UNIFACS UNIVERSIDADE SALVADOR
MESTRADO ACADÊMICO EM SISTEMAS E COMPUTAÇÃO

JOATHAM PEDRO SANTOS DA SILVA

IMPLEMENTAÇÃO DE UMA ARQUITETURA PARA O ENCAMINHAMENTO BASEADO NO ARCABOUÇO CONTEXT-AWARE ADAPTIVE ROUTING FRAMEWORK (CAARF)

Salvador
2015

JOATHAM PEDRO SANTOS DA SILVA

IMPLEMENTAÇÃO DE UMA ARQUITETURA PARA O ENCAMINHAMENTO BASEADO NO ARCABOUÇO CONTEXT-AWARE ADAP TIVE ROUTING FRAMEWORK (CAARF)

Dissertação apresentada à Coordenação do Curso de Mestrado em Sistemas e Computação da UNIFACS Universidade Salvador, Laureate International Universities como requisito parcial para a obtenção do título de Mestre.

Orientador: Prof. Dr. Paulo Sampaio.

Salvador
2015

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

Silva, Joatham Pedro Santos da

Implementação de uma arquitetura para o encaminhamento baseado no arcabouço Context-Aware Adaptive Routing Framework (CAARF)./ Joatham Pedro Santos da Silva.- Salvador: UNIFACS, 2015.

133 f. : il.

Dissertação Programa de Pós-Graduação em Sistemas e Computação de UNIFACS Universidade Salvador, Laureate International Universities como requisito parcial à obtenção do título de Mestre.

Orientador: Prof. Dr. Paulo Sampaio.

1. Redes de computadores. 2. Qualidade de serviço (QoS). 3.Framework. I. Sampaio Paulo, orient. II. Título.

CDD: 004.6

RESUMO

Com a evolução das tecnologias multiplica-se a cada dia o número de usuários, dispositivos e aplicações, contudo, o aumento da largura de banda e da velocidade das transmissões não é o suficiente para garantir a qualidade da entrega e recebimento dos pacotes que trafegam nesta rede. Afim de atender as necessidades dessas novas aplicações, é necessário considerar diferentes aspectos relacionados à infraestrutura de comunicação, tais como controle de tráfego, alocação e gerência de recursos, policiamento, controle de admissão, escalabilidade, entre outros. Atualmente a gerência de tráfego de aplicações em tempo real é realizada baseada nos parâmetros de qualidade de serviço. No entanto, dadas as características emergentes dos usuários e suas aplicações, tais como, mobilidade, flexibilidade e maior demanda de conteúdo, essa gerência tem se tornando ineficiente, levando a uma insatisfação do usuário final na entrega de conteúdo. A proposta de redes sensíveis ao contexto surge como uma solução para mitigar esse problema tornando uma comunicação de dados orientada às características da rede em uma comunicação orientada aos requisitos e características do usuário e dos dispositivos de comunicação que ele utiliza. O conceito de rede sensível ao contexto é aplicado nesta dissertação através da implementação de uma solução para o encaminhamento adaptativo baseado em contexto, contemplando o comportamento e as expectativas do usuário, do sistema computacional e dos recursos de rede. Com este trabalho é possível demonstrar que a implementação de redes sensíveis ao contexto permite melhorar a experiência do usuário e otimizar o acesso aos recursos da rede.

Palavras-chave: Redes Sensíveis ao Contexto. Qualidade de Contexto. Qualidade de Serviço. Qualidade de Experiência. Qualidade de Dispositivos. Encaminhamento Adaptativo.

ABSTRACT

With the evolution of technologies, the number of users, devices and applications increases every day. However, even with the increase on bandwidth and transmission speed the resources available are not enough to ensure delivery and transmission quality of the packets. In order to meet the requirements of these new applications it is important to consider different aspects of the communication platform, such as traffic control, allocation and resources management, policing, admission control, scalability, among others. Currently, the traffic management of real-time applications is carried out based on quality of service metrics. However, given the emerging user's and application's requirements, such as mobility, flexibility, and larger demand of content, traffic management has become inefficient, taking the final user to be unsatisfied with the final delivery of content. The proposal of context-aware networks allows the mitigation of these problems turning data communication which is network oriented into a communication oriented to user's and communication devices' requirements and characteristics. The concept of context-oriented network is applied in this dissertation through the implementation of a solution for the context-aware adaptive routing, considering user's, computational system's and network resources' behavior and expectations. Through this work it is possible to demonstrate that the implementation of context-aware networks allows the improvement of user's experience and the optimization of network's resources access.

Keywords: Sensitive Networks Contextual, Context Quality, Quality of Service, Quality of Experience, Quality Device, and Adaptive Routing.

LISTA DE ABREVIATURAS

BSON	Binary JSON
CAARF	Context-Aware Adaptive Routing Framewok
CBQ	Class Based Queue
CDR	Call Detailed Record
EC	Elementos Contextuais
GMPLS	*Generalized Multi-Protocol Label Switching*
GPS	Global Positioning System
HTB	Hierarchical Token Bucket
JSON	Java Script Object Notation
LDAP	Lightweight Directory Access Protocol
MOS	*Mean Opinion Score*
MPLS	Multi-Protocol Label Switching
MPQM	Moving Picture Quality Metrics
NOSQL	Not Only SQL
NQM	Noise Quality Metric
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
RTCP	Real Time Transport Control Protocol
RTP	Real Time Transport Protocol
RTT	Round-Trip Time
SIP	Session Initiation Protocol
SSIM	Structural Similarity Index
TUQ	User Testing Perceived QoS
UAC	User Agent Client
UAS	User Agent Server

UDP	User Datagram Protocol
UUID	Universally Unique Identifier
XML	Extensible Markup Language

LISTA DE TABELAS

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

Com a evolução das tecnologias e consequentemente das redes de comunicações, multiplica-se a cada dia o número de usuários, dispositivos e aplicações. Junto a este crescimento vem a necessidade de transmissões mais rápidas e confiáveis, maiores larguras de banda, espaço de armazenamento e capacidade de processamento que garantam a satisfação deste cliente/aplicativos.

No entanto, apesar da recente disponibilidade de maior largura de banda e velocidade das transmissões, ainda é necessário proporcionar a garantia de entrega e recebimento dos pacotes que trafegam na rede em um tempo compatível com as exigências de cada aplicação. Afim de ir ao encontro dessas necessidades, alguns aspectos têm sido considerados na gerência de rede, tais como: controle de tráfego, alocação e gerência de recursos, policiamento, controle de admissão, escalabilidade, entre outros.

Ao longo dos últimos 20 anos, diferentes contribuições têm sido propostas na literatura a fim de fornecer garantias de tráfego em tempo real e garantir a utilização equilibrada dos recursos de rede. Em geral, a implementação de soluções de qualidade de serviço (QoS) tem sido a base de forma a atender as expectativas dos usuários. Para este efeito, diferentes contribuições, tais como Serviços Integrados (*IntServ*) (BRANDEN et al., 1994), Serviços Diferenciados (*DiffServ*) (ZHAO et al, 2000), MPLS/GMPLS (ROSEN et at., 1999) e Engenharia de Tráfego (CISCO, 2015) têm sido amplamente utilizadas sobretudo na última década (EL-GENDY et al., 2003).

Atualmente a gerência de tráfego de aplicações em tempo real, em geral, ainda é realizada baseada nos parâmetros de qualidade de serviço. No entanto, dadas às características emergentes dos usuários e das novas aplicações, tais como, mobilidade, flexibilidade e maior demanda de conteúdo, essa gerência tem se tornando ineficiente, levando a uma insatisfação do usuário final na entrega de conteúdo em tempo real e sensíveis ao tempo (SHAIKH et al., 2010).

De fato, as aplicações e serviços emergentes geram uma quantidade cada vez maior de tráfego de dados em tempo real para a rede. Infelizmente, as estratégias de infraestrutura de rede e roteamento não evoluíram no mesmo ritmo que aplicações

de dados, causando constante escassez de recursos e, consequentemente, sob congestionamento.

Portanto, o estudo da literatura indica que, em geral, as soluções de telecomunicação existentes são orientadas e processadas baseadas nos parâmetros de redes (tais como atraso, perdas, etc.). A fim de incorporar o conceito de uma rede centrada nas expectativas dos usuários, diferentes soluções também podem ser propostas visando aferir a mudança de comportamento do usuário e também as mudanças no ambiente de sistema computacional (SCHILIT et at., 1994). Dessa forma, a implementação de redes sensíveis ao contexto pode ser útil para melhorar a experiência do usuário e otimizar o acesso aos recursos da rede. O conceito de rede sensível ao contexto é utilizado nesta dissertação de forma a contemplar o comportamento e as expectativas do usuário, do sistema computacional e dos recursos de rede.

Para este efeito, o presente trabalho aborda a implementação de uma solução para o encaminhamento adaptativo baseado em contexto, utilizando três conceitos principais: Qualidade de Serviço, Qualidade de Experiência e Roteamento Adaptativo.

Esses fatores levam a identificação da problemática a seguir.

1.2 PROBLEMÁTICA

Nesse contexto, foi possível identificar os principais problemas que nortearam a motivação e a proposta deste trabalho de mestrado:

- Diversidade de aplicações, conteúdo e tipos de tráfego para internet, muito além do que ela foi projetada;
- Crescente introdução de redes heterogêneas (MPLS, GMPLS, IP, SDN, etc.);
- Gerenciamento baseado em QoS em certas situações não possui a abrangência necessária quando se considera a grande diversidade de variáveis em diversos cenários e contextos, e;
- A percepção dos usuários mediante ao conteúdo recebido é geralmente ignorada pelas redes atuais.

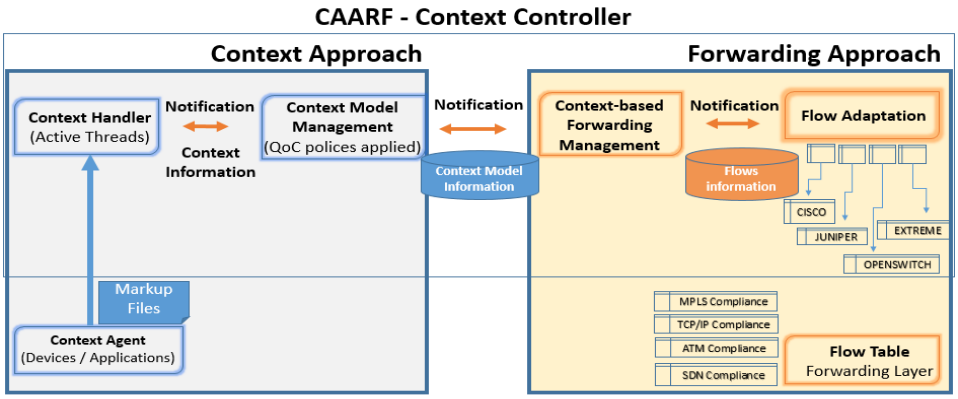
1.3 MOTIVAÇÃO

De forma a contornar os problemas identificados anteriormente, é necessária uma solução integrada para roteamento adaptativo que contemple os requisitos e expectativas do usuário, através do seu comportamento (perfil, mobilidade, satisfação, etc.), o comportamento do sistema computacional (características dos dispositivos de apresentação e comunicação, etc.) e as características e condições de tráfego da rede.

Nesse sentido, o arcabouço *Context-Aware Adaptive Routing Framework* (CAARF), ilustrado na **Erro! Fonte de referência não encontrada.**, tem sido proposto de forma a viabilizar as tomadas de decisões de encaminhamento de dados baseadas em contexto, sem desprezar as informações comumente usadas para Qualidade de Serviço (QoS). Portanto, essa solução permite avaliar não somente os parâmetros usualmente conhecidos como largura de banda, latência, variação no atraso e taxa de erros, mas também parâmetros como o nível de satisfação do usuário em relação à entrega de conteúdo, a capacidade de processamento momentâneo de um dispositivo ou mesmo a capacidade de memória momentânea do mesmo (SILVA, 2015; OLIVEIRA, 2015; SPINOLA, 2015; MUAKAD, 2015).

O projeto do arcabouço CAARF está sendo proposto e implementado como resultado da contribuição integrada de quatro alunos do Programa de Mestrado em Sistemas e Computação da Universidade Salvador (UNIFACS). Neste projeto, os diferentes alunos (Joatham Silva, André Oliveira, Sérgio Spínola e Frederico Muakad) estão realizando contribuições distintas e complementares de forma a proporcionar a modelagem, implementação e validação do arcabouço CAARF.

Figura 1 - Arcabouço *Context-Aware Adaptive Routing Framework* (CAARF)



O arcabouço CAARF, conforme ilustrado na **Erro! Fonte de referência não encontrada.**, possui dois componentes principais, O módulo de gerência de contexto (*Context Approach*) e o módulo de gerência de encaminhamento (*Forwarding Approach*). O módulo de gerência de contexto permite a coleta das notificações de alteração de contexto recebidas dos usuários, dos dispositivos de apresentação e dos dispositivos ativos de comunicação de rede. Esse módulo é responsável por triar essas notificações e indicar apenas aquelas que causam um impacto direto na mudança de contexto do sistema.

O módulo de gerência de encaminhamento é responsável por receber as notificações de mudança de contexto, aplicar as mudanças no contexto do sistema e implementar regras de encaminhamento de acordo com o novo contexto estabelecido. Esse módulo também está encarregado de atualizar as tabelas de fluxo dos equipamentos ativos de comunicação de acordo com as novas alternativas de encaminhamento para os fluxos.

Desta forma, a principal motivação para a proposta deste trabalho foi contribuir com a implementação de uma arquitetura para o arcabouço proposto através de uma abordagem para o encaminhamento adaptativo baseado em contexto. Os objetivos geral e específicos são apresentados na próxima seção.

