.

Desta forma, o gráfico da Figura 5-27 é resultado de uma situação (violação por Codec G.723) onde não ocorre uma recuperação, pois houve um "drop" e portanto, uma inação, ao passo que no gráfico da Figura 5-29 é resultado da situação em que uma regra é aplicada sobre: "violação por Codec G.723 acompanhada da notificação de violação de MOS pelo usuário", decorrendo a recuperação. O resultado observado foi diferente em função de mudanças no contexto, quando foi adicionada a opinião do usuário, i.e o "sentimento" do usuário acerca daquele serviço, e que pode ser contemplado pelo Gerenciador de Contexto.

5.7.8 Cenário 8

O cenário 8 compreende situações onde também ocorrem violações de *timestamps* do sistema. A situação analisada diz respeito ao *timestamp* cujo significado é a caducidade da informação. O sistema implementa a definição de desvio de tempo admissível entre a coleta de uma dada informação e o instante de sua apresentação para processamento. Se a informação fornecida estiver em conformidade com a validade temporal da mesma, o processamento segue o seu fluxo normal, ao passo que se a informação violar a validade temporal, será assinalada uma violação de QoC.

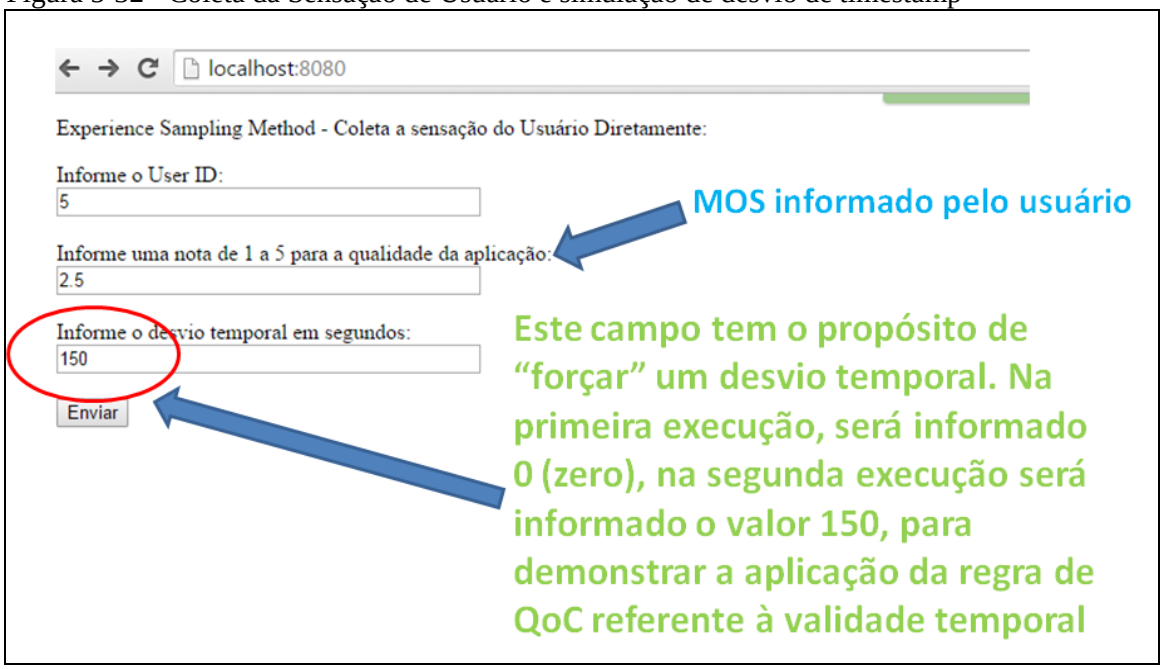
Figura 5-31 - Log de Violação de QoC, código 10

A Figura 5-31 ilustra a representação do histórico de QoC na base de dados logs de contexto, exemplificada por logs de *timestamp*, com código 10, o momento do evento e o desvio temporal registrado.

Para validar estas definições, considere a situação onde ocorre a intervenção do usuário através de uma aplicação que exemplifica a captação de opinião do usuário, que é traduzida em MOS de usuário. Nesta aplicação, ilustrada na Figura 5-30, o usuário emite a opinião acerca de um determinado serviço, associada a um *timestamp*.

Neste script, que funciona sobre o micro-arcabouço *Web* para Python bottle.py, é possível enviar comandos de GET e POST entre o aplicativo em Python e a *interface* com o navegador. No script (Figura 5-32), é disponibilizado um campo que permite arbitrar um atraso forçado no *timestamp*, que introduz uma forma de simular um atraso temporal. Este recurso será empregado a seguir com o propósito de exemplificar como um desvio no *timestamp*, no caso do MOS informado pelo usuário, pode determinar o descarte de uma nova situação de contexto por caducidade.

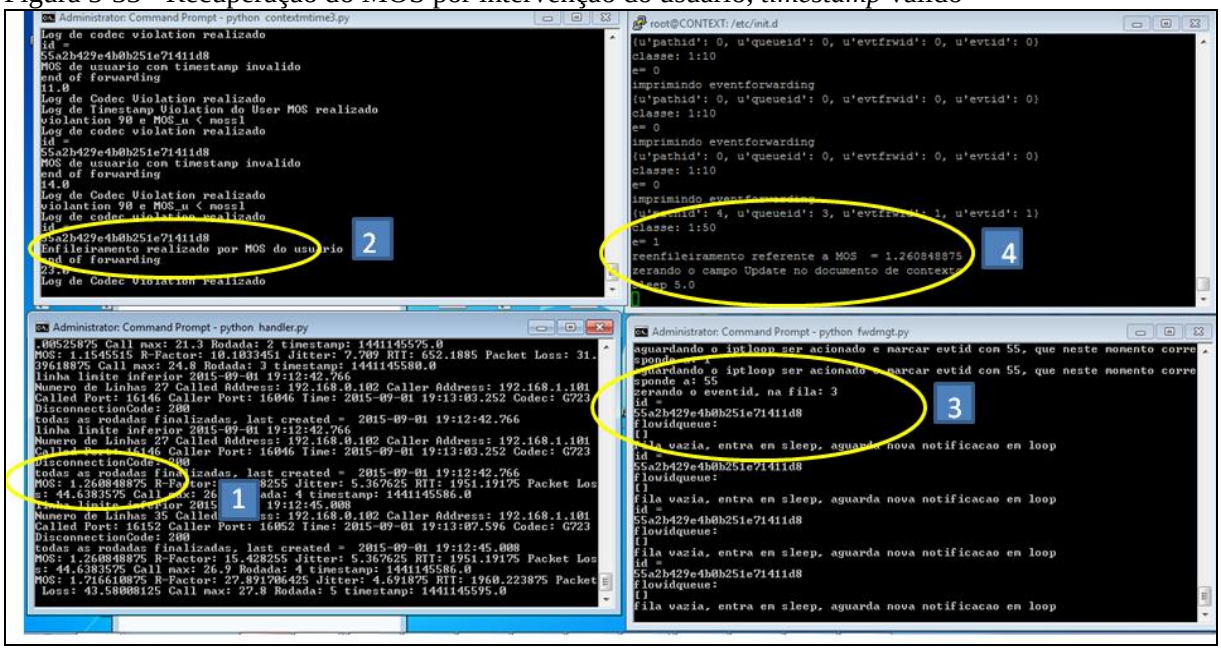
Figura 5-32 - Coleta da Sensação de Usuário e simulação de desvio de timestamp



Este script, referente à Figura 5-32 permite portanto que seja introduzido um fator de desvio temporal que associa a caducidade da informação ao MOS de usuário. Na ilustração, é mostrado como acrescentar um fator de atraso de 150 segundos na informação fornecida ao *Handler*. Serão exemplificadas as situações onde ocorre o desvio temporal zero e o desvio temporal de 150 segundos.

Na Figura 5-33 é apresentada a primeira situação, onde o usuário informa seu MOS e o *timestamp* está em conformidade com a atualidade esperada da informação.

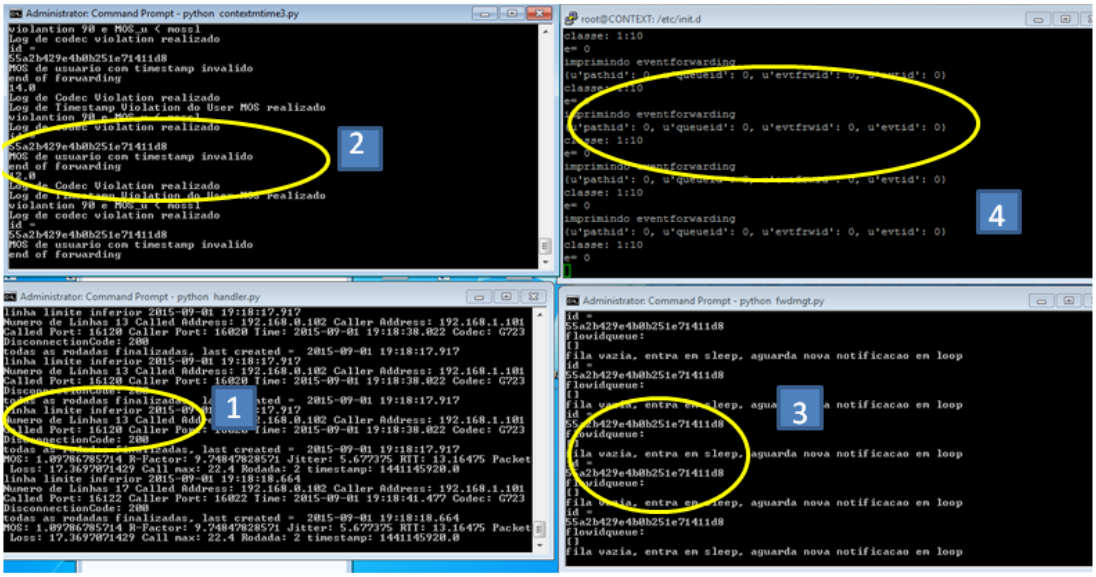
Figura 5-33 - Recuperação do MOS por intervenção do usuário, *timestamp* válido



Na Figura 5-33 pode ser observado que um MOS de 1.26 (1) é tratado pelo *Gerenciador de Contexto* por meio do enfileiramento e notificação (2) para o *Gerenciador de Encaminhamento* (3) em decorrência da sinalização de MOS de usuário. O *Flow Adapter* associa o fluxo à fila 3, classe 1:50 (4), restaurando a qualidade da aplicação, que no caso possui um MOS limitado a 3.95.

Ainda nesta figura são ilustradas as quatro aplicações em execução concomitante. Na situação analisada são disparadas ligações entre UAC e UAS, conforme os cenários precedentes, empregando o Codec G.723, que gera violação de QoC e QoD. Na primeira série da execução, a existência de uma sinalização de MOS do usuário determina o reenfileiramento, uma regra que se sobrepõe ao descarte que ocorreria normalmente por causa da violação de QoC.

Figura 5-34 - Sem recuperação do MOS por intervenção do usuário, devido a timestamp inválido



Na segunda série de chamadas cujos resultados são apresentados na Figura 5-34, também compreendendo o Codec G.723, a informação acerca do *timestamp* do MOS de usuário encontra-se fora da validade temporal admissível, que neste cenário é limitada a 120 segundos.

Esta situação foi simulada acrescentando o valor 150 no campo "informe o desvio temporal em segundos", tal como exemplificado na Figura 5-32. Como o Codec é inválido e o MOS de usuário apresenta caducidade, nenhuma ação é tomada pelo sistema no sentido de recuperar o QoE.

O MOS é coletado (1) com valores bem inferiores ao mínimo estabelecido (1.09786...), ao passo que em (2) são emitidas mensagens de notificação de violação de *timestamp* do MOS de usuário, que irão gerar o *log* correspondente, porém significando que não serão emitidas notificações para o *Módulo de Encaminhamento*, dada a caducidade da informação. Os módulos de *Gerenciamento de Encaminhamento* (3) e *Flow Adapter* (4) seguem suas execuções ciclicamente, porém sem realizar qualquer ação sobre o sistema de enfileiramento, mantendo a classe original (fila) 1:10. Consequentemente, a qualidade das ligações não é reestabelecida. Em resumo, esquematicamente, no Quadro 5-5, foram visualizadas as seguintes situações:

Tabela 5-11 - Sequência de Eventos de QoD, QoE e Temporais, que são submetidos às validações de QoC baseado nas regras estabelecidas

QoD	Temporal	QoE (fator humano)	Notificação
Codec Violado (G.723)	Informação atual	Sem MOS de Usuário	QoD violado, Não gera notificação [Não]
Codec Violado (G.723)	Informação atual	MOS violado de usuário notificado	Percepção do usuário em tempo válido: Notifica.[Sim]
Codec Violado (G.723)	Caducidade temporal	MOS de usuário notificado	Nunca gera. [Não]

5.8 DISCUSSÃO

Os experimentos foram elaborados visando uma demonstração evolutiva dos cenários, inicialmente apurando de forma estática aqueles parâmetros que se desejava medir e o respectivo efeito sobre a recuperação do sistema. Nos dois primeiros cenários, obteve-se sucesso com a execução estática dos cenários e foram dados enfoque a situações de QoE, considerando não haver violações de QoC.