1.4 OBJETIVOS

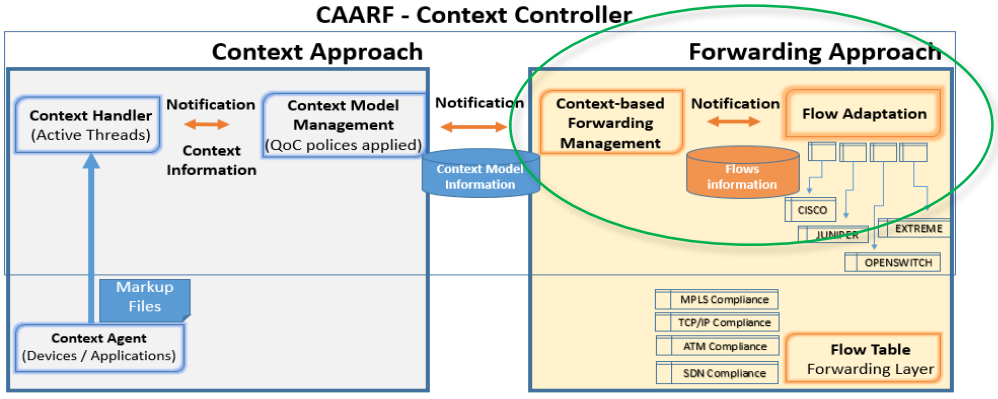
O trabalho apresentado nesta dissertação tem como objetivo geral a implementação de estratégias para a gerência de encaminhamento, como parte de uma arquitetura para a implementação de encaminhamento adaptativo baseado em

contexto, baseada no arcabouço CAARF. A implementação proposta visa promover a qualidade de experiência dos usuários na utilização dos serviços. Nesse sentido, o objetivo geral pode ser dividido em quatro objetivos específicos:

- **Estudo e definição de um modelo e arcabouço para o tratamento do contexto** e a representação das informações necessárias para o encaminhamento de dados;
- **Definição e modelagem de regras de encaminhamento** baseadas em informações contextuais dentro da arquitetura de contexto proposta;
- **Implementação da arquitetura proposta para a gerência de encaminhamento adaptativo** baseada no arcabouço *Context-Aware Adaptive Routing Framework (CAARF)*, e;
- **Proposta de um cenário de validação da solução proposta através da realização de estudos de casos** para otimização do tráfego baseado em contexto utilizando um cenário real.

Nesse sentido o enfoque deste trabalho de mestrado é realizado sobre o módulo de gerência de encaminhamento (*Forwarding Approach*), como ilustrado na **Erro! Fonte de referência não encontrada.**.

Figura 2 - Módulo de Encaminhamento (enfoque do trabalho)



A partir da modelagem e simulação do módulo de gerência de encaminhamento (MUAKAD, 2015), é possível a concepção e a implementação de uma solução para a implementação do processamento das informações contextuais e aplicação das regras de encaminhamento.

1.5 METODOLOGIA

Para a realização do trabalho proposto, a seguinte metodologia foi adotada:

- Realização de um estudo comparativo das soluções e modelos existentes para o encaminhamento baseado em contexto com Engenharia de Trafego;
- Estudo de trabalhos relacionados para auxiliar na proposta e posterior definição de um modelo e arcabouço para o tratamento de contexto e a representação das informações necessárias para o encaminhamento de dados;
- Definição e modelagem do arcabouço CAARF e proposta dos mecanismos e regras necessários para a gerência de encaminhamento baseada em contexto;
- Estudo e identificação de um conjunto de ferramentas necessárias para a implementação da solução proposta, tais como MongoDB, Python, Netfilter, SIP dentre outros;
- Implementação do módulo de gerência de encaminhamento baseado em contexto, e;
- Validação da solução implementada através de estudos de caso baseados em cenários reais, usando técnicas de tratamento de filas.

1.6 CONTRIBUIÇÕES E RESULTADOS ALCANÇADOS

As contribuições do trabalho realizado nesta dissertação de mestrado estão relacionadas com as áreas de roteamento adaptativo de tráfego, Qualidade de Serviço (QoS), Qualidade de Dispositivo (QoD), Qualidade de Experiência (QoE) e Qualidade de Contexto (QoC). Em cada área, foi dado um enfoque específico, como no caso, por exemplo, do **roteamento adaptativo**, visando a capacidade que o modelo e o arcabouço propostos têm se de adaptar durante o funcionamento da rede e as mudanças que podem ocorrer no fluxo e suas respectivas aplicações.

Outra contribuição abordada nesta dissertação é a ampliação da **visão do atendimento orientado ao usuário** e nas suas necessidades, objetivando uma melhor qualidade na sua comunicação. Desta forma, o usuário é tratado não somente

como um agente passivo, aceitando as escolhas de encaminhamento feitas pela rede, mas também um agente ativo, interagindo com a mesma através das suas experiências. Para tanto, no contexto deste trabalho, toda a análise do melhor caminho de encaminhamento dos dados utiliza efetivamente as informações coletadas do usuário para atendê-lo com uma comunicação de melhor qualidade.

Outra contribuição relevante foi a apresentação de **um arcabouço genérico que trata das questões de roteamento adaptativo de tráfego baseado em informações contextuais**. Para tanto, são apresentados os conceitos relacionados a esse arcabouço, bem como suas estruturas, seu funcionamento, a interação entre seus módulos, tratamento dos eventos, regras utilizadas e as entradas e saídas que ocorrem em cada módulo, mais especificamente do módulo de encaminhamento.

Por fim, como contribuição direta, foi proposta a **implementação do módulo de gerência de encaminhamento** baseado no arcabouço proposto e foram realizados cinco **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.

1.7 ORGANIZAÇÃO DA DISSERTAÇÃO

Além deste Capítulo introdutório, esta dissertação está organizada em mais 6 (seis) capítulos, conforme apresentados a seguir:

- Capítulo 2 (**REVISÃO CONTEXTUAL E DE LITERATURA**): onde são introduzidos os aspectos teóricos e conceituais utilizados ao longo da dissertação, além da identificação e estudo dos principais trabalhos relacionados existentes na literatura;
- Capítulo 3 (**ARCABOUÇO BASEADO EM CONTEXTO**): onde são apresentados conceitos e características do arcabouço CAARF;
- Capítulo 4 (**IMPLEMENTAÇÃO: GESTÃO DE ENCAMINHAMENTO**): onde são apresentados os principais resultados da contribuição desta dissertação de mestrado, relacionados à implementação de uma arquitetura para o arcabouço *Context-Aware Adaptive Routing Framework* (CAARF), mais especificamente do módulo funcional de Gestão de encaminhamento;

- Capítulo 5 (**ESTUDOS DE CASO**): são apresentados alguns estudos de casos 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 algumas conclusões acerca destes estudos;
- Capítulo 6 (**CONCLUSÕES E PERSPECTIVAS FUTURAS**): são discutidas as principais conclusões deste trabalho e de sua continuação através das perspectivas futuras.

2 REVISÃO CONTEXTUAL E DE LITERATURA

2.1 INTRODUÇÃO

Neste capítulo são abordados os principais conceitos relacionados com a temática do trabalho desenvolvido. Dentre os principais tópicos abordados neste capitulo estão Qualidade de Serviço (QoS), Qualidade de Experiência (QoE) e Qualidade de Dispositivo (QoD), além de uma breve introdução sobre roteamento adaptativo e engenharia de tráfego. Também são apresentados aspectos relacionados ao contexto, tanto aplicações baseadas em contexto, quanto às redes baseadas em contexto. Por fim, são discutidos alguns trabalhos relacionados de forma a proporcionar um posicionamento com o trabalho desenvolvido.

2.2 REVISÃO CONCEITUAL

Para que possamos compreender os conceitos utilizados nos principais tópicos desta dissertação e de forma a automatizar o entendimento dos termos abordados, esforços destinados a este propósito, são resumidos nesta seção. Os tópicos contemplados nesta seção são: *Qualidade de Serviço*, Roteamento Adaptativo, Qualidade de Experiência, Qualidade do Dispositivo, Engenharia de trafego, Contexto, **Erro! Fonte de referência não encontrada.**, **Erro! Fonte de referência não encontrada.** e por fim **Erro! Fonte de referência não encontrada.**.

2.2.1 Qualidade de Serviço

Com o desenvolvimento de aplicações de multimídia (voz, vídeo e dados), tornou-se necessário, além de técnicas de redução de congestionamento, o fornecimento por parte das redes, de parâmetros de desempenho com uma determinada qualidade, que permitirá assim uma comunicação fim-a-fim confiável, ou ainda a inviabilidade de sua utilização (LU, 1996).

Qualidade de Serviço (QoS) busca atender às expectativas do usuário em virtude do tempo de resposta e da qualidade, muitas vezes subjetiva, do serviço que está sendo provido, ou seja, fidelidade adequada do som e/ou da imagem sem ruídos nem congelamentos, neste caso em aplicações de tempo real.

A qualidade de serviço da rede depende das necessidades da aplicação, ou seja, do que ela requisita da rede a fim de que funcione bem e atenda, por sua vez, às necessidades do usuário. Estes requisitos são traduzidos em parâmetros indicadores do desempenho da rede como, por exemplo, o atraso máximo sofrido pelo tráfego da aplicação entre o computador origem e destino.

Podemos definir QoS como um conjunto de características quantitativas ou qualitativas (ITUT, 1993) (VOGEL, 1995). Estas características são chamadas de parâmetros de especificação da QoS. O conceito de Qualidade de Serviço serve para mensurar a qualidade dos serviços oferecidos por uma rede de comunicações, ou seja, refletir o quanto ela é capaz de atender às expectativas de seus usuários através dos serviços que oferece.

Esse conceito, inicialmente orientado à rede, evoluiu para uma noção mais ampla, contemplando as múltiplas camadas da interação usuário-sistema. QoS envolve aspectos relacionados à própria percepção do usuário, aos requisitos das aplicações e aos recursos disponíveis no sistema, seja no equipamento do usuário ou na rede em si. QoS tem relação ainda com questões organizacionais e administrativas, como segurança, privacidade, contabilidade, política de preços dos serviços, estabelecimento e monitoração de contratos de serviços e grau de disponibilidade da rede (em termos de tempo médio entre falhas e tempo médio de reparo).

QoS é o termo utilizado para identificar as diversas alternativas técnicas que permitem especializar as redes, garantindo que aplicações diferentes recebam tratamento diferente, ou seja, transmissões que exigem maior qualidade da rede, recebem maior prioridade.

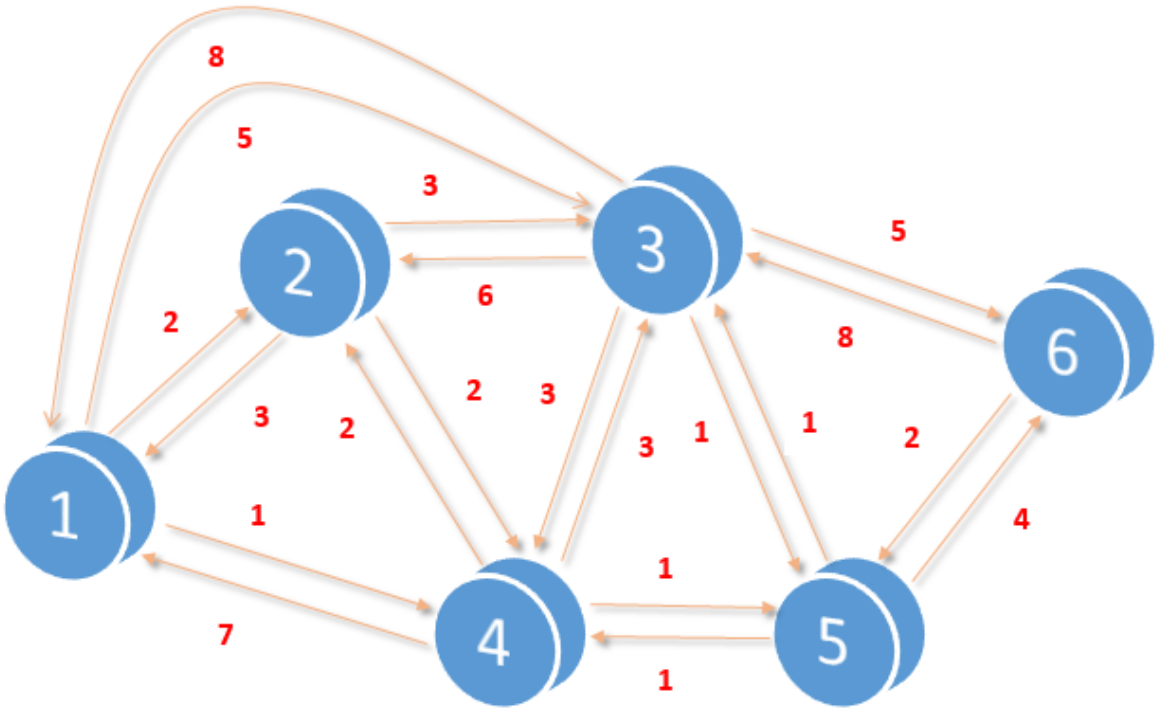
2.2.2 Roteamento Adaptativo

A principal função da camada de rede é encaminhar pacotes da máquina de origem até a máquina de destino. Nas sub-redes, em grande maioria, os pacotes necessitam de vários *nós* para completar o trajeto. Os algoritmos responsáveis por determinar as rotas e as estruturas de dados que eles utilizam constituem um dos elementos mais importantes do projeto da camada de rede (TANEMBAUM, 2003).

Conforme ilustra a **Erro! Fonte de referência não encontrada.**, para determinarmos a melhor rota, um algoritmo de roteamento levaria em consideração

critérios de desempenho como: número de saltos, menor custo do enlace (Menor número de saltos, menor custo), dentre outros.

Figura 3 - Exemplo de Roteamento



Um algoritmo de roteamento é a parte do *software* da camada de rede responsável pela decisão sobre a linha de saída a ser utilizada na transmissão do pacote de entrada. Se a sub-rede fizer uso de datagramas, a decisão deverá ser tomada mais uma vez para cada pacote de dados que é recebido, pois a rota mais indicada pode ter sido alterada desde o último pacote (TANEMBAUM, 2003)

Portanto, a finalidade de um algoritmo de roteamento é simples: dado um conjunto de roteadores conectados por enlaces, um algoritmo descobre um ‘bom’ caminho entre o roteador de fonte e o roteador de destino. Normalmente um ‘bom’ caminho é aquele que tem o ‘menor custo’ (KUROSE; ROSS, 2009)

As pesquisas apresentadas em Lee et al. (1994) e Glazer e Tropper (1990) rendem algumas tentativas de equilíbrio de carga atribuindo pesos aos enlaces levando em conta as condições do trafego local. Para esta atribuição de pesos denominamos roteamento adaptativo, ou roteamento sensível ao trafego.

No roteamento adaptativo alternativo ou, simplesmente, adaptativo (LINS et al, 2006), a seleção de uma rota do conjunto de rotas definido previamente é feita de acordo com informações do estado atual da rede.

2.2.3 Qualidade de Experiência

A Qualidade de Experiência (QoE) é uma métrica que representa o desempenho de um serviço do ponto de vista dos usuários, enfatizando a perspectiva pessoal de satisfação (SCHIMUNECK et al., 2014).

Em redes de computadores, a QoE reflete a percepção do usuário sobre a qualidade de um serviço, que é diretamente influenciada por métricas de Qualidade de Serviço (QoS) como largura de banda, perda de pacotes, etc. A análise dessas métricas de QoS pode indicar se a rede atende aos requisitos desejados pelos usuários, como, por exemplo, assistir a um filme em alta definição. Entretanto, métricas de QoS não permitem avaliar a experiência dos usuários sobre o desempenho da rede. Por exemplo, em um serviço de *Video Stream*, um erro de codificação pode causar a reprodução de um vídeo em baixa qualidade (SCHIMUNECK et al., 2014).

Ainda que as métricas de QoS indiquem uma boa qualidade da rede, o baixo nível de satisfação dos usuários poderia indicar a existência de um problema que não é perceptível através de parâmetros de QoS. Desse modo, para considerar a opinião dos usuários em relação ao serviço oferecido, outras métricas foram criadas e denominadas de QoE. As métricas de QoE referem-se à percepção do usuário, podendo ser expressa em níveis discretos, como por exemplo, excelente, bom, satisfatório, ruim e muito ruim, definidas pelo *Mean Opinion Score* (MOS) (SCHIMUNECK et al., 2014).

O MOS é o resultado da média de pontuação, em um intervalo de 1 a 5, no qual os clientes da rede atribuem uma nota referente ao desempenho do sistema de transmissão (SCHIMUNECK et al., 2014).

2.2.4 Qualidade do Dispositivo

Serviços são executados em componentes de hardware, estes dispositivos possuem também uma qualidade, chamada Qualidade de Dispositivos (QoD). QoD é

qualquer informação sobre as características técnicas e capacidades de um dispositivo (KIM ; LEE 2006).

Os dispositivos que usam a rede possuem características técnicas que os diferenciam dos outros, sejam recursos de processamento, memória, armazenamento secundário ou mesmo recursos como GPS habilitado, sensores, entre outros recursos que podem viabilizar ou não o uso de determinadas funções em aplicações. Dessa forma, determinados dispositivos podem ser viáveis serem usados em determinadas comunicações ou aplicações enquanto outros não possuem requisitos mínimos para usarem determinadas funções. Portanto, a Qualidade de Dispositivo permite aferir no fornecimento de informações referentes as características técnicas de cada dispositivo, e suas capacidades.

2.2.5 Engenharia de trafego

Engenharia de Tráfego é o processo de organizar como o tráfego flui através da rede para que congestionamentos causados pela utilização desigual da rede possam ser evitados. A Engenharia de Tráfego é direcionada à otimização de desempenho de redes operacionais. Em geral, ela engloba a aplicação de princípios tecnológicos e científicos para medir, modelar, caracterizar e controlar o tráfego na Internet e a aplicação dessas técnicas e conhecimentos para atingir determinados objetivos de desempenho (AWDUCHE, D. et al., 1999).

Um objetivo central é facilitar a operação eficiente e confiável da rede enquanto que ao mesmo tempo otimiza a sua utilização e desempenho. Engenharia de Tráfego é considerada uma função indispensável em grandes redes por conta do alto custo dos equipamentos e da natureza comercial e competitiva da Internet.

Como modifica o fluxo natural dos pacotes, podemos utiliza-la para atender a requisitos de QoS de determinados fluxos de dados.

Os objetivos de desempenho da Engenharia de Tráfego podem ser classificados como:

- **Orientados a recursos**. Estão relacionados à otimização dos recursos da rede, como impedir que certas partes da rede se tornem congestionadas, enquanto outras permaneçam com recursos ociosos;
- **Orientados a tráfego**: incluem os aspectos relacionados à manutenção das garantias de QoS dos fluxos de dados (ou agregações de fluxos).

A Engenharia de Tráfego pode ser feita manualmente, ou usando algum tipo de técnica automatizada, usando inclusive MPLS e/ou Roteamento com QoS para descobrir e fixar os caminhos mais adequados a determinadas agregações de fluxos dentro da rede.

2.2.6 Contexto

A computação ubíqua/pervasiva é um paradigma caracterizado pela presença de dispositivos móveis, que cada vez mais fazem parte do cotidiano das pessoas. Estes dispositivos possuem uma considerável capacidade de processamento, recursos de comunicação sem fio e armazenamento de dados. Possuem ainda, funcionalidades diversificadas e interfaces como GPS, rádio e TV, tocadores de áudio, câmeras digitais entre outros, sendo utilizados em aplicações de diversas áreas como: indústria, comércio, turismo, saúde, entretenimento.

Este tipo de computação possui forte ligação com as características do mundo físico, bem como aquelas apresentadas pelos perfis de seus usuários. Tais informações são chamadas de contextos e representam o elemento de entrada para a computação ciente ou sensível ao contexto (LUO REN et al. 2009).

O contexto é qualquer informação que possa ser utilizada para caracterizar a situação de entidades como: pessoa, lugar ou objeto, que sejam consideradas relevantes para interação entre um usuário e uma aplicação (DEY, 2000).

Diversas são as classificações de contexto encontradas na literatura como Chen e Kotz (2000), Viana et al. (2007), Emmanouilidis et al. (2012) e Schuster et al. (2012). Por exemplo, para Chen e Kotz (2000) o contexto apresenta quatro dimensões composta por: contexto computacional, contexto físico, contexto de tempo e contexto do usuário.

Baseado nestas informações contextuais, os serviços podem ser adequar as características do ambiente atual ou ainda do perfil do usuário, promovendo assim, mais otimização e personalização e automaticamente a satisfação do usuário.

A Qualidade de Contexto (QoC) é qualquer informação que descreve a qualidade da informação que é usada como informação de contexto (BUCHHOLZ et al. 2003).

A definição de QoC para Krause e Hochstatter (2005) é qualquer informação inerente que descreve informação de contexto e pode ser usada para determinar o

valor da informação para uma aplicação específica. Isso inclui informações sobre o processo de provisionamento que a informação foi submetida (“histórico”,” idade”), mas não tratam de estimativas sobre os passos de provisionamentos futuros.

Na próxima seção são apresentados alguns trabalhos relacionados encontrados na literatura.

2.3 TRABALHOS RELACIONADOS

A pesquisa realizada contemplou as contribuições relacionadas ao Roteamento Adaptativo baseado em QoS, QoD, QoE, QoC, Engenharia de Trafego, Contexto, entre outros. A seguir são apresentados os trabalhos identificados nas principais áreas de estudo.

2.3.1 QoS Management Framework

Qualidade de Serviço (QoS) foi concebida como uma solução a fim de proporcionar diferentes garantias de tráfego para diferentes aplicações, usuários ou fluxos de dados. Algumas das métricas aplicadas para implementar QoS são taxa de bits, atraso, jitter, entre outros. Essas garantias são importantes se a capacidade da rede é insuficiente, principalmente para aplicações de streaming multimídia em tempo real, como voz sobre IP, jogos online e IP-TV, uma vez que estes necessitam de taxas de bits fixos e são sensíveis a atrasos, um outro caso são redes onde a capacidade é um recurso limitado, por exemplo em comunicação de dados celular.

Vários paradigmas de QoS têm sido propostos, tais como os Serviços Integrados que oferecem garantias para os fluxos individuais (BRADEN et al, 1994), os Serviços Diferenciados, que implementam classes de serviços em roteadores para os tráfegos agregados (BLACK, 1998), o *Multi Protocol Label Switching (MPLS)* que oferece rotas específicas para a entrega de fluxos e melhor utilização dos recursos de largura de banda (ROSEN et al, 1999), e ainda *Generalized Multi-Protocol Label Switching (GMPLS)*, uma extensão do MPLS para suportar redes ópticas etc (SHAIKH et al., 2010).

Com base na experiência adquirida com essas contribuições, diferentes *frameworks* para a gestão de QoS foram propostas a fim de resolver questões específicas sempre propondo uma dissociação entre o plano de controle de rede a

partir do plano de dados e apresentação de um modelo subjacente desenvolvido como um framework QoS. O desenho de alguns dos frameworks de QoS existentes na literatura seguiu um padrão relacionado com a definição de quatro módulos principais:

- **Gestão de rede/recursos do aplicativo** - Responsável pela negociação de Acordo de Nível de Serviço (SLA) com os clientes e comunica os parâmetros associados a um SLA com o respectivo gerente de recursos, além de garantir SLAs conforme os recursos alocados (HONG et al. 2003), (KUSMIEREK et al. 2003), (AL-ALI et al. 2002), (YERIMA. 2011), (DEORA et al. 2011), (AGBOMA et al. 2008) e (CHOWDHURY et al. 2002);
- **Gestão de contexto e adaptação** - Considera-se na concepção da estrutura do modelo, a fim de permitir a gestão de recursos de contexto conscientes (HONG et al. 2003), (AL-ALI et al. 2002), (YERIMA. 2011), (DEORA et al. 2011), (AGBOMA et al. 2008) e (CHOWDHURY et al. 2002);
- **Monitoramento e medição** - Alimenta o banco de dados de gestores de recursos de rede, monitoramento os dados adquiridos no âmbito da rede (HONG et al. 2003), (AL-ALI et al. 2002), (YERIMA. 2011) (AGBOMA et al. 2008) e (CHOWDHURY et al. 2002) e;
- **Gestão de QoS para cada aplicativo** - Responsável pela especificação e negociação de requisitos de QoS de uma aplicação específica (HONG et al. 2003), (KUSMIEREK et al. 2003), (AL-ALI et al. 2002), (YERIMA. 2011) e (CHOWDHURY et al. 2002).

Na maioria dessas contribuições os autores aplicaram o mesmo conjunto de métricas, a fim de propor os seus respectivos frameworks de QoS, como latência, taxa de transferência, falhas e custo. Outras métricas mais específicas também são aplicadas, tais como disponibilidade, segurança, acessibilidade e regularidade.

Para além dos esforços existentes, a fim de proporcionar prioridade e disponibilidade de recursos, é também importante prever meios de otimizar os caminhos de encaminhamento existentes de acordo com o estado da rede. Portanto, também podemos considerar diferentes contribuições na literatura, a fim de fornecer roteamento adaptativo.

2.3.2 Roteamento Adaptativo

Com a dinâmica de utilização das redes IP atuais, o roteamento adaptativo tem sido um meio relativamente eficiente, estudado pelos pesquisadores, para permitir que vários tipos de tráfegos na rede convivam o mais harmoniosamente possível com a infraestrutura existente.

A evolução das aplicações e a crescente disseminação de novas aplicações nas redes de computadores tem sido um importante limitador da qualidade das redes, dada sua incrível massificação nos últimos anos. Inversamente a esse crescimento, a infraestrutura tem estado atrás nessa evolução o que tem causado, constantemente, grande congestionamento nas redes de computadores, principalmente, na internet.

Diante do exposto, muitos estudos sendo feitos estão relacionados a roteamento dinâmico ou roteamento adaptativo, que procura capacitar a rede na adaptação à dinâmica de funcionamento com aumento de tráfego inesperado, por exemplo. A engenharia de tráfego tem sido uma abordagem bastante utilizado como um mecanismo importante na resolução dessa equação complicada entre qualidade da aplicação e a quantidade da banda utilizada para ela. A otimização do uso da banda também é um ponto importante sendo estudado, que é um dos focos principais da engenharia de tráfego com a otimização do uso dos recursos na rede. O estudo dos múltiplos caminhos, possibilitando múltiplas alternativas para os nós, tem sido, também, um importante ponto. Outra abordagem sendo feita é o controle do congestionamento, que viabiliza uma melhor qualidade do canal para todas as aplicações que nele precisam trafegar.

Todas essas abordagens têm sido bastante utilizadas nos estudos sobre roteamento adaptativo com algumas pequenas variações a depender do algoritmo ou do modelo que venham a utilizar. Portanto, podem ser classificados de forma genérica os roteamentos adaptativos da seguinte forma:

- Baseados em contexto (MUSOLESI et al, 2005), (WENNING et al, 2009, (PETZ et al, 2012);
- Baseados em engenharia de tráfego (KANDULA, 2005), (FISCHER, 2006), (KARTHIGA, 2012) e;
- Baseados em dados probabilísticos (XIE, 2005).

A abordagem do roteamento adaptativo baseado em dados de contextos baseou-se em métricas distintas das usadas em qualidade de serviço, como atraso, latência entre outras. As métricas usadas pelos algoritmos ou pelas soluções usadas pelos protocolos baseados em contexto foram as seguintes: taxa de mudança de conectividade de um host, Nível de energia, probabilidade de entrega, Potência do Sinal, Saúde do Nó, Contador de Saltos e taxa de aprendizado.

Outra abordagem utilizada é a do roteamento adaptativo com soluções baseadas em engenharia de tráfego. Nas soluções baseadas nessa abordagem, a engenharia de tráfego é utilizada, principalmente, na otimização do uso dos recursos de banda.