No terceiro cenário foi adicionado um tráfego pertubador adicional, mas na prática, neste cenário, o sistema de encaminhamento adotou a próxima fila disponível, grande o suficiente para acomodar todo o tráfego, sempre que ocorreu a violação de MOS, o que não é o ideal, pois se tinha conhecimento apenas do volume de tráfego em RTP a partir do sensor, sem uma percepção completa do tráfego concorrente. Em decorrência desta constatação adotou-se o controle por portas na sequência dos testes.

O desafio seguinte foi implementar a integração completa entre os submódulos de gerenciamento de contexto e encaminhamento. Esta integração de forma cíclica e integrada foi obtida a partir do cenário 4. Os 3 primeiros cenários foram importantes a fim de extrair e exemplificar as situações principais e ilustrar a dinâmica da base de contexto. Uma vez alcançada a consistência desejada, a partir do cenário 4 obteve-se uma integração efetiva e cíclica e um controle de cada serviço com base em portas e endereçamento IP.

Durante esta etapa foi necessário dispender tempo adicional a fim de compreender como seria possível emitir comandos *iptables* a partir do código em *Python*, quando então ficou evidenciado que esta tarefa deveria ficar a cargo do módulo que seria executado no *gateway*.

Outra questão relevante, está relacionada à sincronia entre os quatro módulos, que são executados de forma independente. Foi necessário dispender um tempo adicional de depuração a fim de obter uma troca de mensagens efetiva através das filas implementadas em banco de dados.

Por fim, foi também necessário selecionar elementos de validação de QoD, temporais e Humanos. No caso do dispositivo, a própria releitura dos estudos realizados revelou que o Codec poderia representar o papel de dispositivo. Para incluir o parâmetro referente ao MOS do usuário foi preciso criar um script que pudesse captar a entrada de teclado, simulando a emissão de uma opinião do usuário em tempo real, enquanto os experimentos estivessem em execução e atualizar a base paralelamente.

Para apurar a caducidade da informação, a fim de testar os fatores temporais, isto é, da informação que é muito antiga e não é mais válida, chegou-se à conclusão de que o ideal seria simular a informação de timestamp a partir da entrada de teclado em tempo real.

Na medida em que ocorreu o desenvolvimento, várias possibilidades foram reveladas de forma a criar situações diferentes para validações de QoC e QoD. Desta forma, o experimento poderia ser enriquecido se houvesse um coletor no *middleware* capaz de capturar uma diversidade maior de dados. Como este coletor encontra-se fora do escopo e âmbito deste trabalho, foi escolhido o parâmetro temporal e de dispositivo Codec, a fim de simular as situações de QoC. O objetivo dos cenários 6, 7 e 8 foi portanto de experimentar diferentes situações de QoC, tanto de Codec, como temporal, como de MOS de usuário.

No Cenário 4, sobretudo, foi possível observar o funcionamento contínuo do sistema e, com relação ao testbed escolhido foi possivel verificar que o Controlador de Contexto sempre tende à recuperação do parâmetro que foi escolhido nestas avaliações, ou seja, o QoE, cumprindo sua função de motor de otimização de tráfego.

Nos experimentos finais foi validada a possibilidade de considerar a apuração de MOS de usuário, ou seja o fator humano, a partir da aplicação. Dado que é possível alimentar a base de contexto com esta informação a partir do scprit aqui desenvolvido, é possível considerar que informações subjetivas possam ser captadas de diversas formas de dentro do aplicativo do usuário, permitindo adicionar o fator humano à avaliação geral do *Controlador de Contexto*.

O arcabouço CAARF se mostrou capaz de processar diferentes tipos de informações, que resultaram em otimização de tráfego em todas as situações apresentadas, excetuando, como

esperado, aquelas em que ocorre a violação de QoC. Este mecanismo confere estabilidade ao sistema, evitando falsos positivos ou negativos que poderiam desestabilizar um sistema de Controle de Encaminhamento.

Foi portanto possível confirmar o funcionamento da abordagem proposta pelo CAARF nas seguintes situações consideradas:

- Quando existe conformidade plena quanto à validade do contexto, e ocorre uma violação de QoE (MOS), a adaptação do roteamento é sempre efetivada;
- Quando não existe conformidade de contexto, o sistema reconhece a não conformidade e gera os logs necessários, descartando as informações, e;
- O sistema pode receber intervenções do usuário, como percepção humana, e, desde que definido por regras de conjugação de QoC e QoE, a adaptação do roteamento é efetivada.

Nos casos de cenários que envolvem aplicações interativas em tempo real, notadamente nas conversações de voz, a percepção de perda de qualidade é mais crítica e imediata do que em praticamente todos os tipos de aplicação. Observa-se portanto que o controle da banda é o fator crítico, e não propriamente a disponibilidade de banda ou vazão, dado que uma rede pode propiciar uma vazão substancial e ainda assim apresentar um resultado em termos de qualidade de experiência em chamadas de voz bastante pobre, como consequência da falta de controle.

Sendo assim, é o controle da aplicação que torna-se o fator preponderante e o objetivo principal na gestão deste tipo de tráfego. Pode ser observado diretamente no cenário 1 uma sequência de recuperação de MOS após a transição de uma percepção ótima de usuário (4.4) para uma queda de percepção qualificada como muito ruim (2,04/2,6). Este evento é imediatamente processado pelo *Gestor de Contexto*, resultando na recuperação do MOS para o nível máximo (4,4).

Esta atuação no sentido do controle pode ser observada em todos os demais cenários, onde os gráficos de transição entre degradação e recuperação sempre apontam para a recuperação para o nível máximo de MOS teórico possível. Desta forma, a otimização/ganho obtidos são a recuperação da qualidade máxima possível para cada dispositivo (Codec) utilizado. Assim, o CAARF consegue sempre proporcionar a otimização de tráfego necessária com base nas

notificações qualificadas e emitidas pelo Gestor de Contexto ao Gestor de Encaminhamento, orientando-se unicamente pelo controle necessário de banda disponível e não pela simples adição de vazão.

5.9 CONCLUSÃO

Neste capítulo foram apresentados 8 cenários que representam diferentes aspectos relacionados à captura, coleta, interação e gerenciamento de contexto. Foram analisadas situações que envolvem dados relacionados à Qualidade de Experiência (QoE), foco deste trabalho, e que foram aplicados a um sistema que se revelou capaz de realizar a recuperação da qualidade esperada em um sistema de funcionamento cíclico em tempo real.

Também foi possível demonstrar situações relacionadas ao QoC, notadamente a violação do *timestamp* das informações de coleta de dados em geral, da coleta de dados de MOS de usuário e qualidade de dispositivo, dada pelas variações de Codec.

Foi igualmente demonstrado o tempo de recuperação do sistema considerando desde o instante da formação de dados de coleta válidos até a efetivação da ação sobre o *Adaptador de Fluxo,* que foi medido. Desta forma, o objetivo de implementar a troca de mensagens efetiva entre a Abordagem de Contexto e a *Abordagem de Encaminhamento* foi alcançado.

As rotinas aqui implementadas foram escritas em código Python, que se revelou de extrema agilidade para o desenvolvimento e que também facilita a compreensão. Os trechos relativos às representações de dados no banco MongoDB foram também apresentados e foi possível mostrar a perfeita integração entre o código implementado e o banco de dados ativado.

Os primeiros cenários, de 1 a 3 foram preparatórios, ao passo que no cenário 4 obteve-se uma integração plena entre os módulos (Gerenciamento de Contexto e Encaminhamento) e a adoção do controle dos fluxos por portas. Neste ponto do experimento a integração entre trabalhos distintos se efetivou. Já no cenário 5 foi realizada a medição do tempo de resposta do sistema e nos três últimos cenários realizadas as validações de situações de QoC.

Um esboço da representação do histórico associado a um determinado usuário/aplicação foi implementado, porém não foi objeto de cenário de estudos. A estrutura de logs criada abre a possibilidade, em futuras extensões a este trabalho, de também tratar os dados históricos de contexto de forma proativa.

Por fim, foram validadas as aplicações de regras sobre diferentes e variados parâmetros, como os de QoE, temporais, QoD e de percepção humana, dotando o sistema da flexibilidade necessária para a adição contínua de novas regras e inclusão de novos parâmetros, o que o torna bastante extensível e permite considerar dados sensoriais, desde que o seja possível adquirí-los.

6 CONCLUSÕES E PERSPECTIVAS FUTURAS

Ao longo deste trabalho de dissertação foi apresentada a proposta do arcabouço *Context-aware Adaptive Routing Framework* (CAARF), com enfoque no módulo de *Abordagem de Contexto* e a implementação de uma versão operacional deste arcabouço.

A seção 6.1 compreende as conclusões gerais alcançadas, o resumo do trabalho realizado, os objetivos propostos e realizados, aspectos do desenvolvimento, vantagens e desvantagens da solução e sua relevância. Finalmente é realizada uma análise do aprendizado atingido, e a seção 6.2 versa sobre as perspectivas futuras.

6.1 CONCLUSÕES

A proposta desta dissertação centrou-se na discussão acerca da utilização de dados/informações contextuais na efetivação de tomada de decisões no encaminhamento adaptativo em fluxos de dados, atuando sobre uma estrutura de interligação de redes de provedores de serviços, envolvendo a qualificação de informações de contexto.

As informações submetidas ao arcabouço são primeiramente validadas quanto à sua qualidade de informação (QoC) e, após esta qualificação, submetidas a uma validação de qualidade de experiência (QoE) e que eventualmente resulta na notificação ao módulo de *Abordagem de Encaminhamento*.

Foi proposto um arcabouço de encaminhamento adaptativo baseado em contexto, o *Context-aware Adaptive Routing Framework* (CAARF), para executar estas funcionalidades e detalhadas as estruturas dos sub-módulos *Context Handler* e *Context Model Management*, integrantes do módulo *Abordagem de Contexto* (*Context Approach*).

As descrições dos dados contextuais foram implementadas, bem classificados os eventos de Qualidade de Contexto que são registrados e processados internamente pelos sub-módulos da *Abordagem de Contexto*. Foram definidos os métodos pelos quais as informações de contexto são coletadas, formatadas e analisadas, bem como os métodos empregados nas notificações a serem enviadas ao módulo de *Abordagem de Encaminhamento*.

O modelo de base de dados contextual foi baseado em documentos e coleções que descrevem os parâmetros e atributos das entidades envolvidas. O modelo de contexto relaciona os atributos das entidades envolvidas, contemplando dados coletados acerca de:

- Qualidade de serviço (QoS);
- Qualidade de dispositivo (QoD);
- Qualidade de experiência (QoE), e;
- Temporais, locacionais e de percepção de usuários.

A abordagem de encaminhamento somente recebe notificações acerca de informações já validadas e consideradas relevantes para efeito de tomada de decisão de encaminhamento, ao passo que aquelas que são rejeitadas são objeto de log e provável descarte. Esta qualificação (QoC) possui o benefício de evitar falsos positivos e negativos e reduz significativamente o risco de geração de notificações espúrias, que poderiam resultar em instabilidade no sistema (loops).

Quanto à persistência das informações de contexto, foi definida neste trabalho a utilização de uma notação padrão fim-a-fim, baseada em formato JSON, envolvendo a preparação dos dados desde a origem no agente de *Middleware.* Estes dados são tornados disponíveis ao módulo *Handler* para preenchimento da base de contexto. A modelagem do banco de dados de informações de contexto empregou a notação JSON, em sua versão binária, BSON, permitindo o trânsito de dados sem necessidade de *parsing* entre os módulos e sub-módulos.

Atendendo ao objetivo de validar o arcabouço proposto e validar o tratamento das informações de contexto, foram implementados os módulos e sub-módulos associados, bem como o sistema de notificações. Um *testbed* foi configurado a fim de propiciar a validação do sistema proposto, demonstrar o funcionamento básico do arcabouço focado em contexto e realizar a validação de QoE.

Quanto aos resultados, foi demonstrada a capacidade de otimização do sistema em termos de recuperação da conformidade do MOS em relação às definições de SLA do usuário/aplicação sempre que ocorreu uma degradação do parâmetro MOS coletado. Foi possível medir que o sistema é capaz de obter a recuperação de MOS em um intervalo da ordem de 5 segundos entre a apresentação de novos dados de contexto e a efetivação da adaptação de fluxo.

Os experimentos realizados validaram os aspectos relacionados à aplicação de critérios de QoD, temporais e por fim de QoE de usuário, ou seja, a percepção de usuário. Para efeito de notificação ao módulo de encaminhamento, a qualificação de informações foi obtida e demonstrada através da aplicação de regras de QoC.

Os resultados obtidos demonstram que a utilização de parâmetros como o MOS, Codec e chamadas concorrentes, é suficiente para fornecer dados essenciais ao *Gestor de Encaminhamento* para que este determine uma ação de adaptação eficaz no encaminhamento dos fluxos no caso de aplicações convergentes interativas.