Para a otimização de roteamento é importante considerar Qualidade de Serviço (QoS) e Qualidade de Experiência (QoE). Como esperado, as contribuições relacionadas com estruturas de QoS se concentram sobre a proposta de protocolos e mecanismos a fim otimizar a disponibilidade de recursos relacionados a equipamentos de rede. QoE, em vez disso, refere-se as expectativas dos usuários e como eles realmente percebem o serviço prestado. A fim de satisfazer as expectativas dos usuários, a implementação de QoS deve ser centrado sobre as perspectivas dos usuários finais, para garantir que a qualidade do serviço atenda aos níveis de QoE esperados.

2.3.3 Qualidade de Experiência

As estratégias de configuração, provisionamento e planejamento de serviços de redes por parte dos provedores de serviço, em geral, implementam as técnicas de QoS mais conhecidas, descritas no capitulo 2.2.1, a fim de garantir a qualidade de serviço aos clientes/usuários. No entanto, o pressuposto de que aplicação diligente de técnicas de QoS a fim de preservar aplicações mais sensíveis na modelagem do tráfego obterá o resultado pretendido, levando em conta somente o ponto de vista do Provedor de Serviços, tem se revelado insuficiente. Em muitas situações, a despeito das técnicas aplicadas, a qualidade esperada não é atingida na borda, gerando insatisfação do consumidor do serviço (SHAIKH et al., 2010).

Esta questão tem se tornado cada vez mais crítica na medida em que as aplicações tradicionais de rede, que compreendem transferência de arquivos, troca de mensagens, comunicação através das redes sociais, vídeos recreativos, como o

Youtube (YOUTUBE, 2015), e-mail e suas formas correlatas, acesso a sites, e-commerce, serviços financeiros e de consultas, somam-se novas aplicações multimídia e serviços que envolvem voz e vídeo interativo, requerendo um gerenciamento de expectativas de usuário muito mais complexo de administrar.

Ao universo de aplicações de voz e vídeo sensíveis ao QoS, anteriormente restritas ao VoIP e videoconferência, vêm sendo acrescidas aplicações que envolvem um grau de interatividade crescente, tais como jogos interativos – muitos dos quais requerem sincronia entre os jogadores, e outras formas de interação multimídia, notadamente videoconferência em escala mais disseminada, *e-learning* por meio de ferramentas de *webminar*, conferências online, vídeo-sob-demanda, e a categoria denominada “tele presença”.

A verificação da satisfação dos usuários destas aplicações sob a ótica do cliente representa um desafio ampliado em termos de medição, requerendo a definição de métricas e aferição que não podem ser obtidas diretamente a partir dos parâmetros clássicos de QoS somente. 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 definidos são uma garantia absoluta de aprovação da qualidade de experiência do usuário.

Qualidade de Experiência (QoE) trata portanto da quantificação da percepção do subjetivo, mudando o ponto de vista da aferição de qualidade da rede para o usuário. Portanto, é “*user-centric*” em contraposição ao “*network-centric*”, envolvendo fatores cognitivos, comportamentais e psicológicos (MITRA et al., 2011), que passa a estudar 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 fatores 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 de usuário a partir de métricas objetivas. Esta é uma característica que embasa a ação proativa em relação à percepção do usuário e não somente a percepção do bom funcionamento da rede, que não necessariamente traduz a “aferição subjetiva do usuário”.

Em outras palavras, o problema com o qual os provedores se defrontam é que não necessariamente boas práticas de QoS ou *Traffic Engineering* resultam em

satisfação do usuário, isto é, o QoE, o que pode levar à perda inesperada de clientes, ainda que os parâmetros de QoS aparentemente estejam bem ajustados.

De maneira geral, o QoS é "*network oriented*" e o QoE é "*user oriented*" (ALRESHOODI et at., 2013). O QoE pode ser entendido também como uma cobertura de quatro domínios: contextual, tecnológico, negócios e humano (LAGHARI et al., 2012), entendido como uma pseudo-camada entre a aplicação e a rede (ALRESHOODI et at., 2013).

Em geral, QoE pode ser correlacionada pela medição da MOS (*Mean Opinion Score*) cujos valores variam entre má experiência, pobre, aceitável, bom e excelente (SKORIN-KAPOV et al., 2012). Esta é uma medida subjetiva que pode ser obtida por meio de uma pesquisa de satisfação ou por inferência recolhidas a partir da correlação com outros parâmetros.

As contribuições existentes na literatura visam mapeamento de parâmetros de QoS e outros parâmetros existentes relacionados à análise e avaliação da qualidade do vídeo para o QoE correspondente por meio de fórmulas matemáticas e parâmetros já existentes para a avaliação da qualidade de vídeo como proposto em (HOßFELD et al., 2007). Algumas contribuições consideram que a natureza da percepção QoE é baseado na correlação entre QoS e QoE, envolvendo parâmetros existentes, que já são aplicados para a medição de aplicações multimídia.

Além das contribuições existentes que visam medir parâmetros objetivos e mais mapeando-os em métricas de QoE, outras contribuições também coletar informações diretamente de usuários usando técnicas como *crowdsourcing* (CHEN et al., 2010).

Apesar de não haver uma única abordagem dominante estabelecida, algumas estratégias adotam métricas obtidas a partir de parâmetros de qualidade de vídeo, permitindo experiências medições eficazes, a fim de correlacionar QoE e MOS. As correlações que são principalmente aplicadas são PESQ/MOS e MOS/QOE (HOßFELD et al., 2007).

Esta seção delineou algumas das contribuições existentes na literatura a fim de correlacionar Qualidade da Experiência com os familiares MOS (*Mean Opinion Score*). Além das métricas já conhecidas de QoS, como *packet loss, packet reordering, jitter, delay*, entre outras, devemos acrescer as métricas pertinentes às aplicações multimídia, é também importante ter em conta os dados contextuais, que

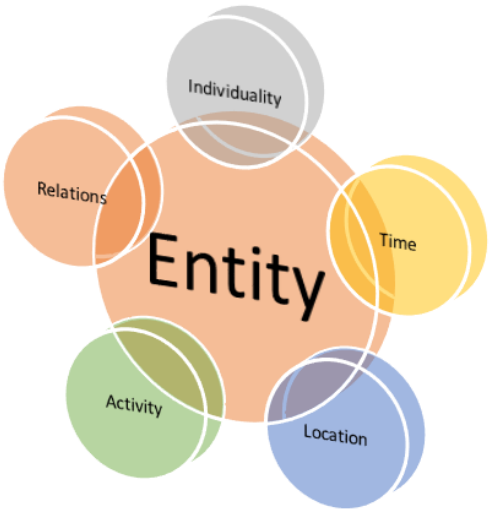
podem ser aplicadas para melhorar o fornecimento de serviços e maximizando assim a experiência do utilizador.

2.3.4 Contexto

Contexto pode ser definido como

> qualquer informação que possa ser usada para caracterizar a situação das entidades (ou seja, se uma pessoa, objeto ou local) que são considerados relevantes para a interação entre um usuário e uma aplicação, incluindo o próprio utilizador e a aplicação. Contexto geralmente está relacionado à localização, identidade e estado de pessoas, grupos e objetos computacionais e físicos. (DEY et at., 2001).

Figura 4 - Cinco características fundamentais para informações de contexto



Fonte: Zimmermann et al. (2007).

O uso de contexto pode ser aplicado a entidades, pessoas, lugares, ou mesmo a um objeto relevante para a aplicação, através da definição das características como individualidade, atividade, localização e de tempo e até mesmo relações com outras entidades (ZIMMERMANN et al., 2007), tal como representado na **Erro! Fonte de referência não encontrada.**.

A definição de contexto também está relacionada com a descrição dos elementos contextuais (ECs). Enquanto contexto refere-se à interação de um agente e uma aplicação, um EC caracteriza o domínio do contexto que este agente está inserido (VIEIRA et al., 2009). Assim, as aplicações sensíveis ao contexto são

capazes de fornecer serviços com tarefas baseadas em ações de assistência, baseada em contexto e adaptação do comportamento do sistema de acordo com a informação contextual (VIEIRA et al., 2009), (NAZARIO et al., 2012). Esses serviços são chamados serviços sensíveis ao contexto (WEISER, 1999).

A fim de avaliar as informações descritas pelo contexto, e o dispositivo que será a fonte desta informação, o conceito de Qualidade de contexto e da qualidade de dispositivos são considerados.

2.3.5 Qualidade de Contexto

Embora a contribuição dos sistemas cientes de contexto pode ser expressiva, a sua eficácia só pode ser alcançada se as informações de contexto estiverem devidamente definidas. Portanto, a definição de Qualidade de Contexto (QoC) é necessária a fim de proporcionar o entendimento da correlação entre QoC, QoS e qualidade do dispositivo (QoD). Este último está relacionado com os componentes de hardware envolvidas no fornecimento da informação de contexto (BUCHHOLZ et al., 2003), (NAZARIO et al., 2012).

A fim de proporcionar algumas métricas para QoC (BUCHHOLZ et al., 2003) são definidos: Precisão da informação (*Precision*); Probabilidade das informações estarem corretas *(Probability of Correctness*); Nível de confiança nas fontes de informação (*Trust-worthiness*); Resolução dos níveis de granularidade de informações (*Resolution*); Atualização das informações relativas às suas características temporais (*Up-to-dateness*).

No entanto, é importante a adoção de políticas claras, a fim de proporcionar a correta análise de informações contextuais e estar em conformidade com QoC (MANZOOR et al. 2009). Vale mencionar que existem trabalhos que buscam propostas para melhorar a adoção de políticas para QoC baseadas não apenas no contexto atual, mas também nos efeitos da informação errônea de informações de contexto com baixa qualidade e seus efeitos para os sistemas, tentando minimizar a adoção sobrecarga de políticas ineficientes, como por exemplo o Proteus (AGBOMA et al., 2008).

2.3.6 Qualidade do Dispositivo

Quanto ao fornecimento de informações de contexto, as diferenças, além de relacionadas aos acordos de QoC estabelecidos, estarão limitadas também a QoD existente, onde, por exemplo, o sistema de posicionamento global (*Global Positioning System – GPS*) de cada dispositivo podem ter níveis de precisão diferentes, ou mesmo um dispositivo pode não fornecer alguns parâmetros em comparação a outro pelo simples fato de não ter tal *hardware* ou a capacidade de coletar tais informações (BUCHHOLZ et al, 2003). Sendo assim, a QoD vai fornecer informações referentes as características técnicas de cada dispositivo, e suas capacidades (NAZARIO et al., 2012).

Portanto, QoD irá fornecer informações sobre as características técnicas de cada dispositivo e as suas capacidades (VIEIRA et al, 2009). Um estudo comparativo de algumas contribuições relacionadas com roteamento ciente de contexto foi realizado em (PETZ et al, 2012), (YASAR et al, 2010), (WENNING et al ; MASCOLO et al (2006). A maioria dessas contribuições estão relacionadas com a utilização do contexto aplicado principalmente às redes sem fio. Nestes estudos, o uso de contexto permitiu melhoramentos, principalmente em: estabilidade da ligação de comunicação, o aumento da largura de banda (por diminuir em cima), maior autonomia baterias, atraso mais curto e escalabilidade. Esses ambientes diferem muito de redes cabeadas, principalmente devido à capacidade de armazenamento e processamento de restrições, a limitação a vida da bateria e, em alguns casos, a largura de banda limitada.

2.4 CONCLUSÃO

Neste capítulo foi realizada uma abordagem teórica dos principais aspectos que envolvem o desenvolvimento deste trabalho, com destaque para Qualidade de Experiência, Qualidade de Serviço, Roteamento Adaptativo e Contexto. Também foram apresentadas algumas ferramentas utilizadas para o entendimento e conclusão da implementação proposta.

Para melhor fundamentar o desenvolvimento deste trabalho, foram levantados aspectos relacionados às soluções que tenham como base algum modelo de contexto estabelecido, além da apresentação de algumas ferramentas usadas para automatizar

a validação da solução proposta, bem como suas principais características. Em resumo os assuntos abordados neste capítulo foram fundamentais para contextualizar e dar suporte à proposta de trabalho realizada nesta dissertação.

3 ARCABOUÇO BASEADO EM CONTEXTO

Este capítulo visa apresentar o arcabouço utilizado, bem como o modelo de contexto adjacente. As Seções 3.1 apresenta uma introdução sobre o capitulo. A seção 3.2 apresenta o modelo de contexto e suas características. A Seção 3.3 detalha o arcabouço CAARF, discutindo o seu modelo funcional. A Seção 3.4 trata da abordagem operacional do CAARF. A seção 3.5 são apresentados os submódulos do componente *Context-Based Forwarding Management*, bem como alguns aspectos funcionais necessários para sua implementação. A seção 3.6 apresenta a gestão de encaminhamento relacionadas a abordagem deste trabalho. E por último, na seção **Erro! Fonte de referência não encontrada.** é apresentada a conclusão do capitulo.

3.1 INTRODUÇÃO

Existem inúmeras definições para arcabouço na literatura. Para Johnson (1997) e Gamma et al (1995), um arcabouço é um conjunto de objetos que colaboram com o objetivo de atender a um conjunto de responsabilidades para uma aplicação especifica ou um domínio de aplicação. Já para Mattsson (2000), um arcabouço é uma estrutura desenvolvida com o objetivo de atingir a máxima reutilização, representada como um conjunto de classes abstratas e concretas, com grande potencial de especialização.

Este capítulo apresenta uma visão geral do modelo e do arcabouço adotados no âmbito da nossa contribuição para apoiar as tomadas de decisões de encaminhamento de dados baseadas em contexto, além de serem pré-requisitos para realização dos experimentos e simulações, denominados *Context Model* e *Context-Aware Adaptative Routing Framework (CAARF)*.

O modelo de contexto utilizado neste trabalho de Oliveira (2015) é aplicado para a implementação de uma solução de roteamento adaptativo baseado em informações contextuais. Esse modelo é genérico permitindo vários cenários de rede e, além disso, introduz a adaptação baseada na experiência do usuário.

O CAARF apresenta uma visão do contexto da rede, sem desprezar as informações comumente usadas para Qualidade de Serviço (QoS), tais como largura de banda, latência, taxa de erros e variação no atraso, procurando focar na

experiência do usuário, o que a difere da maioria das arquiteturas, atualmente, utilizadas.

Este arcabouço apresenta uma nova implementação no tratamento das questões de tráfego dentro de uma rede, pois permite avaliar não somente os parâmetros usualmente conhecidos como largura de banda, latência, variação no atraso e taxa de erros, mas também parâmetros como capacidade de processamento momentâneo de um dispositivo ou mesmo a capacidade de memória momentânea de um dispositivo. Esse novo formato de interpretação das respostas, que a rede permite captar, faz com que a realidade da rede seja conhecida com um pouco mais de precisão e dessa forma permitir uma resposta mais real e precisa em relação ao contexto de avaliação como um todo.

O arcabouço CAARF é detalhado apresentando suas características, as interações entre os submódulos internos, seus papeis, as regras estabelecidas, o tratamento dos eventos e a interação da arquitetura com o domínio de rede onde está inserida.

3.2 MODELO DE CONTEXTO

Antes de ser descrito o arcabouço baseado em contexto para roteamento adaptativo, faz-se necessária uma descrição do modelo de contexto na qual o arcabouço utilizado é baseado e que norteia a definição das funcionalidades, características e funcionamento do CAARF.

O modelo de contexto utilizado nesse trabalho é aplicado para uma solução de roteamento adaptativo baseado em informações contextuais. Esse modelo é genérico permitindo vários cenários de rede e, além disso, introduz a adaptação baseada na experiência do usuário. O modelo ainda consegue descrever o estado de uma entidade particular, que pode ser um dispositivo, um roteador, um comutador, um usuário, uma aplicação, etc.

Seguem algumas informações relacionadas a entidade descrita no modelo (**Erro! Fonte de referência não encontrada.**). Essas características são agrupadas da seguinte forma:

- **Individualidade** – Descreve informações particulares sobre a entidade, tais como identificação, endereçamento, protocolos, entre outras informações;

- **Tempo** – Descreve informações relacionadas ao tempo tais como última atualização relacionada à entidade;
- **Localização** – Descreve aspectos relacionados à localização real ou virtual da entidade tais como localização baseada no geoposicionamento, de sua casa, edifício, escritório, endereço da rede, entre outros;
- **Atividade** – Descreve metas, tarefas, ações executadas pela entidade;
- **Relações** – Descreve o relacionamento dessa entidade com outras entidades, dentro e fora do domínio de rede, suas dependências, conexões com objetos, pessoas, serviços, entre outros.

Além dessas características descritas, o modelo de contexto é enriquecido com outros aspectos importantes, tais como:

- **Qualidade de Experiência (QoE)** – Esse aspecto descreve um grupo de parâmetros relacionados às experiências e perspectivas do usuário, como MoS (*Means Opinion Score*);
- **Qualidade de Dispositivo (QoD)** – Esse aspecto descreve um grupo de parâmetros relacionados às características dos dispositivos tais como capacidade de armazenamento, poder computacional, nível de precisão dos dados coletados, entre outros;
- **Qualidade de Serviço (QoS)** – Esse aspecto descreve um grupo de parâmetros relacionados às métricas, tanto qualitativas quanto quantitativas, considerando o *Service Level Agreement* (SLA) entre o usuário e a plataforma. Algumas das métricas consideradas são a largura de banda, latência e variação do atraso.

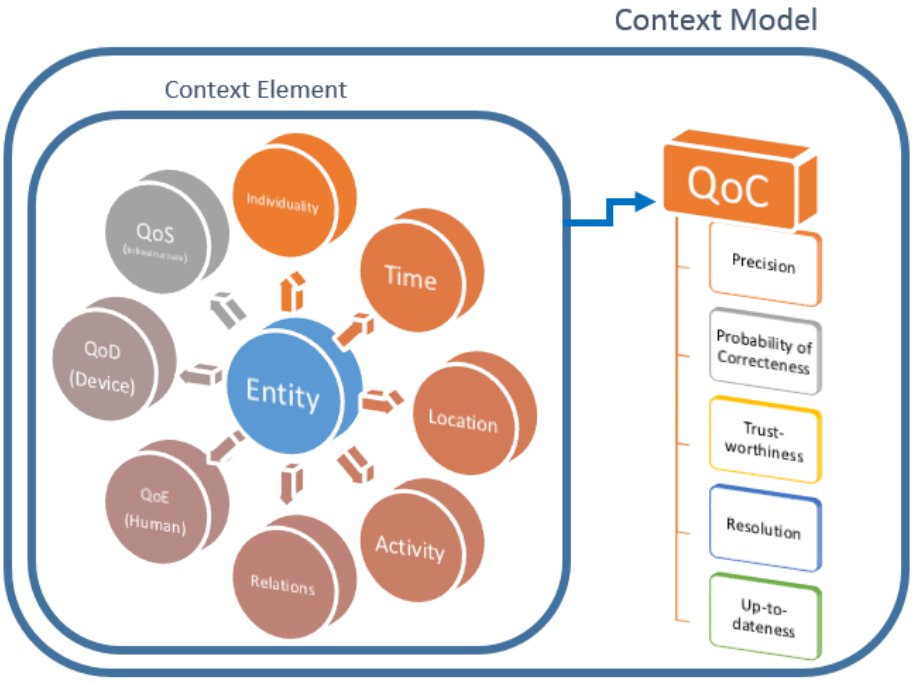
As informações relacionadas às características e aspectos, anteriormente, citados são validadas baseadas em algumas métricas, voltadas a qualidade de contexto (QoC). Essas métricas são:

- **Precisão** – Está relacionado ao nível de precisão das informações que permitem avaliar sua relevância;
- **Probabilidade de correção** – Está relacionada à probabilidade da informação estar correta;

- **Confiabilidade** – Está relacionado ao nível de confiabilidade sobre a fonte da informação em questão;
- **Resolução** – Está relacionado ao nível de granularidade de uma determinada informação;
- **Capacidade de Atualização** – Está relacionado à forma pela qual aquela informação fornecida é atualizada.

É importante salientar que a qualidade de contexto não é uma característica ou mesmo um aspecto de informação relacionado à entidade e sim um conjunto de restrições que devem ser aplicadas às informações das entidades no que diz respeito às métricas descritas acima.

Figura 5 - Modelo de Contexto



Baseado no modelo de contexto é possível introduzir o arcabouço CAARF.

3.3 CONTEXT-AWARE ADAPTIVE ROUTING FRAMEWOK (CAARF)

Após introduzir o modelo de contexto juntamente com seus aspectos e características, é possível apresentar o arcabouço CAARF que descreve tais funcionalidades de forma explícita mostrando a relação das entidades dentro de um

domínio de rede com o seu contexto. Esse arcabouço, como já citado, foi concebido com base na descrição de dois grupos funcionais principais:

- ***Context Management Modules*** – Tem a responsabilidade de analisar e filtrar as informações contextuais;
- ***Forwarding Management Modules*** – Tem a responsabilidade de processar as informações contextuais e aplicar as regras de encaminhamento na camada de encaminhamento da rede.

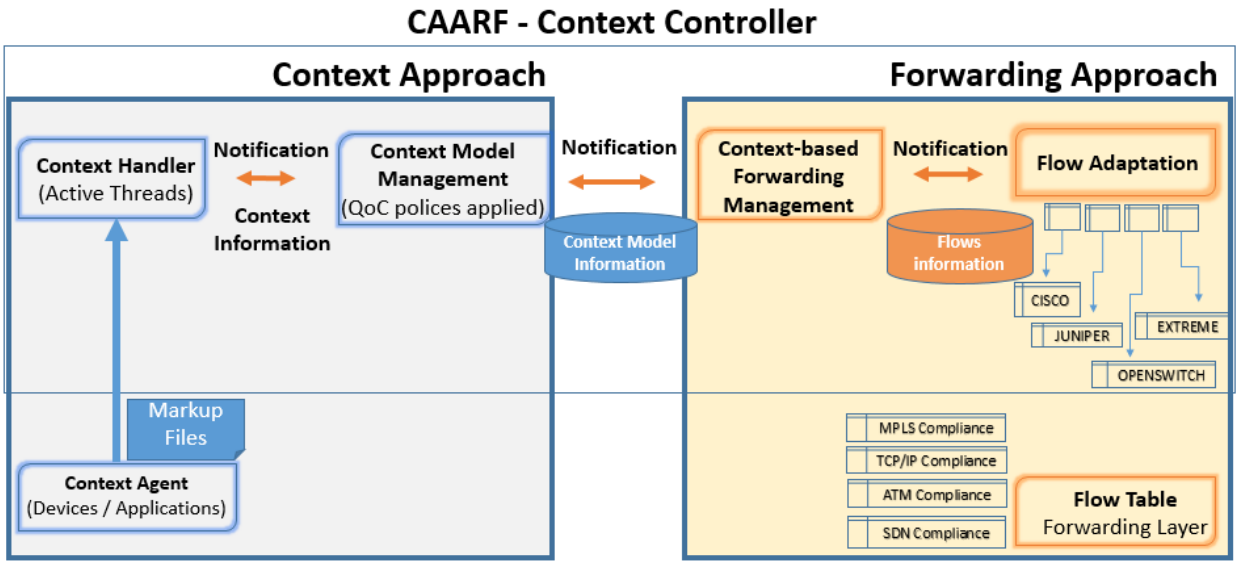
A nomenclatura dos módulos de gestão de contexto (*Context management module*) e de gestão de encaminhamento (*Forwarding management module*), assim como dos seus respectivos submódulos, foi definida na língua inglesa por questões de generalização e utilização nas publicações científicas. Por essa razão, esses termos são utilizados em inglês nesta dissertação.

Seguem algumas funcionalidades associadas a este arcabouço:

- Coletar e compartilhar informações contextuais;
- Centralizar o armazenamento de informações contextuais;
- Realizar validação e avaliação da informação contextual baseada nas políticas definidas pela QoC (Qualidade de Contexto), e;
- Suportar consulta e utilização da informação contextual a fim de atualizar o encaminhamento para serviços sensíveis ao contexto.

O arcabouço em questão chama-se CAARF (*Context-Aware Adaptive Routing Framework*) e foi baseado e concebido com a integração de diferentes funcionalidades e dividido em módulos, conforme ilustrado na **Erro! Fonte de referência não encontrada.**.

Figura 6 - Arcabouço CAARF



O primeiro grupo funcional apresentado na **Erro! Fonte de referência não encontrada.** está relacionado com os módulos que coletam, armazenam e tratam as informações contextuais. Esses submódulos são os seguintes:

- ***Context Agent*** – Responsável por receber as informações contextuais dos dispositivos de comunicação ativos, aplicações, usuários e suas relações. Essas informações são repassadas para o *Context Handler*;
- ***Context Handler*** – Responsável por receber as informações contextuais do Context Agent e armazená-las no *Context Database*;
- ***Context Model Management*** – Responsável por processar as regras de qualidade de contexto (QoC), definir o novo contexto de rede com as informações tratadas, e tornar essas informações disponíveis para serem acessadas pelo *Context-Based Forwarding Management*.

O segundo grupo funcional apresentado na **Erro! Fonte de referência não encontrada.** está relacionado com os módulos que analisam o contexto definido nos módulos de contexto e estabelecem ou não novos caminhos a serem aplicados aos fluxos a serem usados pelas relações ativas na rede.

Além de definir novos caminhos para os fluxos, estes podem ser aplicados nas tabelas de fluxos da camada *Forwarding Layer*. Esses sub-módulos são os seguintes:

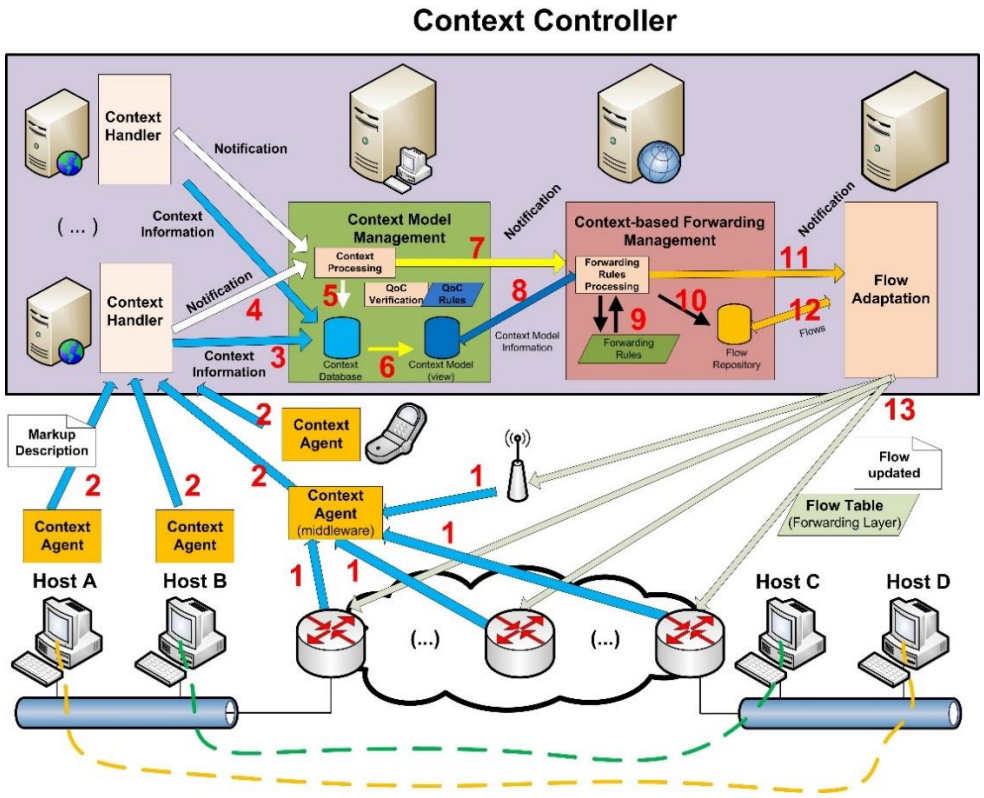
- ***Context-Based Forwarding Management*** – Responsável por processar as regras de encaminhamento baseadas em informações contextuais;
- ***Flow Adaptation*** – Responsável pela notificação de informação de encaminhamento atualizada nos respectivos dispositivos de comunicação ativos como *switches* ou roteadores, e;
- ***Flow Table*** – São as tabelas de fluxo da camada de encaminhamento (comutadores / roteadores).