Demonstrou-se a possibilidade de conjugar regras envolvendo a percepção de usuário, por meio de apuração de experiência de usuário, juntamente com regras associadas à temporalidade, relacionadas à caducidade da informação. A esta qualificação foi acrescida a aplicação de regras de QoD relacionadas com a adequação/conformidade do dispositivo utilizado, ou seja, a capacidade de um dado dispositivo fornecer o desempenho requerido por uma dada aplicação.

O desafio de integrar dois trabalhos distintos, envolvendo a *Abordagem de Contexto* e *Abordagem de Encaminhamento* foi atingido. Uma implementação reduzida de Middleware e Agente Coletor no *Middleware* - além do escopo inicial do trabalho, foi implementada, tornando possível demonstrar a perfeita integração entre todos os módulos e componentes ciclicamente.

Foram atendidos, portanto, os objetivos de desenvolver e submeter à validação experimental uma solução de encaminhamento adaptativo baseado em informações de contexto para redes convergentes, por meio de implementação de um modelo de contexto que representasse dispositivos, aplicações e usuários de redes convergentes, temporais, percepção de usuários, locacionais e de dispositivos.

Através da implementação e posterior experimentação, os objetivos propostos foram alcançados, com a definição da arquitetura, prototipação e validação, definição dos módulos de contexto, definição da persistência em JSON e da persistência do repositório em NoSQL , o *Context Model Database*, definição das regras de QoC, experimentação em diversos cenários e a análise dos resultados obtidos, demonstrando o funcionamento do arcabouço proposto.

No entanto, foram encontradas dificuldades durante o desenvolvimento deste trabalho, principalmente na integração entre os módulos para que eles funcionassem coordenamente, bem como questões que suscitaram dúvidas que foram sendo resolvidas ao longo do trabalho, elencadas a seguir:

- Como definir experiência de usuário?
- Como determinar o atraso temporal?
- Como determinar QoD?
- Como determinar a captura real de dados locacionais e ambientais?

As duas primeiras perguntas foram respondidas através da implementação de uma aplicação que permitiu simular a coleta da percepção do usuário, assincronamente ao experimento, interferindo diretamente no resultado. O terceiro requisito foi atendido por meio da utilização do conceito de Codecs como dispositivos.

O último requisito requer o desenvolvimento de sensores especializados e implicaria na aplicação em uma rede móvel, o que estaria fora do escopo deste trabalho. No entanto, estes fatores estão previstos na base de contexto e o caminho para obtê-los é sugerido no protótipo de Middleware, mas a experimentação real requer um trabalho direcionado para um cenário diverso e específico.

São consideradas vantagens do arcabouço proposto sua flexibilidade para futuras extensões do modelo de dados, o que foi comprovado ao longo deste trabalho pelo uso de base de dados baseada em documentos. Este tipo de representação permite a adição, supressão e alteração de campos e descrições sem que tais modificações causem qualquer impacto ao sistema. Torna-se possível definir representações inteiramente novas e diferentes entre si, parciais ou completas, sem alterar a arquitetura de implementação.

A implementação adotada admite a interação com qualquer tipo de concepção de módulo de coleta em *Middleware*. A separação plena entre os módulos de *Abordagem de Contexto* e *Abordagem de encaminhamento* foi obtida e os sub-módulos funcionam de forma independente, comunicando-se entre si através das estruturas de fila e também por flags/semáforos de sinalização.

Uma vantagem adicional foi verificada na interoperabilidade e funcionamento autônomo entre os módulos. Assim, o *Middleware*, o *Handler* e o *Gerenciamento de Contexto* são executados como processos independentes. O *Adaptador de Fluxo* pôde ser executado em uma máquina Debian Linux separada, ao passo que o banco de dados é disponibilizado na nuvem.

Analisando o trabalho aqui apresentado, considera-se sua relevância pela efetivação do encaminhamento em redes de computadores baseado em informações de contexto, ao propor formas viáveis, efetivas, eficientes e práticas, que empregam um padrão de representação homogêneo fim-a-fim, simplificando a tarefa de coletar, classificar e qualificar estas informações desde sua coleta até a efetivação da adaptação dinâmica do tráfego.

Nos cenários considerados para efetivação da gestão do encaminhamento de tráfego foi possível verificar a capacidade do sistema de buscar a adaptação necessária ao fluxo a fim de acomodar o fluxo de dados de voz interativa, de característica mais sensível, sempre retornando ao maior nível de MOS teórico possível para cada dispositivo (Codec) empregado, uma vez validado.

Esta capacidade foi efetivada sobretudo sobre o controle do fluxo, e não necessariamente na dotação de maior vazão e relações otimizadas de atraso, mas no fornecimento de uma rota qualificada no sentido de dotar uma pequena parcela de banda – aquela estritamente necessária, porém devidamente controlada. Em outras palavras, no caso de tráfegos de voz interativas, por exemplo, uma pequena parcela de banda controlada e qualificada tem uma efetividade muito superior ao fornecimento de uma banda de grandeza superior, porém sem o mesmo controle. O objetivo buscado e alcançado nestes casos, foi o de promover mecanismos que propiciem o fornecimento de bandas qualificadas para tráfegos que por natureza são sensíveis ao controle.

E por fim, constatou-se ser possível realizar a adaptação do fluxo em um período de tempo da ordem de 5 segundos, em linha com o desejado. A adaptação assíncrona a partir da captação da opinião do usuário foi também objeto de ensaio e demonstrada no cenário 8 (conforme apresentado no capítulo 5), apontando para uma solução que pode ser estendida aos agentes de *Middleware.*

Outro quesito relevante foi a demonstração da possibilidade de integrar ferramentas de sensoriamento já existentes no mercado com o *Handler*, ampliando e potencializando substancialmente o alcance da solução e sua disponibilização como sistema para tomada de

decisões de adaptação dinâmica de fluxos. A agregação da experiência do usuário permite estender o alcance deste trabalho a fatores de qualificação disponibilizados e adquiridos dos usuários. Por fim, os formatos propostos favorecem a extensibilidade pela adição contínua e ilimitada de novos parâmetros de contexto.

O aprendizado obtido neste trabalho ampliou os horizontes de conhecimento que permitiram aprofundar e extrair novas possibilidades relacionadas à gestão e encaminhamento de tráfego sensível às condições da rede, e dos dados de contexto no sentido ampliado. Permitiu ainda aumentar o entendimento acerca da definição de contexto, um vasto campo para investigações. Também foi possível compreender a existência de diversos níveis de abstração e a possibilidade de definir níveis adicionais, como "gerência da gerência", que permitiria efetuar, por exempo, um gerenciamento orientado a QoE atuando sobre o sistema de controle de uma rede SDN.

6.2 PERSPECTIVAS FUTURAS

O arcabouço proposto propicia e favorece a elaboração de uma estrutura de *Middleware* mais complexa, dada a extensa variedade de dados de contexto que podem ser considerados para apuração. O desenvolvimento de um *Middleware* associado ao arcabouço proposto é compreendido como um trabalho extenso e detalhado, que permitiria introduzir novos dados de contexto, e por consequência ensejar a ampliação da estrutura de gestão de contexto aqui proposta. A partir desta necessidade, alguns campos de estudo aplicados ao contexto poderiam ser visualizados:

- Explorar a perspectiva do usuário quanto ao uso a partir de uma aplicação;
- Agentes de *Middleware* capazes de coletar os seguintes dados:
    - Sensoriais (dados corpóreos, como batimentos cardíacos e pressão sanguínea)
    - Ambientais (pressão atmosférica, ruído, temperatura)
    - Locacionais (dados de GPS)
    - Movimento
- Incorporação de outras variáveis que podem ser captadas de dispositivos móveis;
- Análise histórica dos dados para construir um módulo de predição de banda de utilização;
- Utilização dos dados coletados para aplicar estatísticas de uso;

- Definição de horários (histogramas) para definir uso apropriado da banda, e;
- Aplicação a redes Wi-Fi:
    - Dados de dispositivo;
    - Potência de sinal;
    - Alcance, e;
    - Tipo de rede Wi-Fi (padrão, velocidade).

Estas frentes de trabalho possuem o potencial de abrir novas áreas de estudo e a definição do encaminhamento também pela criticidade da aplicação, como em aplicações hospitalares em redes móveis de sensores, por exemplo. Para tanto, a construção aqui empregada, favorecendo a legibilidade e expressividade, propicia e viabiliza a continuidade deste projeto.

REFERÊNCIAS

ALRESHOODI, Mohammed; WOODS, John. **Survey on QoE\ QoS correlation models for multimedia services. arXiv preprint arXiv:1306.0221**. [S.l.]: [s.n], 2013.

ALVAREZ, Santiago. **QoS for IP/MPLS Networking**. [S.l.]: Cisco Press, 2006. ISBN-10: 1-58705-233-4

AWDUCHE et. al. **Overview and Principles of Internet TE**. RFC 3272. IETF. [S.l.]: [s.n], may 2002.

BLAKE, S. et al. **An Architecture for Differentiated Services.** RFC 2475, IETF. [S.l.]: [s.n], dec. 1998.

BRADEN, R. et al. **Integrated Services in the Internet architecture**: an overview, RFC 1633, IETF. [S.l.]: [s.n], 1994.

BRADEN, R., et al. **Resource Reservation Protocol (RSVP).** RFC 2205, IETF. [S.l.]: [S.n], 1997/1998.

BUCHHOLZ, T.; KUPPER, A.; SCHIFFERS, M. Quality of context: what it is and why it need. In: WORKSHOP OF THE HP OPENVIEW UNIVERSITY ASSOCIATION 2003 (HPOVUA2003). 2003. **Proc…** 2003.

CARVALHO, Leandro et al. An E-model implementation for speech quality evaluation in VoIP systems. **null. IEEE,** p. 933-938, 2005.

CHEN, Kuan-Ta et al. Quadrant of euphoria: a crowdsourcing platform for QoE assessment. **Network**, IEEE, v. 24, n. 2, p. 28-35, 2010.

CHEN, Guanling et al. **A survey of context-aware mobile computing research**. Technical Report TR2000-381. [S.l.]: Dept. of Computer Science, Dartmouth College, 2000.

COLE, Robert G.; ROSENBLUTH, Joshua H. Voice over IP performance monitoring. **ACM SIGCOMM Computer Communication Review**, v. 31, n. 2, p. 9-24, 2001.

COLLINS, D. **Carrier Grade Voice Over IP.** Second Edition. [S.l.]: McGraw Hill, 2003. ISBN 0-07-140634-4.

CORREDOR, Iván; MARTÍNEZ, José F.; FAMILIAR, Miguel S. Bringing pervasive embedded networks to the service cloud: A lightweight middleware approach. **Journal of systems architecture**, v. 57, n. 10, p. 916-933, 2011.

CRAWLEY, E. et al. **A framework for QoS-based routing in the internet**. Internet Draft (expired in September, 1997) draft-ietf-qosrframework-00. Internet Engineering Task Force (IEFT) . [S.l.]: [s.n], 1998.

DEY, Anind K. "Understanding and Using Contex.t". **Personal and Ubiquitous Computing Journal,** v.5, n. 1, 2001, pp. 4-7.

DUDASH, R., "**Software Stakeholder Management**". Disponível em: <http://www.iqps.net/iqps/pdf/pnsqcsoftwarestakeholderpaper.pdf>. Acesso em: 28 ago. 2015.

FISCHER, Simon; KAMMENHUBER, Nils; FELDMANN, Anja. REPLEX: dynamic traffic engineering based on wardrop routing policies. In: ACM CONEXT CONFERENCE. 2006. **Proceedings …** 2006.

FORTZ, B. ; THORUP, M. Optimizing OSPF/IS-IS weights in a changing world. **IEEE Journal on Selected Areas in Communications**, v.20, p.756-767, 2002.

GRACIÀ, René et al. An overview of quality of experience measurement challenges for video applications in IP networks. In:WIRED/WIRELESS INTERNET COMMUNICATIONS. Springer Berlin Heidelberg, 2010. **Proceedings…** 2010. p. 252-263.

HOßFELD, Tobias; TRAN-GIA, Phuoc; FIEDLER, Markus. Quantification of quality of experience for edge-based applications. In: MANAGING Traffic Performance in Converged Networks. Springer Berlin Heidelberg, 2007. p. 361-373.

HOßFELD, Tobias et al. Testing the IQX hypothesis for exponential interdependency between QoS and QoE of voice codecs iLBC and G. 711. In: ITC SPECIALIST SEMINAR ON QUALITY OF EXPERIENCE, 18., 2008. **Proceedings…** 2008, p. 105-114.

IOVANNA, Paola; Roberto Sabella; Marina Settembre. A Traffic Engineering System for Multilayer Networks Based on the GMPLS Paradigm. Ericsson Lab Italy. **IEEE Network,** mar.apr. 2003.

KANDULA, Srikanth et al. Walking the tightrope: Responsive yet stable traffic engineering. **ACM SIGCOMM Computer Communication Review.** ACM, p. 253-264, 2005.

KARTHIGA, S; BALAMURUGAN, M. S. Traffic Engineering System Based on Adaptative Multipath Routing. **International Journal of Emerging Technology and Advanced Engineering**, v. 3, n. 2, 2013.

KIM, Younghee; LEE, Keumsuk. A quality measurement method of context information in ubiquitous environments. In: HYBRID INFORMATION TECHNOLOGY, 2006. ICHIT'06. INTERNATIONAL CONFERENCE ON. IEEE, 2006. **Proceedings…** 2006. p. 576-581.

KORKEA-AHO, M. Context-Aware Applications Survey. In: INTERNATIONAL CONFERENCE ON HYBRID INFORMATION TECHNOLOGY (ICHIT'06), 2006. **Proceedings…** 2006.

LAGHARI, Khalil Ur Rehman; CONNELLY, Kay. Toward total quality of experience: A QoE model in a communication ecosystem. **Communications Magazine,** IEEE, v. 50, n. 4, p. 58-65, 2012.

LAUWERS et al. **Automating Unified Communications Quality of Experience using SDN, IMTC UC SDN**. [S.l.]: SDN Activity Group, 2014.

MAMMERI, Zoubir. Framework for parameter mapping to provide end-to-end QoS guarantees in IntServ/DiffServ architectures. **Computer Communications**, v. 28, n. 9, p. 1074-1092, 2005.

MASCOLO, Cecilia; MUSOLESI, Mirco. SCAR: context-aware adaptive routing in delay tolerant mobile sensor networks. In: INTERNATIONAL CONFERENCE ON WIRELESS COMMUNICATIONS AND MOBILE COMPUTING. ACM, 2006. **Proceedings…** 2006. p. 533-538.

MITRA, Karan; ZASLAVSKY, Arkady; ÅHLUND, Christer. A probabilistic context-aware approach for quality of experience measurement in pervasive systems. In: 2011 ACM SYMPOSIUM ON APPLIED COMPUTING. 2011. **Proceedings…** 2011. p. 419-424.

MU, J. et al. An adaptive routing optimization and energy-balancing algorithm in zigbee hierarchical networks. **EURASIP Journal on Wireless Communications and Networking,** v. 43, 2014.

MUAKAD, C. F. J. Gerência de encaminhamento adaptativo baseado em contexto. 2015. Dissertação (Mestrado)- Universidade Salvador – UNIFACS. Salvador, 2015.

OLIVEIRA, A. Mecanismo de Notificação baseado em Contexto para encaminhamento adaptativo. 2015. Dissertação (Mestrado)- Universidade Salvador – UNIFACS. Salvador, 2015.

MUSOLESI, M.; Hailes, S.; Mascolo, C. Adaptive Routing for Intermittently Connected Mobile Ad Hoc Networks. In: WORLD OF WIRELESS MOBILE AND MULTIMEDIA NETWORKS. SIXTH IEEE INTERNATIONAL SYMPOSIUM. 2005. **Proceedings…** 2005. p.183 – 189.

OLIVEIRA, A. et al. **Context-aware framework for adaptive routing**. Miami: Advance, 2014.

PAVLOVSKI, Christopher J. Service delivery platforms in practice [IP Multimedia Systems (IMS) Infrastructure and Services]. **Communications Magazine**, IEEE, v. 45, n. 3, p. 114-121, 2007.

PETZ, Agoston et al. An architecture for context-aware adaptation of routing in delay-tolerant networks. In: EXTREME CONFERENCE ON COMMUNICATION, 4., 2012. **Proceedings…** 2012.

PRESSMAN, R. S. **Engenharia de Software.** 6. Ed. [S.l.]: McGraw-Hill, 2006. ISBN: 8586804576

REICHL, Peter et al. The logarithmic nature of QoE and the role of the Weber-Fechner law in QoE assessment. In: COMMUNICATIONS (ICC), 2010 IEEE INTERNATIONAL CONFERENCE ON IEEE, 2010. **Proceedings…** 2010. p. 1-5.

ROBERTSON, S.; ROBERTSON, J. Mastering the Requirements Process: getting requirements right. 3. ed. [S.l.]: [s.n.]. ISBN-13: 978-0321815743.

SHAIKH, Junaid; FIEDLER, Markus; COLLANGE, Denis. Quality of Experience from user and network perspectives. **Annals of telecommunications-annales des télécommunications,** v. 65, n. 1-2, p. 47-57, 2010.

SKORIN-KAPOV, Lea; VARELA, Martín. A multi-dimensional view of QoE: the ARCU model. In: MIPRO, 2012 INTERNATIONAL CONVENTION. IEEE, 35., 2012. **Proceedings…** 2012. p. 662-666.

STANKIEWICZ, Rafal; JAJSZCZYK, Andrzej. A survey of QoE assurance in converged networks. **Computer Networks**, v. 55, n. 7, p. 1459-1473, 2011.

VENKATARAMAN, Mukundan et al. Towards a video QoE definition in converged networks. In: DIGITAL TELECOMMUNICATIONS, 2007. ICDT'07. INTERNATIONAL CONFERENCE ON. IEEE, 2., 2007. **Proceedings…** p. 16-16.

VIEIRA et al. Investigating the Specificities of Contextual Elements Management: The CEManTIKA Approach. In: INTERNATIONAL AND INTERDISCIPLINARY CONFERENCE ON MODELING AND USING CONTEXT (CONTEXT'07), LNAI 4635, 6., 2007. **Proceedings…** 2007. p. 493-506.

VIGOUREUX, Martin et al. Multilayer Traffic Engineering for GMPLS-Enabled Networks. **IEEE Communications Magazine,** jul. 2005.

WANG, Yubing. Survey of objective video quality measurements. 2006. Disponível em: Disponível em: <http://digitalcommons.wpi.edu>. Acesso em: 12 jan. 2012

WANG, N. et al. An overview of routing optimization for internet traffic engineering. **Communications Surveys & Tutorials,** IEEE, v.10, n.1, p.36-56, 2008.

WENNING, Bernd-Ludwig; TIMM-GIEL, Andreas; GÖRG, Carmelita. **A generic framework for context-aware routing and its implementation in wireless sensor networks**. [S.l.]: ITG-Fachbericht-Mobilkommunikation-Technologien und Anwendungen, 2009.

XIE, Haiyong et al. On self adaptive routing in dynamic environments-an evaluation and design using a simple, probabilistic scheme. In: NETWORK PROTOCOLS, 2004. ICNP 2004. IEEE INTERNATIONAL CONFERENCE ON. IEEE, 12., 2004. **Proceedings…** 2004. p. 12-23.

YASAR, A.; PREUVENEERS, D.; BERBERS, Y. Evaluation framework for adaptive context-aware routing in large scale mobile peer-to-peer systems. **Peer-to-Peer Networking and Applications**, v. 4, n. 1, p. 37-49, 2010.

ZHAO, S.; RAYCHAUDHURI, D. Policy-based Adaptive Routing in Mobile Ad-Hoc Wireless Networks. In: IEEE SARNOFF SYMPOSIUM, 2006. **Proceedings…** New Jersey: Princeton, mar. 27th-28th 2006.

APÊNDICE A - Publicações do Autor

OLIVEIRA, A. L. C.; MUAKAD, C. F. J.; SPINOLA, S. S.; SILVA, J. P. S; SAMPAIO, P. N. M. Context-aware framework for adaptive routing. In: INTERNATIONAL WORKSHOP ON ADVANCES IN ICP INFRASTRUCTURES AND SERVICES, 3., 2014, Miami, EUA. **Proceedings…** December, 2014.

MUAKAD, C. F. J.; SPINOLA, S. S.; OLIVEIRA, A. L. C**.;** SILVA, J. P. S.; SAMPAIO, P. N. M. Providing adaptive traffic routing based on user and network context. 2015. In: AICSSA'2015. 2015. **Short paper…** 2015..

SPINOLA, S. S.; MUAKAD, C. F. J.; SILVA, J.P.S.; OLIVEIRA, A. L. C**.;** SAMPAIO, P. N. M.; BRITO, M.P.L. Providing Adaptive Traffic Routing Based on User and Network Context. In: ESM'2015 CONFERENCE, 2015, Leicester, United Kingdom. **Proceedings…** 2015. p.410-417.

SILVA, J.P.S.; OLIVEIRA, A. L. C.; MUAKAD, C. F. J.; SPINOLA, S. S.; SILVA, SAMPAIO, P. N. M. **Context-aware Adaptive Routing Framework: A Queueing Management Approach**. 2015. Aprovado como "full paper".

APÊNDICE B - Referências Web

BOTTLE, 2015. Disponível em: http://www.admin-magazine.com/Archive/2013/13/A-web-application-with-MongoDB-and-Bottle. Acesso em 10/08/2015.

[BSON, 2015] Disponível em: http://bsonspec.org/. Acesso em 10/08/2015.

[Casado e Mckeown, 2005] The Virtual Network System. Disponível em http://yuba.stanford.edu/~casado/vns_sigcse.pdf. Acesso em 22/11/2005

[DATA MODEL MONGODB, 2015]] Disponível em: http://docs.mongodb.org/master/MongoDB-data-models-guide.pdf. Acesso em 10/08/2015.

[DS-TE, 2015] MPLS Traffic Engineering - DiffServ Aware (DS-TE).] Disponível em: http://www.cisco.com/c/en/us/td/docs/ios/mpls/configuration/guide/12_2sy/mp_12_2sy_book/mp_te_diffserv_aw.pdf. Acesso em 10/08/2015.

[GMPLS, 2015] Generalized Multiprotocol Label Switching. IEC.] Disponível em: http://www.hit.bme.hu/~jakab/edu/litr/GMPLS/GMPLS_Tutorial.pdf. Acesso em 10/08/2015.

[ITU-T Rec G.1000, 2001] Recommendation G.1000. Communications quality of service: a framework and definitions, November 2001. Disponível em: https://www.itu.int/rec/T-REC-G.1000-200111-I/en. Acesso em 10/08/2015.

[ITU-T Rec P.800.1, 2003] Recommendation P.800.1(03/03). Mean Opinion Score (MOS) terminology, Approved in 2003-03. Disponível em: https://www.itu.int/rec/T-REC-P.800.1-200303-S/en. Acesso em 16/12/2015.

[ITU-T Rec P.10, 2006] ITU-T Recommendation P.10. Vocabulary for performance and quality of service, July 2006. Disponível em: https://www.itu.int/rec/T-REC-P.10-200607-I/en. Acesso em 10/08/2015.

[ITU-T Rec P.800, 1996] ITU-T Rec. P.800. Methods For Subjective Determination of Transmission Quality. Disponível em: https://www.itu.int/rec/T-REC-P.800-199608-I/en. Acesso em 10/08/2015.

[ITU-T Rec G.107, 2003] ITU-T Rec. G107. The E-Model, A Computational Model For Use in Transmission Planning. 2003. Disponível em: https://www.itu.int/rec/T-REC-G.107-201506-I/en. Acesso em 10/08/2015.

[ITU-T Rec. H.263, 2005] Disponível em: http://www.itu.int/rec/T-REC-H.263-200501-I/en. Acesso em 10/08/2015

[ITU-T Rec H.264, 2014] Disponível em: ITU-T Rec. H.264. https://www.itu.int/rec/T-REC-H.264-201402-I/en. Acesso em 10/08/2015

[JSON,2015] Disponível em: http://www.json.org/. Acesso em 10/08/2015

[JSON-SCHEMA, 2015] Disponível em: http://json-schema.org/examples.html. Acesso em 13/08/2015

[Moltchanov, 2011] MOLTCHANOV, D. Routing Protocols for Ad-Hoc Networks. Lecture notes 2011. Disponível em: http://www.cs.tut./kurssit/TLT-2616.

[MONGODB UNIVERSITY, 2015] Disponível em: https://university.mongodb.com/. Acesso em 13/08/2015.

[MONGOLAB, 2015] Disponível em: https://mongolab.com/. Acesso em 13/08/2015.

[NETFILTER, 2015] Disponível em: http://www.netfilter.org/. Acesso em 16/11/2015.

[ONF, 2015] Open Network Foundation. Disponível em: https://www.opennetworking.org/sdn-resources/sdn-definition. Acesso em 10/08/2015

[PAESSLER, 2014] Disponível em: http://www.paessler.com/blog/2014/09/18/. Acesso em 10/08/2015.

[PRTG, 2015] Disponível em: http://sensors-of-the-week/prtg-qos-quality-of-service-round-trip-sensor-monitor-network-connection. Acesso em 10/08/2015.

[RFC 2702, 1999] Disponível em: http://www.rfc-base.org/rfc-2702.html. Acesso em 10/08/2015.

[RFC 3031, 2001] Disponível em: https://tools.ietf.org/html/rfc3031. Acesso em 10/08/2015.

[RFC 3209, 2001] Disponível em: https://tools.ietf.org/html/rfc3209. Acesso em 10/08/2015.

[RFC 3213, 2002] Disponível em: https://tools.ietf.org/html/rfc3213. Acesso em 10/08/2015.