Após introduzir os módulos funcionais do arcabouço CAARF, a sua abordagem operacional pode ser apresentada de forma a proporcionar uma melhor compreensão do seu funcionamento.

3.4 CAARF: ABORDAGEM OPERACIONAL

Nesta seção o funcionamento passo a passo do *Context-Aware Adaptive Routing Framewok* (CAARF) é discutido, conforme ilustrado na **Erro! Fonte de referência não encontrada.**.

Figura 7 - CAARF - Abordagem Operacional



1. O ***Context Agent/Middleware*** é responsável por coletar informações contextuais dos comutadores e estações de trabalho (hosts), além de parâmetros de QoD e QoS;
2. Num segundo momento, o ***Context Agent*** e o ***Context Agent Middleware*** enviam as informações contextuais para o ***Context Handler*****.**
3. O ***Buffer Management***, que atua dentro do ***Context Handler***, registra as informações recebidas do ***Context Agent*** e ***Context Agent Middleware*** dentro do ***Context Database***;
4. Um outro submódulo dentro do ***Context Handler*** chamado ***Notification Scheduling***, notifica a chegada de novas informações de contexto para o ***Context Model Management***;
5. Agora já dentro do ***Context Model Management*** é acionado o ***Context Processing*****,** que por sua vez tem o papel de receber as notificações de novas informações de contexto, e a partir daí, recupera informações de contexto do ***Context Database*** e as submete ao submódulo **QoC *Verification***, que irá aplicar as regras de QoC pré-definidas pelo Administrador da rede;

6. Em seguida o ***Context Processing***, registra o novo contexto no ***Context Model View***;
7. O ***Context Processing*** também notifica a modificação do contexto para o módulo ***Context-based Forwarding Management***;
8. ***Forwarding Rules Processing***, submódulo ***do Context-Based Forwarding Management***, recebe as notificações de novo contexto e a consulta no ***Context Model View***;
9. Após essa consulta, o ***Forwarding Rules Verification*** aplica as regras de encaminhamento pré-definidas pelo administrador da rede;
10. ***Forwarding Rules Processing***, submódulo do ***Context-Based Forwarding Management***, registra os fluxos atualizados no ***Flow Repository***;
11. Notifica o ***Flow Adaptation*** da atualização de novas regras de fluxo a serem aplicadas nos comutadores;
12. ***Flow Adaptation*** consulta as novas regras de fluxo no ***Current Flow View***;
13. E por fim aplica as novas regras de fluxo contidas no ***Current Flow View*** nos comutadores da rede.

Na próxima seção o módulo *Context-Based Forwarding Management* é discutido mais detalhadamente, uma vez que o objetivo deste trabalho é a proposta de uma abordagem para a implementação deste módulo.

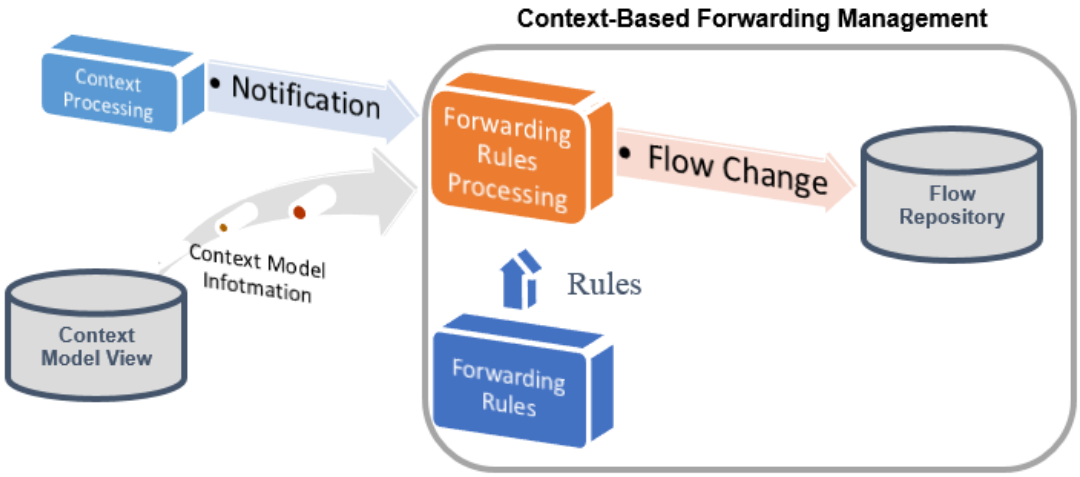
3.5 CONTEXT-BASED FORWARDING MANAGEMENT

Nesta sessão são apresentados os submódulos do componente *Context-Based Forwarding Management*, e alguns aspectos funcionais necessários para a sua implementação. Esse módulo possui dois submódulo principais, o *Context-Based Forwarding Management* e *Flow Adaptation*.

3.5.1 Módulo *Context-Based Forwarding Management*

O módulo *Context-based Forwarding Management* é dividido em quatro sub módulos, conforme ilustrado na **Erro! Fonte de referência não encontrada.**.

Figura 8 - Módulo *Context-Based Forwarding Management*



Os submódulos de *Context-Based Forwarding Management* são:

- ***Forwarding Rules Processing:*** Responsável por processar as regras de encaminhamento, pré-definidas pelo administrador de rede, no tratamento dos fluxos. Recebe a notificação do submódulo *Context Processing* e consulta as informações no *Context Model Database*;
    - Após criação das novas regras, compara com as regras atualmente em uso nos dispositivos e avalia se o índice de modificação foi maior que o mínimo pré-determinado para atualização dos dispositivos, se sim, autoriza transmissão;
- ***Forwarding Rules***: Responsável por armazenar as regras de encaminhamento que serão usadas no processamento de regras de encaminhamento para a criação ou não de novos fluxos nos comutadores;
- ***Flow Repository***: Responsável por armazenar os fluxos que serão encaminhados pelo submódulo *Flow Adaptation* para atualização das tabelas de fluxos dos comutadores. Armazena não só regras em uso nos dispositivos da rede, como também as novas regras criadas, mesmo que não sejam consideradas aptas para aplicação no novo cenário da rede.

Alguns dos eventos relacionados a este componente são apresentados a seguir (Uma relação completa dos eventos definidos nesse componente é apresentada no **Anexo E** – **Eventos do CAARF**):

- Eventos de Recebimento;

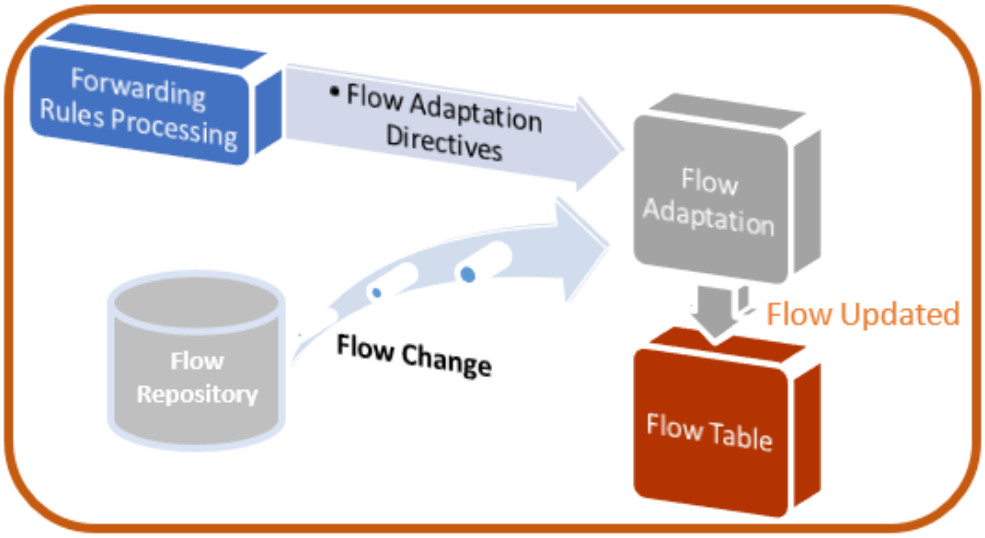
- Recebimento da inclusão de novas notificações a serem tratadas;

- Eventos de Notificação;
  - Notificação de novos fluxos para o *Flow Adaptation*;
- Outros Eventos
  - Leitura das informações de contexto atualizadas no *Context Model Database*;
  - Processamento das regras de fluxo pré-definidas;
  - Salva das novas regras de encaminhamento no *Flow Repository*.

3.5.2 Módulo *Flow Adaptation*

Flow Adaptation é responsável por receber as notificações de novos caminhos para os fluxos a serem aplicados ou mesmo atualizações de caminhos já existentes feitas pelo *Context-Based Forwarding Management* e, baseado no *Flow Repository*, por atualizar as tabelas de fluxos dos comutadores. O módulo *Flow Adaptation* é ilustrado na **Erro! Fonte de referência não encontrada.**.

Figura 9 - Módulo *Flow Adaptation*



A seguir alguns eventos relacionados a este componente (Uma relação completa dos eventos definidos nesse componente é apresentada no **Anexo E – Eventos do CAARF**):

- Eventos de Recebimento;

    - Recebimento da inclusão de novas regras de encaminhamento;
- Outros Eventos;
    - Consulta ao *Flow Repository*;
    - Aplicação de novas regras de encaminhamento nos comutadores.

Após discutir os submódulos do módulo *Context-Based Forwarding Management* e seus respectivos eventos, os principais aspectos sobre a gestão de encaminhamento são apresentados.

3.6 GESTÃO DE ENCAMINHAMENTO

Após haver a aquisição de novas informações de contexto dos inúmeros dispositivos e/ou aplicações da rede, os dados são avaliados pelo módulo *Context Model Management* e, no fim dessa análise, as informações já tratadas desses dispositivos são gravadas no repositório de contexto e, através de notificações, o módulo *Context-Based Forwarding Management* é informado de sua disponibilidade para análise e processamento. Desta forma, as informações de contexto atualizadas são processadas e, por fim, é avaliada a necessidade de mudança do encaminhamento dos fluxos associados a tais eventos.

Dessa forma, é possível reavaliar, de tempos em tempos, se os caminhos definidos para determinados eventos são os mais adequados, permitindo aliar com mais eficiência os recursos da infraestrutura com a necessidade do usuário final, proporcionando assim uma melhor condição de uso da rede por partes dos usuários em geral e um uso otimizado dos recursos disponíveis sem priorizar qualquer tipo de tráfego.

Com isso, um tráfego de dados usando um determinado caminho e, durante as análises de contexto periódicas, que são feitas de acordo com o que for definido pelo administrador da rede, determinado fluxo pode ser e encaminhado usando outra estratégica mais eficaz para o contexto, sem prejudicar a comum ligação entre usuários. Em abordagens mais tradicionais, essa mudança é impossível, em alguns casos, e pouco eficiente, em outros, enquanto ocorre a transmissão dos pacotes de uma de terminada conexão, impedindo uma rápida adaptação da rede às suas próprias demandas, também, às demandas do usuário.

Visando a melhor qualidade possível para uma dada comunicação, existem regras que são definidas pelo administrador da rede que norteiam as escolhas dos novos caminhos. Essas regras estão ligadas a várias situações que devem ser analisadas e que podem ser relacionadas aos próprios caminhos registrados dentro do repositório do controlador de contexto. Uma forma de tratamento dessas regras pode ser feita dividindo-as em:

- **Regras baseadas em mudança de contexto:** Regras relacionadas às mudanças de contexto já processadas e disponíveis para analises de mudanças de encaminhamento. Com base no novo contexto, são avaliados todos os caminhos e definidos quais os principais ou alternativos que ofereçam uma melhor qualidade para uma dada relação que foi tratada na base de contexto. Segue abaixo alguns exemplos que estão relacionados com esse grupo de regras:
    - Mudança de contexto por degradação de MOS, inclusive uma métrica importante para a validação deste trabalho.
    - Mudança de contexto por conta de um alto ruído na comunicação de uma relação usando uma aplicação VOIP.
- **Regras baseadas na mudança da conexão:** Regras relacionadas às condições de tráfego das conexões. Um exemplo seria em um dado momento uma determinada conexão indisponível, proporcionando atrasos ou qualquer irregularidade que prejudique o uso de certas aplicações. Abaixo alguns exemplos relacionados a esse grupo de regras:
    - Indisponibilidade da conexão;
    - Degradação da conexão relacionada a: Diminuição da largura de banda, aumento do atraso, aumento na taxa de erros ou aumento do *jitter*;
    - Identificação de uma sobrecarga de tráfegos na conexão.
- **Regras baseadas na mudança da camada encaminhamento:** Regras relacionadas às condições que os equipamentos da camada de encaminhamento estão no momento da avaliação dos caminhos. Por exemplo, é perfeitamente possível que um roteador esteja sem conectividade ou mesmo que os parâmetros coletados não estejam compatíveis com uma determinada aplicação. Além disso, pode ser que, embasados nas análises do módulo *Context-based Forwarding*

Management, este roteador esteja com sua capacidade muito utilizada, correndo o risco de comprometer não só seus próprios, mas também os novos fluxos que serão escolhidos. Abaixo alguns exemplos relacionados com esse grupo de regras:

- Indisponibilidade do roteador;
- Sobrecarga de processamento no roteador;
- Falha de hardware no roteador, e
- *Timeout* na comunicação com o roteador.

A Tabela 1 apresenta esses grupos de regras de encaminhamento de forma mais detalhada, regras essas que devem ser analisadas pelo módulo *Context-Based Forwarding Management* voltadas para a implementação do trabalho.