[RFC 3261, 2002] SIP: Session Initiation Protocol. Disponível em: https://www.ietf.org/rfc/rfc3261.txt. Acesso em 10/08/2015.

[RFC 3551, 2003] Disponível em: http://tools.ietf.org/html/rfc3551. Acesso em 10/08/2015.

[RFC 3495, 2004] Disponível em: https://tools.ietf.org/html/rfc3945. Acesso em 10/08/2015.

[RFC 4204, 2005] Disponível em: https://tools.ietf.org/rfc/rfc4204.txt. Acesso em 10/08/2015.

[RFC 4594, 2006] Disponível em: https://tools.ietf.org/html/rfc4594. Acesso em 10/08/2015.

[RFC 5151, 2008] Disponível em: https://tools.ietf.org/html/rfc5151. Acesso em 10/08/2015.

[RFC 6184, 2011] Disponível em: https://tools.ietf.org/html/rfc6184. Acesso em 10/08/2015.

[RFC 6190, 2011] Disponível em: https://tools.ietf.org/html/rfc6190. Acesso em 10/08/2015.

[RFC 6416, 2011] Disponível em: https://tools.ietf.org/html/rfc6416. Acesso em 10/08/2015.

[RFC 7426, 2015] Disponível em: https://tools.ietf.org/html/rfc7426. Acesso em 10/08/2015.

[STARTRINITY, 2015] Disponível em: http://www.startrinity.com. Acesso em 10/08/2015.

[SDN-DEFINITION, 2015] Disponível em: https://www.opennetworking.org/sdn-resources/sdn-definition. Acesso em 10/08/2015.

[TAMOS, 2015] Disponível em: http://www.tamos.com/htmlhelp/voip-analysis/mosandr_factor.htm. Acesso em 10/08/2015.

[VITEL, 2012] Disponível em: http://www.vitel.com.tr/dokuman /vidyowp_svc_and_video_communication.pdf. Acesso em 10/08/2015.

[Znaty e Dauphin, 2015] ZNATY, Simon e DAUPHIN; Jean_Luis. IP Multimedia Subsystem : Principles and Architecture. Disponível em http://www.efort.com/media_pdf/IMS_ENG.pdf. Acesso em 16/11/2015.

APÊNDICE C - Códigos Python Referentes aos Sub-Módulos Handler e Context Approach

Submódulo Context Model Management
Descrição: Módulo de Validação de QoC e QoE.
Busca constantemente a atualização na base de contexto. Quando liberada faz a leitura do documento cujo flag de update =1; lê os parâmetros de contexto: timestamp, MOS, current calls,codec, MOS de usuário. Faz log de violações de QoC e, se o processamento de QoC não tiver registro de violação, verifica por violação de QoE. Se houver violação de QoE, é gerada uma notificação.

```
import pymongo
import sys
import time
import csv
import datetime
import socket
import struct, time
import time
from time import sleep
from socket import AF_INET, SOCK_DGRAM


# establish a connection to the database
#checa o timestamp, compara com o NTP, maximo de chamadas simultaneas, consistencia do R-Factor versus MOS,
localidade,

# establish a connection to the database
t1=time.time()
print 'tempo init =',t1
connection = pymongo.MongoClient('mongodb://admin:sxsxsxsx@ds029541.mongolab.com:29541/context')
db=connection['context']
contextual = db.contexto6
profile= db.userprofile
q=db.notifyfw
event=db.eventos
flowadapt = db.flowadapt
qc=db.qoc_context
qlog=db.qoc
hist=db.handler


def qocqoe_start():

        while True:
#               print 'context controller aguardando processamento de QoC e QoE'
                sleep(0.5)
                qoc_check()
#               print 'retorna de checagem de QoC e QoE'


def qoc_check():

        query={"Context_Update.Update":1}
        cursor = contextual.find(query)
```

```
#float mos = (float)(1 + 0.035 * R + R * (R - 60) * (100 - R) * 7E-6)
        for doc in cursor:

                obid=doc['_id']
                m=doc.get('QoE')
                timestamp=m['timestamp']
                tntp=getNTPTime()
                t4=qoctime(tntp,timestamp)
                print t4
                curcalls=m['curr_calls']
                codec=m['Codec']

#####################################################################
#  function qoc codec
#  verifica se o codec valida
#####################################################################

# verificando a validade do Codec, o retorno 90 significa codec invalidado
                violation=qoc_10(t4, timestamp)
                if violation != 10:
                        violation=qoc_90(codec)

                R=m['R-factor']
#encontra o MOS online
                moson=m['MOS']
#               print moson
                mr = float(1 + 0.035 * R + R * (R - 60) * (100 - R) * 7E-6)
#               print mr
                m=doc.get('Location')
                latitude=m['latitude']
                longitude=m['longitude']
                userid=doc['user_id']
                m=doc.get('User_Perception')
                MOS_u=m['MOS_u']
                timeu=m['timestamp']
                tmos=tntp-timeu
                tv=qoc_15(tmos, tntp)
                flow_id=doc['flow_id']
                query={"user.user_id":userid}
                docqoc=profile.find_one(query)
                sla=docqoc.get('SLA_APP')
                mossl=sla[0]['MOS_SLA']
#               print mossl
                maxcalls=sla[0]['max_calls']
#               print maxcalls
                audioc=sla[0]['audio_codecs']
#               print audioc
#               print len (audioc)
#               print audioc[0]
                loc=docqoc.get('Location')
#               print loc
#               print loc['latitude']
#               print loc['longitude']
                t=time.time()
#               print t
#               print (t - timestamp)
#               print 'violation ='
#               print violation
# cheque do MOS
                if ((moson < mossl)and(violation==0)):
                        print 'Detectado MOS violation: notificando o flow adapter: ',moson
                        query = {'session':'true'}
                        cursor = q.find(query)
                        for queue in cursor:
                                id=queue['_id']
#                               print 'id ='
#                               print id
```

```
                                        q.update({'_id':id},{'$push':{'flowidqueue':flow_id}},upsert=True)
                                        print 'Flow Adapt Notificado'
                                        contextual.update({'_id':obid},{"$set":{'Context_Update':{'Update':2}}},upsert=True)
                                        hist.update({'session':5},{'$push':{'trace':"fwd notificado/2"}},upsert=True)
                                        ts=time.time()
                                        event.save({'notificacao_para_forward_management':ts})
#                                       raw_input('checar se update== 2')

                        elif ((moson>=mossl)and(violation==0)):
                                print 'MOS dentro dos parametros requeridos:','MOS online', moson, 'maior que MOS
SLA',mossl
#                               print moson
#                               print 'maior que'
#                               print mossl
                                violation=-1
                                contextual.update({'_id':obid},{"$set":{'Context_Update':{'Update':0}}},upsert=True)
                                hist.update({'session':5},{'$push':{'trace':"conformidade, ler proximo
contexto/0"}},upsert=True)
#                               raw_input('checar update, tem que ser zero')

                        if ((violation==10)):
                                print 'timestamp violation'
                                contextual.update({'_id':obid},{"$set":{'Context_Update':{'Update':0}}},upsert=True)

#############################################################################
# caso especial, codec violation, mas MOS_u tem precedencia neste caso
#
#############################################################################

#                       print 'violation', violation, 'MOS_u', MOS_u, 'MOS Online', mossl

#############################################################################
# se o Codec for invalido , code 90,  mas foi sinalizado um MOS de usuario,  mesmo assim trocar a fila
#
#############################################################################

                        if ((violation==90)and(MOS_u < mossl)):
                                print 'violantion 90 e MOS_u < mossl'
                                query = {'user_id':5}
                                cur = qlog.find_one(query)
                                id=cur['_id']
                                qlog.update({'_id':id},{'$push':{'log':{'MOS-
u':MOS_u,'flow_id':flow_id,'code':100,'timestamp':t1}}})
                                print 'Log de codec violation realizado'
                                query = {'session':'true'}
                                cursor = q.find(query)
                                for queue in cursor:
                                        id=queue['_id']
                                        print 'id ='
                                        print id
# se o timestamp do MOS de usuario for valido
                                        if tv!=15:
                                                q.update({'_id':id},{'$push':{'flowidqueue':flow_id}},upsert=True)

                contextual.update({'_id':obid},{"$set":{'Context_Update':{'Update':2}}},upsert=True)
                                                hist.update({'session':5},{'$push':{'trace':"fwd notificado/2"}},upsert=True)
                                                print 'Enfileiramento realizado por MOS do usuario'
                                        elif tv==15:

                contextual.update({'_id':obid},{"$set":{'Context_Update':{'Update':0}}},upsert=True)
                                                print 'MOS de usuario com timestamp invalido'

                        print 'end of forwarding'
                return()
# verifica a validade de Codec, se o Codec infringente, retorna codigo 90 e registra no log

def qoc_90(cx):
```

```
	v=90
#	print 'function codec'
#	print cx
#	print 'verifica validade do codec'
	c=qc.find_one({'code':90})
	cod=c.get('audio-codec')
#	print cod[0]
	if cx==cod[0]:
		v=0
#	print cod[1]
	if cx==cod[1]:
		v=0
	if v==90:
		query = {'user_id':5}
		cur = qlog.find_one(query)
		id=cur['_id']
		qlog.update({'_id':id},{'$push':{'log':{'Codec':cx,'flow_id':4,'code':90,'timestamp':t1}}})
		print 'Log de Codec Violation realizado'

	return(v)

def qoc_10(tx, t1):
	v=10
	c=qc.find_one({'code':10})
	dev=c['deviation']
	if tx<dev:
		v=0
	else:
		query = {'user_id':5}
		cur = qlog.find_one(query)
		id=cur['_id']
		qlog.update({'_id':id},{'$push':{'log':{'timestamp violation':tx,'flow_id':4,'code':10,'timestamp':t1}}})
		print 'Log de Timestamp Violation realizado'
	return (v)

def qoc_15(tx, t1):
	v=15
	c=qc.find_one({'code':10})
	dev=c['deviation']
	if tx<dev:
		v=0
	else:
		query = {'user_id':5}
		cur = qlog.find_one(query)
		id=cur['_id']
		qlog.update({'_id':id},{'$push':{'log':{'timestamp violation':tx,'flow_id':4,'code':15,'timestamp':t1}}})
		print 'Log de Timestamp Violation do User MOS realizado'
	return (v)

def getNTPTime(host = "a.st1.ntp.br"):

	port = 123
	buf = 1024
	address = (host,port)
	msg = '\x1b' + 47 * '\0'

#reference time (in seconds since 1900-01-01 00:00:00)
	TIME1970 = 2208988800L # 1970-01-01 00:00:00

	# connect to server
	client = socket.socket( AF_INET, SOCK_DGRAM)
	client.sendto(msg, address)
	msg, address = client.recvfrom( buf )

	t = struct.unpack( "!12I", msg )[10]
	t -= TIME1970
#	print t
```

```
        return (t)


def qoctime(t1,t2):
        t3=t1-t2
        return (t3)


qocqoe_start()
```