Tabela 1 - Relação de regras de encaminhamento (Gestão de Encaminhamento)

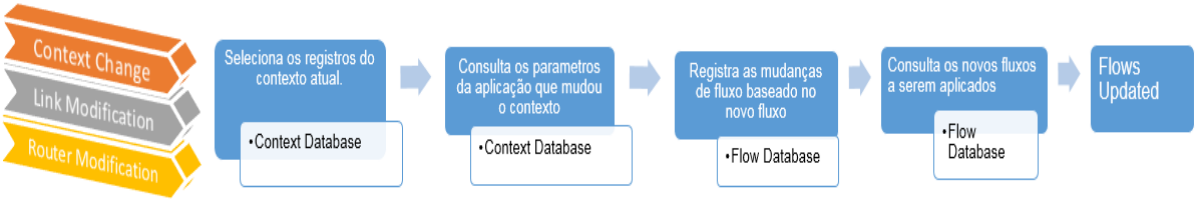
Informação Contextual	Agente de Coleta	Evento	Regras	Motivo
QoE / QoD / QoS	Gerenciamento de Contexto	Mudança de Contexto	Verificar no repositório de contexto se existe solicitação do usuário	Necessário para a mudança de caminho por solicitação do usuário
QoD	Conexão	Indisponibilidade	Verificar a cada coleta de dados se existe disponibilidade do link	Necessário para a mudança de caminho de eventos e invalidar caminho no banco com informações de encaminhamento
QoS	Conexão	Parâmetros de rede	Verificar a cada coleta de dados se estão mantidos os parâmetros de rede como atraso, banda, *jitter*	Necessário para a mudança de caminho de eventos e mudar essas informações no banco com dados de encaminhamento

Informação Contextual	Agente de Coleta	Evento	Regras	Motivo
QoD	Conexão	Sobrecarga	Verificação feita a cada mudança de contexto	Necessária verificação pelo módulo para permitir a escolha de outros caminhos menos congestionados
QoD	Roteador	Timeout	Sempre verificar se a comunicação com os roteadores está intermitente	Necessário para a mudança de caminho para evitar períodos de indisponibilidade do roteador
QoD	Roteador	Indisponibilidade	Sempre verificar se os roteadores estão ativos. Depois de várias verificações que o dado é invalidado no banco	Necessário para a mudança de caminho de eventos e invalidar roteador no banco com informações de encaminhamento
QoS	Roteador	Mudança de Capacidade	Sempre verificar se os roteadores ativos continuam com a mesma capacidade de processamento	Necessário para a mudança de caminho de eventos em caso de mudança de capacidade que pode afetar aplicações que usam tal caminho
QoD	Roteador	Overhead de Processamento	Sempre verificar se os roteadores ativos continuam com a mesma capacidade de processamento	Necessário para a mudança de caminho de eventos momentaneament e em caso de identificação de stress do roteador
QoD	Roteador	Falha de Hardware	Verificar de tempos em tempos se os roteadores ativos	Necessário para a mudança de caminho de eventos momentaneament

Informação Contextual	Agente de Coleta	Evento	Regras	Motivo
			continuam com a mesma disponibilidade de recursos de hardware	e em caso de identificação de perda de capacidade

Todo o tratamento que é dado à informação de contexto dentro do processo de análise, e consequentemente seu encaminhamento, é descrito através do esquema ilustrado na **Erro! Fonte de referência não encontrada.**.

Figura 10 - Tratamento da informação de contexto



A **Erro! Fonte de referência não encontrada.** tem como objetivo apresentar o fluxo principal de verificação das regras de encaminhamento, regras estas que são executadas no submódulo *Forwarding Processing Rules*. É importante observarmos algumas circunstâncias que influenciam no processo de validação das mesmas:

- Mudança de Contexto em que se encontra;
- Mudança do canal em que está atuando;
- Mudança no roteador ou no comutador.

De acordo com a **Erro! Fonte de referência não encontrada.**, a seleção dos registros no *Context Model View* é a primeira das cinco fases. Dada a sua importância na obtenção das informações atuais do contexto da rede, torna-se mais viável a recuperação dos parâmetros padrões (informações de contexto) destas aplicações. Com essas informações recuperadas do *Context Database,* para cada contexto de cada relação é identificada a aplicação que está sendo utilizada e, a partir dela, são recuperados os parâmetros padrões destas aplicações.

Em relação aos parâmetros das aplicações recuperados, são obtidas as opções de caminho para esse contexto e, assim, dentre várias opções, é escolhida a que melhor adapte o tráfego do novo contexto a um caminho com melhor qualidade, em comparação com o caminho anteriormente usado por essa relação.

Com as escolhas de caminho realizadas para todos os novos contextos da rede, pode ser realizada a etapa final que é a identificação dos novos caminhos de fluxo a serem aplicados e assim aplicá-los na camada de encaminhamento a fim de melhor o tráfego das relações afetadas pela mudança de contexto naquele instante.

3.7 CONCLUSÃO

De forma a implementar uma solução baseada na qualidade de experiência do usuário, foi aplicado o arcabouço para o Roteamento Adaptativo Baseado em Contexto (CAARF).

Este arcabouço, foi apresentado levando em consideração sua visão expandida e ao mesmo tempo proporcionando uma abordagem estratégica versus uma abordagem baseada somente em parâmetros de rede que atuam apenas sobre QoS, Layer2, Layer3 e etc.

Sua estratégia está diretamente ligada ao seu conteúdo amplo, permitindo conhecer o histórico de uso das aplicações/usuários, as próprias percepções dos usuários, bem como as variáveis de Network como: QoS, QoD e QoE.

Dessa forma é possível afirmar que um usuário qualquer através de parâmetros de qualidade de experiência, indique indícios de baixa qualidade, essa informação é tratada pelo controlador de contexto e processe a mudança de caminho, possibilitando com isso uma melhoria exponencial na conexão, e consequentemente no ganho de qualidade na comunicação para o usuário final.

De forma resumida, o arcabouço definido pelo CAARF define, portanto, uma forma de realizar o roteamento adaptativo de tráfego com base em informações contextuais, dando importância a dados relacionados à experiência do usuário.

4 IMPLEMENTAÇÃO: GESTÃO DE ENCAMINHAMENTO

4.1 INTRODUÇÃO

Neste capítulo são apresentados os principais aspectos relacionados à implementação de uma arquitetura para o arcabouço *Context-Aware Adaptive Routing Framework* (CAARF), mais especificamente dando enfoque ao módulo funcional de gestão de encaminhamento.

Para a apresentação dos aspectos inerentes à implementação de uma arquitetura para o CAARF, são abordados ao longo deste capítulo os seus requisitos funcionais e não funcionais; a sua arquitetura; modelagem da persistência; as linguagens e ferramentas responsáveis pela implementação da arquitetura; os requisitos da instalação e preparação do ambiente de implementação; uma breve discussão sobre as dificuldades e lições aprendidas; e conclusão acerca das contribuições apresentadas ao longo deste capítulo.

4.2 REQUISITOS FUNCIONAIS E NÃO FUNCIONAIS

Os requisitos de software representam descrições dos serviços que devem ser fornecidos pelo sistema e as suas restrições operacionais (SOMMERVILLE, 2007). Em (PFLEEGER, 2004) o requisito de um sistema é descrito como uma característica do sistema ou a descrição de algo que o sistema é capaz de realizar para atingir os seus objetivos, e em (ROBERTSON & ROBERTSON, 2006) um requisito é definido como algo que o produto tem de fazer ou uma qualidade que ele precisa apresentar.

Com base nessas definições, pode-se dizer que os requisitos de um sistema incluem as especificações dos serviços que o sistema tem de oferecer, restrições de operação e até restrições inerentes ao seu processo de desenvolvimento. Dentre os diferentes tipos de requisitos dentro da engenharia de software, os requisitos de um sistema normalmente são caracterizados como funcionais ou não funcionais (SOMMERVILLE, 2007).

4.2.1 Requisitos Funcionais

Os requisitos funcionais tratam dos serviços que o sistema deve prover, descrevendo o que o sistema deve fazer, podendo descrever, ainda, como o sistema deve reagir a entradas específicas, bem como o que o sistema não deve fazer (SOMMERVILLE, 2007). Para implementação da arquitetura baseada no arcabouço CAARF, foram levadas em consideração os seguintes requisitos:

- O módulo de encaminhamento deve ter condições de identificar o novo contexto;
- A arquitetura para o arcabouço deverá suportar a implementação de diferentes cenários de validação;
- O módulo de encaminhamento deve possuir de forma atualizada todos os possíveis caminhos cadastrados;
- As regras de encaminhamento devem ser previamente cadastradas pelo administrador;
- O módulo de encaminhamento deve ser capaz de receber notificações provenientes do *Context Model Management*;
- O módulo de encaminhamento deve ser capaz de consultar a tabela de flow_id (utilizada para identificar uma conexão, uma aplicação e um usuário) parâmetros relacionados à rede (QoS), ao usuário (QoE) e aos dispositivos de rede (QoD);
- O módulo de encaminhamento deve poder calcular novos requisitos de encaminhamento;
- O módulo de encaminhamento deve procurar na tabela de rotas/filas a nova rota;
- O módulo de encaminhamento deve notificar o *Flow Adaptation* sobre nova rota;
- O módulo *Flow Adaptation* deve aplicar os novos caminhos na camada de encaminhamento sejam em roteadores ou mesmo em comutadores;
- O módulo de encaminhamento deve gerar log de alterações;
- Antes de qualquer análise de caminho, o módulo de encaminhamento deve realizar a verificação das conexões usadas por este. Em caso de alguma conexão estar inativa ou mesmo foi detectada com problema,

todos os caminhos que utilizam esta conexão não devem ser selecionados para a escolha de um novo caminho;

- O módulo de encaminhamento (*Forwarding*) deve calcular a banda necessária com base em "*concurrent calls*" e no Codec empregado;
- O módulo de encaminhamento (*Forwarding*) deve determinar a utilização de nova fila para aquele fluxo de ligações;
- O módulo de encaminhamento (*Forwarding*) deve poder criar filas e simular as rotas disponíveis através do algoritmo HTB (*Hierarchical Token Bucket*);
- A arquitetura deverá permitir a visualização do status da fila que está sendo usada em tempo real;

4.2.2 Requisitos não funcionais

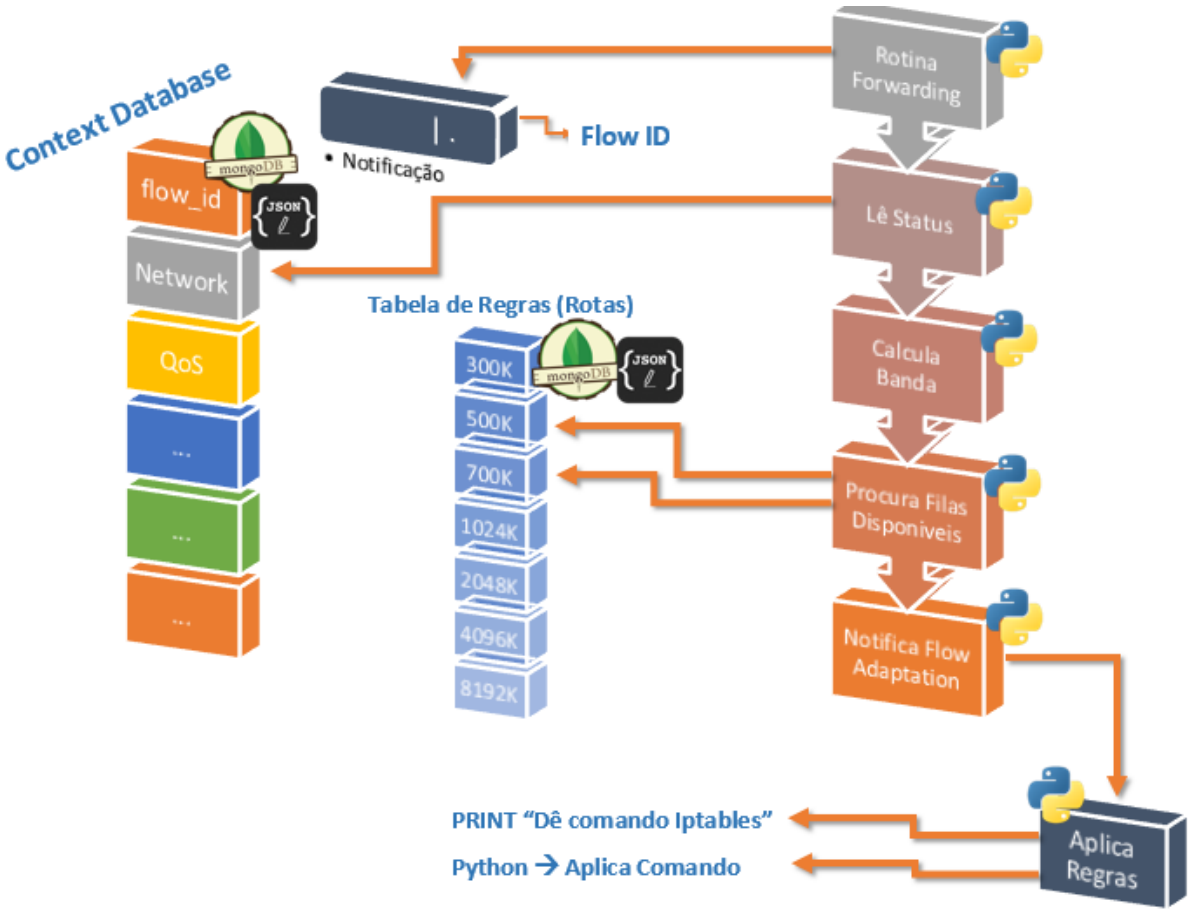
Os requisitos não funcionais descrevem restrições sobre os serviços ou funções oferecidas pelo sistema, tendo origem nas necessidades dos usuários, em restrições políticas organizacionais, em restrições de orçamento, em necessidades de interoperabilidade com outros softwares ou hardwares ou em fatores externos como regulamentos e legislações (SOMMERVILLE, 2007). Dentre os diversos requisitos não funcionais inerentes à implementação de uma arquitetura para o arcabouço CAARF estão:

- As rotinas implementadas para o tratamento das notificações devem ser independentes, todas consultando o banco de dados em nuvem;
- O código deve ser devidamente comentado com o propósito de facilitar possíveis manutenções;
- É necessário um software de coleta de eventos;
- O ambiente de testes deve garantir confiabilidade, configurando todos os códigos de forma estática, eliminando assim a indisponibilidade do serviço devido a uma interrupção brusca;
- A solução implementada deve apresentar um ganho em performance, dada a necessidade de reposta otimizada do sistema;
- É necessária uma máquina centralizadora permitindo, através de rotas, a comunicação de duas redes diferentes.