Submódulo Handler
Descrição: Módulo de interação e coleta com Middleware e alimentador da base de contexto.
É a fusão do módulo de Agente de Middleware e Context Handler. O agente de Middleware constantemente lê o arquivo cvs que está sendo constantemente atualizado pelo Startrinity (o sensor). A partir de um parâmetro de amostragem definido, (8) neste caso, uma nova atualização de contexto por parte do Agente é disponibilizada. A sinalização é dada pela aquisição completada. O Handler realiza a atualização da base de contexto, contanto que a mesma tenha sido liberada pelo módulo de Gerenciamento de Contexto ou pelo Adaptador de Fluxos, para evitar sobrescrição do contexto. Todos os valores tratados e convertidos, bem como a resolução de tipagem, são inseridos na base de contexto e o flag de Update=1. Esta é a sinalização para Gerenciador de Contexto para que este possa realizar as validações de QoC e QoE.

```
import csv
import sys
import datetime
import time
import pymongo

from time import sleep
#-----------------------------------------------------------------------
def csv_reader():
        """
        Read a CSV file using csv.DictReader
        """
##########   abre o banco de dados na nuvem => Mongolab ###########
#
#
##########   leitura continua do banco sobre a maquina ligada ao csv online


        ts1=time.time()

        connection = pymongo.MongoClient('mongodb://admin:sxsxsxsx@ds029541.mongolab.com:29541/context')
        db=connection['context']
        context = db.contexto6
        event = db.eventos
# adicionado o limite de leitura, para que se possa esperar os novos preenchimentos
        limit=0

##########   abre o banco de dados gerado pelo Middleware ##########

        while True:
#               abre o banco de dados  do dia
```

```
#               arq='qoe85.csv', arquivo de teste padrao

                arq='\\Users\\sergios\\AppData\\Roaming\\StarTrinity SipTester\\CDR\\20150901.csv'
#               arq=raw_input('digite o nome do arquivo')
                sleep(0.5)
                with open(arq,'rb') as csvfile:
                        reader = csv.DictReader(csvfile, delimiter=';')
                        mj=mk=my=mi=mf=0.00
                        l=l1=l2=l3=l4=l5=0
                        count=0
                        numlines=0
                        sportmin=16500
                        sportmax=16000
                        dportmin=16500
                        dportmax=16000
                        ccmax=0
                        round=0


                        for line in reader:

                                excmr=0
                                last_created=line["Created"]
                                dcode=line["DisconnectionCode"]
                                if (dcode=="200"):
                                        j=line["RTP_Called_G107MOS"]
                                        nj=j.replace(',','.')
                                        try:
                                                njf=float(nj)
                                                mj=(mj+njf)
                                                l1+=1
                                        except ValueError:
                                                excmr+=1
                                        k=line["RTP_Called_G107R"]
                                        nk=k.replace(',','.')
                                        try:
                                                nkf=float(nk)
                                                mk=(mk+nkf)
                                                l2+=1
                                        except ValueError:
                                                excmr+=1
                                        w=line["RTP_Caller_Address"]
                                        y=line["RTCP_RTT_Mean"]
                                        ny=y.replace(',','.')
                                        try:
                                                nyf=float(ny)
                                                my=(my+nyf)
                                                l3+=1
                                        except ValueError:
                                                excmr+=1
                                        z=line["CallerId"]
                                        i=line["RTCP_Called_Jitter_Mean"]
                                        n=i.replace(',','.')
                                        try:
                                                m=float(n)
                                                mi=(m+mi)
                                                l4+=1
                                        except ValueError:
                                                excmr+=1
                                        f=line["RTP_Called_LostPackets"]
                                        nf=f.replace(',','.')
                                        try:
                                                nff=float(nf)
                                                mf=(nff+mf)
                                                l5+=1
                                        except ValueError:
                                                excmr+=2
```

```
                                        g=line["RTP_Called_Address"]
                                        h=line["RTP_Called_Port"]
                                        try:
                                                h=int(h)
                                        except ValueError:
                                                print 'rtp called port read error'
                                        if (h < dportmin):
                                                dportmin=h
                                        if h>dportmax:
                                                dportmax=h
                                        r=line["RTP_Caller_Port"]
                                        try:
                                                r=int(r)
                                        except ValueError:
                                                print 'rtp caller port error'
                                        if r<sportmin:
                                                sportmin=r
                                        if r>sportmax:
                                                sportmax=r
                                        p=line["RTP_Caller_Address"]
                                        s=line["Destroyed"]
                                        t=line["RTP_Codec"]
                                        cc=line["Current_Calls_Count"]
                                        ccf=cc.replace(',','.')
                                        cc=float(ccf)
                                        if cc>ccmax:
                                                ccmax=cc
#                                               print 'ccmax',ccmax
                                        count+=1
#                                       if count == 25:
#                                       if count == 12:
                                        if count == 8:
#                                               print 'ccmax=',ccmax
                                                timestamp = time.mktime(datetime.datetime.strptime(s, "%Y-%m-
%d %H:%M:%S.%f").timetuple())
                                                varmos=mj/l1
                                                varrfact=mk/l2
                                                varjitter=mi/l4
                                                varrtt=my/l3
                                                varploss=mf/l5
                                                flow_id=04
                                                round+=1
# para evitar repetir as linhas e ultrapassar o limite corrente, last_created tem que ser maior que ultima linha processada

                                                if (last_created>limit):
                                                        print 'MOS:',varmos,'R-Factor:',varrfact,'Jitter:',
varjitter,'RTT:',varrtt,'Packet Loss:',varploss,'Call max:',ccmax,'Rodada:',round,'timestamp:',timestamp
                                                        cntxt=context.find_one({"session":5})
                                                        up=cntxt.get('Context_Update')
                                                        update=up['Update']

# enquanto esta tomada de contexto nao for tratada

                                                        while ((update==1) or (update==2)):
#                                                               print 'upadate=cntxt[update]',update
                                                                cntxt=context.find_one({"session":5})
                                                                up=cntxt.get('Context_Update')
                                                                update=up['Update']


# quando o tratamento do contexto anterior for tratado, o update vira para zero e novo update realizado



                context.update({"session":5},{"$set":{'QoE':{'MOS':varmos,'R-
factor':varrfact,'Codec':t,'time':s,'timestamp':timestamp,"curr_calls":ccmax}}}, upsert=True)
```

```
            context.update({"session":5},{"$set":{'QoS':{'p_loss':varploss,'round_trip_delay':varrtt,'jitter':varjitter}}},
upsert=True)

            context.update({"session":5},{"$set":{'Network':{'src_ip':p,'dest_ip':g,'dest_port_min':dportmin,'dest_port_max':dp
ortmax,'src_port_max':sportmax,'src_port_min':sportmin,'flow_id':flow_id}}}, upsert=True)

            context.update({"session":5},{"$set":{'sip_session':{'Codec':t,'protocol':'udp','rtp':'true'}}},upsert=True)

            context.update({"session":5},{"$set":{'Context_Update':{'Update':1}}},upsert=True)
                                                                ts=time.time()
                                                                event.save({'handleratualizacontexo':ts})

#                                                       print 'limite superior',last_created, limit
                                                        count=0
                                                        mj=mk=my=mi=mf=0.00
                                                        l=l1=l2=l3=l4=l5=0
                                                        ts2=time.time()
#                                                       print 'tempo decorrido',ts2-ts1
#                                                       sleep(0.5)
                                        numlines+=1
#                                       print 'last_created',last_created
#adicionado o last_created
                        limit=last_created
                        print 'linha limite inferior',limit
                        print 'Numero de Linhas',numlines,'Called Address:',g,'Caller Address:', p,'Called Port:',h, 'Caller Port:', r
,'Time:', s, 'Codec:', t,'DisconnectionCode:',dcode
                        print 'todas as rodadas finalizadas, last created = ', last_created


#---------------------------------------------------------------------


csv_reader()
```

Sub-módulo MOS de Usuário
Descrição: Sub-módulo destinado a inserir o MOS de usuário e variações temporais assincronamente.
Este submódulo utiliza o micro-arcabouço Web Bottle.py e permite que o usuário, a partir de ua tela Web em um browser insira a sensação/percepção de usuário em termos de uma nota de 1 a 5 em MOS. É possível também, para efeito de testes realizados no experimento, atribuir fatores de atraso à informação, de tal forma que simule uma caducidade da informação. A rotina principal está associada a dois templates HTML, para permitir a entrada de dados no browser e a resposta. Este modelo pode ser expandido para aplicações mais complexas ou ser embutido em alguma aplicação que possibililite ao usuário informar seu nível de satisfação.

Rotina de Inserção de MOS de usuário

```
#!/usr/bin/python

import bottle
import pymongo
import time
import sys
import datetime
```

```
import socket
import struct, time
import time
from time import sleep
from socket import AF_INET, SOCK_DGRAM

#---------------------------------------------------------------------

connection = pymongo.MongoClient('mongodb://admin:sxsxsxsx@ds029541.mongolab.com:29541/context')
db=connection['context']
contextual = db.contexto6
profile= db.userprofile

@bottle.route('/')

def index():
        return bottle.template('mainmosu')
@bottle.post('/mosinsert')

def survey():

        user_id = bottle.request.forms.get("user_id")
        user_mos = bottle.request.forms.get("MOS_u")
        timedev = bottle.request.forms.get("timedev")

        print user_id
        mos_u=float(user_mos)
        timedev=int(timedev)
        print timedev

        print user_mos

#       print samples

        if (user_mos == None or user_mos == ""):
                user_mos="No MOS defined..."
        app="sip_pbx_call"
        t1=getNTPTime()
#   simula erro, soma 120 segundos
        t1-=timedev
        contextual.update({"user_id":5},{"$set":{'Context_Update':{'Update':1}}},upsert=True)
        contextual.update({"user_id":5},{"$set":{'User_Perception':{'application_id':app,'MOS_u':mos_u,'ti
mestamp':t1}}},upsert=True)
        return bottle.template('mosinsert', {"user_id":user_id, "user_mos":user_mos})


#---------------------------------------------------------------------

def getNTPTime(host = "a.st1.ntp.br"):

        port = 123
        buf = 1024
        address = (host,port)
        msg = '\x1b' + 47 * '\0'

#reference time (in seconds since 1900-01-01 00:00:00)
        TIME1970 = 2208988800L # 1970-01-01 00:00:00

        # connect to server
        client = socket.socket( AF_INET, SOCK_DGRAM)
        client.sendto(msg, address)
        msg, address = client.recvfrom( buf )

        t = struct.unpack( "!12I", msg )[10]
        t -= TIME1970
#       print t
        return (t)
```

bottle.run(host='localhost', port=8080, debug=1)

Template Mainmosu.tpl

```
<!DOCTYPE html>
<html>

<head>
<title>Context Controller Admin</title>
</head>

<body>
<p>Experience Sampling Method - Coleta a sensação do Usuário Diretamente: </p>

<form action="/mosinsert" method="POST">
        Informe o User ID: <br>
        <input type="text" name="user_id" size=40 value=""><br><br>

        Informe uma nota de 1 a 5 para a qualidade da aplicação: <br>
        <input type="text" name="MOS_u" size=40 value=""><br><br>

        Informe o desvio temporal em segundos: <br>
        <input type="text" name="timedev" size=40 value=""><br><br>

        <input type="submit" value="Enviar"><br>

</form>


</body>
</html>
```

Template Mosinsert.tpl

```
<!DOCTYPE html>
<html>
<head>
<title>Resultado</title>
</head>

<body>
<h1>MOS Inserido: {{user_mos}}</h1><br>
Com sucesso para usuário {{user_id}}
</body>

</html>
```

Sub-módulo Gerenciamento de Encaminhamento
Descrição: Este módulo pertence à Abordagem de Encaminhamento e é o responsável por receber as notificações encaminhadas pelo Gerenciador de Contexto.
Este submódulo recebe a notificação de violação de QoE por parte do Gerenciador de encaminhamento por meio de uma fila de notificações, que é permanentemente lida. A fila conté a informação acerca do flow_id e com base neste parâmetro o Gerenciador de Encaminhamento recupera os documentos de contexto, e com base nos cálculos efeitos sobre estes parâmetros, notifica o Adaptadore de Fluxos para realizar um comando iptables.

Submódulo Forward Management

```
import pymongo
import sys
import time
import csv
import datetime

from time import sleep
connection = pymongo.MongoClient('mongodb://admin:sxsxsxsx@ds029541.mongolab.com:29541/context')
db=connection['context']
contextual = db.contexto6
q = db.notifyfw
flowadapt = db.flowadapt
event=db.eventos


def find():

########################################
#A rotina fica em loop          while
#percorre uma fila de notificacao para encontrar o flow_id
########################################

        while True:
                query = {'session':'true'}
                cursor = q.find(query)
                for queue in cursor:
                        g=queue.get('flowidqueue')
                        id=queue['_id']
                        print 'id ='
                        print id
                        print 'flowidqueue:'
                        print g
#se a fila nao estiver vazia, encontrar o proximo elemento da fila
#retira o elemento da fila pelo comando pop:-1, pop invertido

                        if not (g==[]):
                                flow_to_treat=g[0]
                                q.update({'_id':id},{'$pop':{'flowidqueue':-1}}, upsert=True)
                                print 'chamar a function forwarding, informando o flow_id =',
flow_to_treat
                                forwarding(flow_to_treat)
                        else:
                                print 'fila vazia, entra em sleep, aguarda nova notificacao em loop'
```

```
                                        sleep (0.5)


def forwarding(flow):

        flow_to_treat=flow
        print 'function forwarding'
        print 'flow_to_treat',flow_to_treat
        doc = contextual.find_one({'flow_id': flow_to_treat})
        flow = flowadapt.find_one({'flow_id': flow_to_treat})

        # leitura dos dados de contexto:network, calcular o range de portas para determinar o comando de
fila do iptables

        m=doc.get('Network')
        sportmax = m['src_port_max']
        sportmin = m['src_port_min']
        dportmax = m['dest_port_max']
        dportmin = m['dest_port_min']
        if sportmin < dportmin:
                lrange=sportmin
        else:
                lrange=dportmin
        if dportmax > sportmax:
                hrange=dportmax
        else:
                hrange=sportmax

        print 'range'
        print lrange,hrange
        m=doc.get('QoE')
        cc=m['curr_calls']
        m=doc.get('sip_session')
        codec=m['Codec']
        print codec
        if codec == "G711A":
                bandwidth = int (cc*100)
        if codec == "G729":
                bandwidth = int (cc*45)
        if codec == "G723":
                bandwidth = int (cc*30)

        #procurar pelo mapa de filas e determinar a nova fila#

        g=flow.get('queue')
        comp=len(g)
        obf=flow['_id']
        print obf
        print 'comprimento'
        print comp
        print 'bandwidth'
        print bandwidth
        print 'queue0'
        print g[0]['description']
        i=0
        updt=0
        while ((i<(comp-1)) and (updt==0)):
                i+=1
                dif=g[i]['description']-bandwidth
                print dif
                if (dif > 0):
                        print 'update ==1'
                        updt=1
        print 'update:', updt, 'e queue: ', i
        if updt==1:
```

```
        flowadapt.update({"_id":obf},{"$set":{'eventforwarding':{'evtfrwid':1,'evtid':1,'pathid':flow_to_trea
t,'queueid':i}}},upsert=True)
                print 'NOVA FILA:'
                print i


#############################################################################################
#######################
# espera pelo comando ser aplicado ao iptables
# o modulo iptloop deve aplicar a regra para alocar a fila nova
# o iptloop, que roda no Gateway aplica a regra no netfilter e marca evitid como um valor diferente de 1 e o
programa sai do loop de espera
#############################################################################################
#######################

                while (updt==1):
                        flow = flowadapt.find_one({'flow_id': flow_to_treat})
                        u=flow.get('eventforwarding')
                        updt=u['evtid']
                        print 'aguardando o iptloop ser acionado e marcar evtid com 55, que neste
momento corresponde a:',updt
                        sleep(0.5)

# zera o event id, e retorna para o loop geral
        print 'zerando o eventid, na fila:',i
        flowadapt.update({"_id":obf},{"$set":{'eventforwarding':{'evtfrwid':0,'evtid':0,'pathid':flow_to_trea
t,'queueid':i}}},upsert=True)
        t2=time.time()
# campo fwdmgtliber marca o tempo em que foi liberado o flow adapter
        event.save({'fwdmgtliberado':t2})
        flowadapt.update({"_id":obf},{"$push":{'eventlog':{'timestamp':t2, 'newqueue':i}}},upsert=True)
#---------------------------------------------------------------------

        return()


find()
```

Módulo Gerenciamento de Encaminhamento
Descrição: Este módulo pertence à Adaptação de Fluxos e é o responsável por receber a notificação encaminhada pelo Gerenciador de Encaminhamento e realizar a adaptação do fluxo.
Este submódulo recebe a notificação do adaptador de fluxos acerca da nova fila a ser determinada no Netfilter. Esta rotina, nesta implementação, está sendo executada dentro do gateway, embora possa ser executada em qualquer localidade. A ação final tomada pelo módulo é a aplicação do comando iptables e atualizar o flag de update na base de contexto. Esta liberação permite que novos dados sejam coletados pelo Handler.

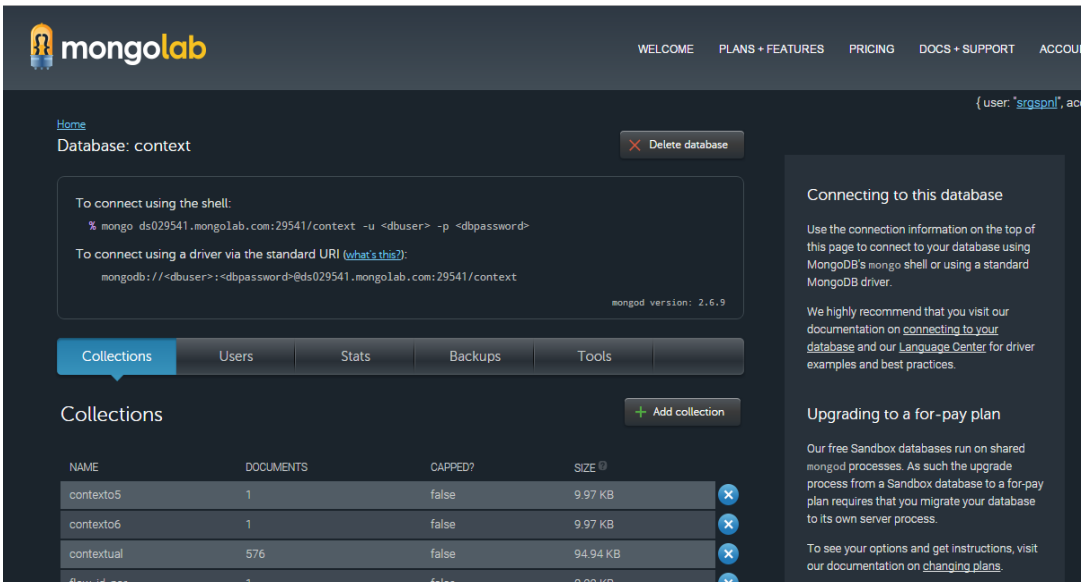
Submódulo Flow Adaptor

```
import subprocess
import sys
import datetime
import time
import pymongo
from time import sleep

connection = pymongo.MongoClient('mongodb://admin:sxsxsxsx@ds029541.mongolab.com:29541/context')
db=connection['context']
contextual = db.contexto6
flowadapt = db.flowadapt
#---------------------------------------------------------------------

def iptloop():
        while True:
                print 'sleep 5'
                sleep (5)
                print 'chama ipt'
                ipt()


def ipt():
#loop de busca por flowadatp com evtfwrdid = 1
#por simplicidade, neste caso, flow_to_treat =4
#pega o documento referente ao flowadapt e obtem a fila/classe designada

        flow_to_treat=4
        e=0
        while (e==0):
                f = flowadapt.find_one({'flow_id': flow_to_treat})
                n=f.get('eventforwarding')
                print 'estado do eventforwarding: ',n
                i=n['queueid']
                m=f.get('queue')
                print 'classe atual', m[i]['class']
                q=m[i]['class']
                e=n['evtid']
#               print 'e=',e
                sleep(2.0)
#               print 'sleep de 2 segundos'
#---------------------------------------------------------------------
# comandos que se aplicam no netfilter, no caso do software residente no Linux

        if q=="1:10":
```

```
                p=subprocess.Popen(["iptables","-t","mangle","A","FORWARD","-o","eth1","-
p","udp","-m","multiport","--sport","16000:16500","-j","CLASSIFY","--set-class", "1:10"],
stdout=subprocess.PIPE)
        if q=="1:20":
                p=subprocess.Popen(["iptables","-t","mangle","A","FORWARD","-o","eth1","-
p","udp","-m","multiport","--sport","16000:16500","-j","CLASSIFY","--set-class", "1:20"],
stdout=subprocess.PIPE)
        if q=="1:30":
                p=subprocess.Popen(["iptables","-t","mangle","A","FORWARD","-o","eth1","-
p","udp","-m","multiport","--sport","16000:16500","-j","CLASSIFY","--set-class", "1:30"],
stdout=subprocess.PIPE)
        if q=="1:40":
                p= subprocess.Popen(["iptables","-t","mangle","A","FORWARD","-o","eth1","-
p","udp","-m","multiport","--sport","16000:16500","-j","CLASSIFY","--set-class", "1:40"],
stdout=subprocess.PIPE)
        if q=="1:50":
                p=subprocess.Popen(["iptables","-t","mangle","A","FORWARD","-o","eth1","-
p","udp","-m","multiport","--sport","16000:16500","-j","CLASSIFY","--set-class", "1:50"],
stdout=subprocess.PIPE)
        if q=="1:60":
                p=subprocess.Popen(["iptables","-t","mangle","A","FORWARD","-o","eth1","-
p","udp","-m","multiport","--sport","16000:16500","-j","CLASSIFY","--set-class", "1:60"],
stdout=subprocess.PIPE)
        if q=="1:70":
                p=subprocess.Popen(["iptables","-t","mangle","A","FORWARD","-o","eth1","-
p","udp","-m","multiport","--sport","16000:16500","-j","CLASSIFY","--set-class", "1:70"],
stdout=subprocess.PIPE)

################################################
# marcar o evtfrwdrid como 55
# zerar o update na base de contexto
# com a zeragem, o contexto pode ser renovado
################################################

        f = flowadapt.find_one({'flow_id': flow_to_treat})
        obf=f['_id']
        flowadapt.update({'_id':obf},{'$set':{'eventforwarding':{'evtid':55}}})
        query={"flow_id":flow_to_treat}
        cntx = contextual.find_one(query)
        obid = cntx['_id']
        m=cntx.get('QoE')
        moson=m['MOS']
        print 'reefileiramento referente a MOS  =',moson
        print ('zerando o campo Update no documento de contexto')
        contextual.update({'_id':obid},{"$set":{'Context_Update':{'Update':0}}},upsert=True)
        return()


iptloop()
```

APÊNDICE D – Representação do banco de dados JSON/BSON

O MongoDB é disponibilizado como serviço de banco de dados por diversos provedores de serviços de armazenagem na Nuvem. O MongoLab [MongoLab, 2015] é um serviço de administração e gerência de hospedagem de banco de dados MongoDB, que propicia a publicação e utilização de várias bases de dados para fins de prototipação – limitadas ao espaço de 500MB no MongoLab, porém suficientes para os propósitos desta implementação, motivo pelo qual foi escolhido como plataforma de serviços de banco de dados na Nuvem neste trabalho.



Serviço de banco de dados como serviço, coleções de MongoDB [MONGOLAB, 2015]

Na abordagem deste estudo favorecemos o uso da abordagem não normalizada, contudo, não integralmente, tendo sido usado um número reduzido de *collections*, sobretudo para facilitar o entendimento. Embora fosse possível modelar os dados representados neste projeto em um único documento JSON, tal abordagem a princípio dificultaria o entendimento da estrutura e poderia representar um aumento de complexidade indesejável. Desta forma, foram criadas duas grandes "*collections*", sendo uma associada ao administrador do sistema, contemplando todos os dados cadastrais de natureza estática, e uma segunda *collection*, que representa o contexto dinâmico em si. Essencialmente a conexão ao banco é realizada através do driver de conexão para Python, a biblioteca pymongo, que permite realizar a conexão ao banco na nuvem utilizando as credenciais de acesso fornecidas pelo MongoLab. No exemplo abaixo

encontram-se as linhas de código Python que permitem realizar a conexão ao banco de contexto e abrir uma conexão para a collection (coleção) denominada contexto6.

```
import pymongo

connection=pymongo.MongoClient('mongodb://admin:sxsxsxsx@ds029541.mongolab.com:29541/context')
db=connection['context']
contextual = db.contexto6
```

Foram criados documentos de apoio que servem ao propósito de notificação e logs, e que fazem uso dos recursos de tratamento de filas e pilhas embutidos no MongoDB. De maneira geral, buscou-se o objetivo de manter um compromisso entre a inteligibilidade e o desempenho.

Coleção de Documentos de Contexto (Context6)
Descrição: documentos que descrevem o estado atual dos parâmetros de contexto referentes a um dado usuário, sessão e fluxo.
Descrevem o usário, sessão, fluxo, sessão SIP, tempo, localização, QoD, QoS, o SLA da aplicação e perceção do usuário, o flag de update é auxiliar na sinalização entre os módulos.

```
{
    "_id" : ObjectId("55d79626e4b0550b9e2b303c"),
    "session" : NumberInt(5),
    "user_id" : NumberInt(5),
    "flow_id" : NumberInt(4),
    "update" : NumberInt(1),
    "sip_session" : {
        "Codec" : "G723",
        "protocol" : "udp",
        "rtp" : "true"
    },
    "time" : {
        "NTP_server" : "a.st1.ntp.br",
        "Last_NTP_Update" : "string"
    },
    "Location" : {
        "latitude" : -12.9611703,
        "longitude" : -38.432011,
        "altitude" : "none",
        "atm_pressure" : "none"
    },
    "QoD" : {
        "has_gps" : "false",
        "gps_precision" : "none",
        "process_cores" : "two",
        "screen_resolution" : "string",
        "processor_overload" : "false",
        "resource_fault" : "string"
    },
    "Network" : {
        "src_port_max" : NumberInt(16086),
        "src_ip" : "192.168.1.101",
        "src_port_min" : NumberInt(16000),
        "dest_port_min" : NumberInt(16100),
        "flow_id" : NumberInt(4),
        "dest_port_max" : NumberInt(16186),
        "dest_ip" : "192.168.0.102"
    },
    "QoE" : {
```

```
        "MOS" : 1.0977152857142856,
        "timestamp" : 1.441146023E9,
        "curr_calls" : 28.3,
        "Codec" : "G723",
        "R-factor" : 9.018165999999999,
        "time" : "2015-09-01 19:20:23.122"
    },
    "QoS" : {
        "p_loss" : 20.87575,
        "round_trip_delay" : 1839.369625,
        "jitter" : 6.133
    },
    "SLA_APP" : {
        "application_name" : "sip_pbx_call",
        "MOS_SLA" : NumberInt(4),
        "Change_Priority" : "false",
        "Requirements" : [
            "QoE",
            "QoS",
            "Network",
            "sip_session"
        ]
    },
    "User_Perception" : {
        "timestamp" : NumberLong(1441145777),
        "MOS_u" : 2.5,
        "application_id" : "sip_pbx_call"
    },
    "Context_Update" : {
        "Update" : NumberInt(0)
    }
}
```


Coleção de Perfil de Usuário (User Profile)
Descrição: documentos que descrevem o cadastro de usuário, as localidades registradas, Codecs admitidos, conforme a aplicação, dados cadastrais gerais, de rede, e requisitos.
Descrevem o cadastro detalhado do usuário, parâmetros da aplicação, requisitos de codecs e SLA, rede, e um parâmetro ‘samples’, que está representado, embora não tenha sido utilizado, para indicar a taxa de amostragem para o agente de contexto.


```
{
    "_id" : ObjectId("55bbf0e4e4b0230ce9104c57"),
    "user" : {
        "nome_do_cliente" : "string",
        "razão_social" : "string",
        "CNPJ" : "string",
        "endereço" : "string",
        "cep" : "cep",
        "pais" : "brasil",
        "cidade" : "salvador",
        "bairro" : "pituba",
        "user_id" : NumberInt(5)
    },
    "sip_register" : {
        "sip_user" : "sip:telco@ip10.com.br",
        "rtp_port_ini" : "16000",
        "rtp_port_fim" : "16500"
    },
    "Location" : {
        "latitude" : -12.9612703,
        "longitude" : -38.434011
    },
    "Device" : [
        {
            "device_id" : NumberInt(177),
            "device_name" : "string",
            "mac_address" : "cc:a4:62:dd:be:50",
            "mobile" : "true",
```

```
            "connection" : "802.1ac",
            "os" : "android",
            "has_gps" : "false",
            "gps_precision" : "none",
            "process_cores" : "two",
            "screen_resolution" : "string"
        },
        {
            "device_id" : NumberInt(155),
            "device_name" : "string",
            "mac_address" : "9e:8c:63:58:96:65",
            "mobile" : "true",
            "connection" : "802.1g",
            "os" : "ios",
            "has_gps" : "false",
            "gps_precision" : "none",
            "process_cores" : "two",
            "screen_resolution" : "string"
        }
    ],
    "Network" : {
        "ip_v4" : [
            "192.168.1.101",
            "75.82.0.37"
        ],
        "ip_v6" : [
            "FE80::0202:B3FF:FE1E:8329"
        ]
    },
    "SLA_APP" : [
        {
            "application_name" : "sip_pbx_call",
            "MOS_SLA" : NumberInt(4),
            "max_calls" : NumberInt(30),
            "Requirements" : [
                "QoE",
                "QoS",
                "Network",
                "sip_session"
            ],
            "audio_codecs" : [
                "G729",
                "G711"
            ]
        },
        {
            "application_name" : "sip_conf",
            "MOS_SLA" : NumberInt(4),
            "max_calls" : NumberInt(5),
            "Requirements" : [
                "QoE",
                "QoS",
                "Network",
                "sip_session"
            ],
            "video_codecs" : [
                "H263",
                "H264"
            ]
        }
    ],
    "samples" : NumberInt(25)
}
```


Coleção de Documentos de Log de violação de QoC
Descrição: documentos que descrevem os logs coletados.
Descrevem os logs coletados por usuário e o tipo de violação e código, em um array de sub-documentos.

```
{
    "_id" : ObjectId("55d1f398e4b0548e0c84fe90"),
    "user_id" : NumberInt(5),
    "log" : [
                {
            "code" : NumberInt(90),
            "timestamp" : 1.440021288501E9,
            "Codec" : "G723",
            "flow_id" : NumberInt(4)
        },
        {
            "code" : NumberInt(90),
            "timestamp" : 1.440021511861E9,
            "Codec" : "G723",
            "flow_id" : NumberInt(4)
        },
        {
            "timestamp" : 1.441043939E9,
            "timestamp violation" : 397.6229999065399,
            "code" : NumberInt(10),
            "flow_id" : NumberInt(4)
        },
        {
            "code" : NumberInt(90),
            "timestamp" : 1.441122047794E9,
            "Codec" : "G723",
            "flow_id" : NumberInt(4)
        },


       ... {

            "timestamp" : 1.441145881075E9,
            "code" : NumberInt(100),
            "MOS-u" : 2.5,
            "flow_id" : NumberInt(4)
        }
    ]
}
```


Fila de Notificação entre sub-módulo de Gerenciamento de Contexto e o Gerenciamento de Encaminhamento
Descrição: documento que descreve a fila de flow_ids a serem tratados.
O documento notifyfw possui um campo array “flowidqueue” que implementa uma FIFO. O primeiro número a entrar (4) é também o primeiro a sair (4).


```
{
    "_id" : ObjectId("55a2b429e4b0b251e71411d8"),
    "notification" : "true",
    "session" : "true",
    "flowidqueue" : [
        NumberInt(4),
        NumberInt(6)
    ]
}
```

APÊNDICE E – REPRESENTAÇÃO DAS REGRAS DE QOC

Representação das Regras de QoC
Descrição: Regras de QoC representadas no MongoDB - MongoLab.
Documentos de representação de QoC, descrição, condição de desvio, código de violação para efeito de log.

```
{
    "_id" : ObjectId("55c53153736e393aa4b5dbfc"),
    "code" : NumberInt(10),
    "description" : "timestamp violation",
    "deviation" : NumberInt(120),
    "action" : "drop"
}
{
    "_id" : ObjectId("55c53153736e393aa4b5dbfd"),
    "code" : NumberInt(40),
    "description" : "location not valid",
    "longitude" : 0.01,
    "latitude" : 0.01,
    "action" : "alarm"
}
{
    "_id" : ObjectId("55c53159736e39bb70c1f1ce"),
    "code" : NumberInt(100),
    "description" : "User perception mismatch",
    "deviation" : 0.7,
    "action" : "alarm"
}
{
    "_id" : ObjectId("55cf9fce699d8e0344d9b2a2"),
    "code" : NumberInt(60),
    "description" : "video codec invalid",
    "action" : "drop"
}
{
    "_id" : ObjectId("55d4d355e4b095eeb88a5f46"),
    "code" : NumberInt(90),
    "audio-codec" : [
        "G729",
        "G711A"
    ],
    "description" : "audio codec invalid",
    "action" : "drop"
}
{
    "_id" : ObjectId("55e03a4ae4b01b38147dd0e4"),
    "code" : NumberInt(20),
    "description" : "Max concurrente calls",
    "deviation" : 0.1,
    "action" : "alarm and drop"
}
```

```
{
    "_id" : ObjectId("5616e0efe4b04099f9011667"),
    "code" : NumberInt(200),
    "description" : "user not registered",
    "deviation" : [
        "non-alpha",
        "not in profile base"
    ],
    "action" : "drop"
}
{
    "_id" : ObjectId("5616e14ee4b04099f901166b"),
    "code" : NumberInt(300),
    "description" : "source_ip",
    "deviation" : "source_ip not registered in profile base",
    "action" : "drop"
}
{
    "_id" : ObjectId("5616e1a8e4b04099f9011675"),
    "code" : NumberInt(301),
    "description" : "dest_ip",
    "deviation" : "dest_ip not registered in profile base",
    "action" : "drop"
}
{
    "_id" : ObjectId("5616e271e4b04099f901168f"),
    "code" : NumberInt(400),
    "description" : "Rfactor calculation into MOS not valid",
    "deviation" : 0.1,
    "action" : "drop"
}
{
    "_id" : ObjectId("5616e2efe4b04099f9011699"),
    "code" : NumberInt(500),
    "description" : "GPS not numeric",
    "deviation" : "alpha not valid",
    "action" : "drop"
}
{
    "_id" : ObjectId("5616e341e4b04099f901169f"),
    "code" : NumberInt(410),
    "description" : "MOS out of range",
    "deviation" : [
        NumberInt(1),
        NumberInt(5)
    ],
    "action" : "drop"
}
{
    "_id" : ObjectId("5616e39fe4b04099f90116a2"),
    "code" : NumberInt(420),
    "description" : "R-factor out of rangedest_ip",
    "deviation" : [
        NumberInt(1),
        NumberInt(100)
    ],
    "action" : "drop"
}
{
```

```
    "_id" : ObjectId("5616e3ebe4b04099f90116aa"),
    "code" : NumberInt(600),
    "description" : "UDP fault",
    "deviation" : "UDP was expected and not found",
    "action" : "drop"
}
{
    "_id" : ObjectId("55e03abae4b01b38147dd0f0"),
    "code" : NumberInt(50),
    "description" : "RTP port range mismatch",
    "deviation" : NumberInt(0),
    "action" : "alarm and drop"
}
{
    "_id" : ObjectId("5616e547e4b04099f90116c1"),
    "code" : NumberInt(55),
    "description" : "SIP port range mismatch",
    "deviation" : NumberInt(0),
    "action" : "alarm and drop"
}
{
    "_id" : ObjectId("5616e55ce4b04099f90116c3"),
    "code" : NumberInt(58),
    "description" : "SDP port range mismatch",
    "deviation" : NumberInt(0),
    "action" : "alarm and drop"
}
{
    "_id" : ObjectId("5616e5c9e4b04099f90116cb"),
    "code" : NumberInt(650),
    "description" : "source port range mismatch",
    "deviation" : [
        NumberInt(1),
        NumberInt(65365)
    ],
    "action" : "alarm and drop"
}
{
    "_id" : ObjectId("5616e5f0e4b04099f90116ce"),
    "code" : NumberInt(670),
    "description" : "destination port range mismatch",
    "deviation" : [
        NumberInt(1),
        NumberInt(65365)
    ],
    "action" : "alarm and drop"
}
{
    "_id" : ObjectId("5616e663e4b04099f90116d9"),
    "code" : NumberInt(800),
    "description" : "flow_id",
    "deviation" : "not integer",
    "action" : "alarm and drop"
}
{
    "_id" : ObjectId("5616e6a9e4b04099f90116dc"),
    "code" : NumberInt(900),
    "description" : "user not alphanumeric",
    "deviation" : "not alpha",
```

```
    "action" : "alarm and drop"
}
{
    "_id" : ObjectId("5616f135e4b04099f90117fe"),
    "code" : NumberInt(850),
    "description" : "background_noise_decibels",
    "deviation" : [
        NumberInt(55)
    ],
    "action" : "alarm and drop"
}
{
    "_id" : ObjectId("5616f2d7e4b04099f901180c"),
    "code" : NumberInt(890),
    "description" : "below_min_pixel_resolution",
    "deviation" : [
        "CIF",
        "352x288"
    ],
    "action" : "alarm and drop"
}
{
    "_id" : ObjectId("5616fec6e4b04099f90118be"),
    "code" : NumberInt(980),
    "description" : "sip_session_not_valid",
    "deviation" : "not alphanumeric",
    "action" : "alarm and drop"
}
{
    "_id" : ObjectId("561718fbe4b04099f9011a35"),
    "code" : NumberInt(9900),
    "description" : "application_not_defined",
    "deviation" : "not registered",
    "action" : "alarm and drop"
}
```

ANEXO A – Cálculos de Mos e R-FACTOR

Software de Emulação de Chamadas e Coleta Sartriniy realiza as sequências de chamadas e fornece a avaliação de dados, que preenche uma base de dados. Esta base pode ser SQL ou, opcionalmente, pode ser preenchido continuamente um arquivo CSV. O módulo de Agente de Middleware possui uma rotina que lê o arquivo de CSV e remove do mesmo apenas os novos dados disponíveis.

Foi solicitado à Startrinity que tornasse disponível a rotina em C que efetua os cálculos de MOS e R-Factor a partir dos parâmetros de QoS apurados. A listagem da parte do código fonte que é tornada acessível é listada a seguir.

Código Fonte em C do cálculo de R-Factor[Startrinity]

```
 float burstRate = ((float)lostPacketsCount / packetsAndJbStatistics.burstsCount) /
                   (1.0f / (1.0f - (float)lostPacketsCount /
packetsAndJbStatistics.detectedPacketsCount));
            if (burstRate < 0) burstRate = -burstRate;

            float packetLossRate =
(float)lostPacketsCount/packetsAndJbStatistics.detectedPacketsCount;

        // according to G.113 Table I.3
        double Ie; // equipment impairment factor
             double Bpl; // packet-loss robustment factor
             double R0 = 93.2; // basic SNR
             if (lastPacketRtpHeader.pt == PayloadType G729)
             {
                 Ie = 11; // assuming G729-A with VAD, ptime=20
                 Bpl = 19;
             }
             else if (lastPacketRtpHeader.pt == PayloadType G723)
        {
            Ie = 15; // assuming G.723.1 6.3kB/s, ptime=30
            Bpl = 16.1;
        }
             else
             {
                 // else assuming that it is G711, ptime=10, with PLC
                 Ie = 0;
                 Bpl = 25.1;
             }
             double Ppl = packetLossRate*100;
        double Ie_eff = Ie+(95-Ie)*Ppl/(Ppl/burstRate+Bpl); // equation 7-29 from G.107E
             double R = R0-Ie;
        if (Ppl != 0 && burstRate != 0)
            R -= Ie_eff;
        if (R > 100) R = 100; else if (R < 0) R = 0;
             rtpStreamStatistics->G107R = (float)R;
        float mos = (float)(1 + 0.035 * R + R * (R - 60) * (100 - R) * 7E-6); //
from http://www.hamilton.ie/dwmalone/assem et al.pdf
        if (mos < 1) mos = 1;
        else if (mos > 4.5f) mos = 4.5f;
             rtpStreamStatistics->G107MOS = mos;
```

[Sergey Aleshin, Developer, Owner
LinkedIn: http://www.linkedin.com/pub/sergey-a/17/980/824 | Web: startrinity.com | Russia, Kazan, 420127 Chuikova street, building 64/11]