4.3 MODELAGEM DA ARQUITETURA

A fim de facilitar a compreensão e análise dos dados obtidos, apresentamos a modelagem da arquitetura proposta:

Figura 11 - Modelagem da Arquitetura



A **Erro! Fonte de referência não encontrada.** tem como objetivo ilustrar o funcionamento da arquitetura proposta, mais especificamente voltada à abordagem de encaminhamento.

A partir de uma coleta obtida através de um agente de coleta (*Context Agent/Midleware*), alinhada com o modelo de contexto, é gerada uma notificação para a abordagem de encaminhamento (*Context-Based Forwarding Management*). A partir desta notificação, que pode ser qualquer dado relacionado à qualidade do usuário para com uma aplicação, uma rotina em Python fica responsável por obter o *Flow_id*, um parâmetro que foi criado para podermos associar o fluxo com o usuário, com a

sessão e com a aplicação, basicamente a identificação do fluxo daquele usuário. Na solução proposta neste trabalho, o MOS (*Means Opinion Score*) está sendo utilizado para aferir a qualidade dessa comunicação.

A partir desse *Flow_id*, é possível através de uma leitura de estado, observar os outros parâmetros relacionados aquele usuário, que foi identificado através do *Flow_id*, como por exemplo:

- Informações relacionadas à rede (*Network*):
- Endereço IP de origem;
- Endereço IP de destino;
- Endereço Mac;
- Informações de Qualidade de Serviço:
- Perda de pacotes;
- Latência;
- *Jitter;*

Essas e outras informações capacitam a continuidade do fluxo, e consequentemente seu objetivo final, que é o de proporcionar um caminho de melhor qualidade para aquele *Flow_id*.

Importante citar que a obtenção dessas informações é feita através do *Context Database* escrito em MongoDB (MONGODB, 2015) auxiliados pelo *JSON/BSON* (MONGODB, 2015), que serão discutidos nas próximas sessões.

Após a obtenção dessas informações, outra rotina irá realizar o cálculo da largura de banda disponível, que por sua vez será mensurada a partir das quantidades de chamadas atuais e o tipo de codec utilizado. De forma a ilustrar essa rotina, o estudo de caso apresentado no Capítulo 5 apresenta um cenário de implementação, onde é representado um determinado sistema de PBX IP, SIP, escalável, e um fluxo variável de chamadas entre o host origem – onde se localizam os clientes chamadores, e o host destino - neste caso o sistema de PBX IP, que é o atendedor das chamadas telefônicas VoIP.

Com o cálculo realizado é possível consultar na tabela de rotas qual melhor rota para aquele fluxo, ou para aquela quantidade de ligações. Com isso, uma outra rotina notifica o módulo *Flow Adaptation* que por sua vez aplica as regras de encaminhamento.

Do ponto de vista do Gerenciador de Encaminhamento, o Gerenciador de Contexto informa a você um número de *Flow_id* (por aquela FIFO de comunicação). O Gerenciador de Encaminhamento utiliza este *Flow_id* para identificar o documento referente à conexão que está ativa naquele momento (um documento de um usuário/aplicação no MongoDB). Este documento contém todas as relações ativas naquele momento, incluindo as de conexão. O *Flow_id* seria como uma chave de acesso ao documento que deve ser acessado pelo Encaminhador a fim de executar os cálculos necessários à mudança de fila de encaminhamento.

A implementação das regras poderá ser realizada através de uma mensagem na tela, do tipo “Execute a regra X no *iptables*” ou o código Python pode se comunicar com o módulo Netfilter/HTB (NETFILTER, 2015). O Netfilter é o módulo que fornece ao sistema operacional Linux as funções de firewall, NAT e log dos dados que trafegam por rede de computadores. Já o HTB é responsável pela implementação de uma fila com suporte a várias classes para controle de tráfego e a partir disso é possível configurar uma nova rota associada a uma fila que se adapte ao fluxo utilizado.

Na **Erro! Fonte de referência não encontrada.** o comando *iptables* é chamado e na tabela mangle (**iptables -t mangle**), é feito o encaminhamento para os ips destino, utilizando a *chain forward* **(-A FORWARD**), e definindo para cada um deles um alvo (**-j CLASSIFY**), que por sua vez é a classificação da fila correspondente a um tipo de banda diferente (**--set-class 1:30**)

Figura 12 - Exemplo de configuração de IP de acordo com a classificação da fila

```
1  # Configurando a rede interna de acordo com a classificação da fila definida - Fila 1024K
2
3  iptables -t mangle -A FORWARD -d 192.168.0.102 -j CLASSIFY --set-class 1:30
4  iptables -t mangle -A FORWARD -d 192.168.0.101 -j CLASSIFY --set-class 1:30
5  iptables -t mangle -A FORWARD -d 192.168.1.101 -j CLASSIFY --set-class 1:30
```

4.4 MODELAGEM DA PERSISTÊNCIA

Para implementação de uma arquitetura para o arcabouço CAARF, a persistência foi dividida em duas partes: a persistência em relação à coleta dos dados para serem analisados pelo controlador de contexto (CAARF) e a persistência relacionada aos dados armazenados pelo controlador de contexto para o seu funcionamento.

Tanto na coleta de dados feito de forma distribuída pelos *Context Agent*, quanto nos dispositivos de encaminhamento, que tem sua coleta feita pelo *Context Agent Middleware*, é definida uma estrutura de marcação utilizando JSON/BSON.

As informações definidas são: individualidade, localização, relação, atividade e tempo, além dos dados relacionados à Qualidade de Serviço, tais como atraso, *jitter* e largura de banda; Qualidade de Dispositivo, tais como capacidade de processamento de um roteador ou comutador ou mesmo a identificação da existência de GPS em um determinado dispositivo que irá se comunicar na rede e, por fim; qualidade de experiência, que seria a impressão do usuário sobre o uso de determinada aplicação, como o MOS (*Mean Opinion Score*).

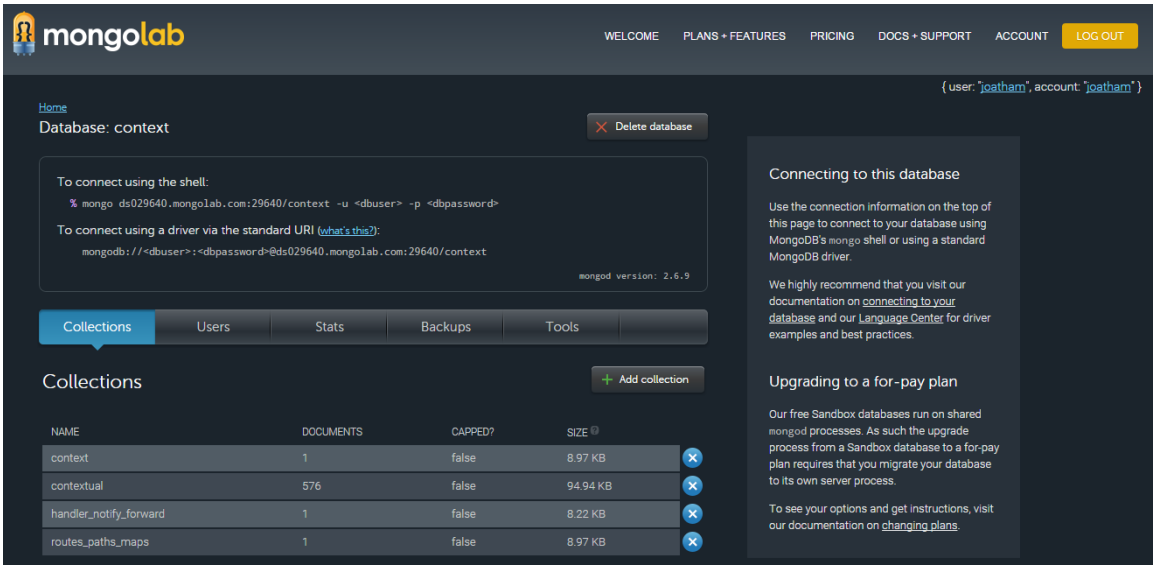
A estrutura de marcação *usada é o JSON/BSON (JavaScript Object Notation/Binary JSON*) (JSON, 2015), por conta de algumas características como por exemplo:

- Simplicidade - *JSON* é muito mais simples que outras linguagens de marcação como XML;
- Extensibilidade - *JSON* não é extensível porque ele não precisa ser. *JSON* não é uma linguagem de marcação de documentos, por isso não é necessário definir novas *tags* ou atributos para representar dados nele;

Portanto, conforme descrito, são apresentados nos **Anexo A** – **Mapa de Filas (Bson/Json e Mongodb)** e **Anexo B** – **Fila Notification (Bson/Json e Mongodb)** alguns exemplos de JSON/BSON de coleta de dados que são gerados pelos *Context Agent* distribuídos na rede. Esses *JSON's* irão refletir o contexto da rede naquele momento em que os dados estão sendo coletados.

Outro aspecto que influenciou diretamente na performance e no ganho de tempo da implementação realizada, foi a utilização do MongoDB/MonogLab. Dado que o banco de dados é disponibilizado nas nuvens, todos os scripts em Python para a realização dos testes são executados de forma independente, pois todos consultam e acessam o banco de dados MongoDB nas nuvens via MongoLab, através do comando de acesso: (mongodb://<dbuser>:<dbpassword>@ds029640.mongolab.com:29640/context), como ilustra a **Erro! Fonte de referência não encontrada.**.

Figura 13 - Representação do MongoLab



Na **Erro! Fonte de referência não encontrada.** a seguir é descrito a representação *JSon* no MongoDB já com alguns dados coletados:

Figura 14 - Representação JSON no MongoDB com alguns dados coletados


```
{
    "_id": {
        "$oid": "558aa522e4b0be7838515c55"
    },
    "flow_id": 4,
    "path": [
        {
            "pathid": 4,
            "fwrid_ini": 0,
            "fwrid_fin": 0,
            "hop_count": 0
        }
    ],
    "link": [
        {
            "linkid": 4,
            "fwrid_ini": 0,
            "fwrid_fin": 0,
            "status": "true"
        }
    ],
}
```


Essas regras confirmam as características acima citadas pela escolha da linguagem, se mostrando de fácil percepção e implementação. O resto das regras é descrito nos **Anexo A** – **Mapa de Filas (Bson/Json e Mongodb)** e **Anexo B** – **Fila Notification (Bson/Json e Mongodb**.

Dentro do controlador de contexto, foi definida uma segunda estrutura de persistência já sido citada acima. Para todas as informações tratadas dentro do controlador, existe uma estrutura de tabelas normalizadas e estruturadas com uma certa flexibilidade, e representadas por regras, que permitem a construção de um controlador de contexto adaptável a uma série de tecnologias de rede, sem especificamente ficar aderida a uma arquitetura.

4.4.1 Regras de encaminhamento

Após a aquisição de novas informações de contexto dos inúmeros dispositivos e/ou aplicações da rede, os dados são avaliados pelo módulo *Context Model Management* e, no fim dessa análise, as informações já tratadas desses dispositivos são gravadas no repositório de contexto. Através de notificações, o módulo *Context-based Forwarding Management* é informado de sua disponibilidade para análise e processamento.

Desta forma, as informações de contexto atualizadas são processadas e, por fim, é avaliada a necessidade de mudança do encaminhamento dos fluxos associados a tais eventos.

Com isso, é possível reavaliar, de tempos em tempos, se os caminhos definidos para determinados eventos são os mais adequados, permitindo aliar com mais eficiência os recursos da infraestrutura com a necessidade do usuário final, proporcionando assim uma melhor condição de uso da rede por partes dos usuários em geral e um uso otimizado dos recursos disponíveis sem priorizar qualquer tipo de tráfego.

Portanto, é possível iniciar um tráfego de dados usando um determinado caminho e, durante as análises de contexto periódicas, que são feitas de acordo com o que for definido pelo administrador da rede, determinado fluxo pode ser encaminhado usando outra estratégia mais eficiente para o contexto, sem prejudicar a comunicação entre usuários. Em abordagens mais tradicionais, essa mudança é impossível em alguns casos, e pouco eficiente em outros, enquanto ocorre a transmissão dos pacotes de uma de terminada conexão, impedindo uma rápida adaptação da rede às suas próprias demandas, também às demandas do usuário.

A seguir algumas regras de encaminhamento que devem ser analisadas pelo módulo *Context-based Forwarding Management* especificamente para a implementação voltado para filas e Netfilter:

- RELATION – Armazena todos os dados coletados pelo *Context Agent/Context Agent Middleware* e que são enviados, posteriormente, para o *Context Handler* no qual os insere, inicialmente, e cujos dados são tratados pelo *Context Model Management* para verificação das regras de contexto e depois para verificação do contexto;
- PATH – Armazena os dados dos caminhos cadastrados pelo administrador de rede. Esses registros descrevem um cadastro básico descritivo do caminho sem detalhar origem-destino e informa a quantidade de saltos que existe naquele caminho;
- LINK – Armazena os dados de conexões entre os roteadores ou comutadores. As conexões são parte do caminho completo e para estes são registradas, em caso de problema, através de mudança de estado, as conexões inativas;
- LINKPARAMETER – Armazena os dados sobre os parâmetros cadastrados no *Context Database* para cada conexão cadastrada pelo administrador de rede. Cada conexão pode considerar diferentes parâmetros importantes para a análise das regras de encaminhamento e, portanto, de acordo com cada conexão cadastrada podem haver;
- PATHLINK – Armazena os dados de caminhos e as conexões associadas. Essa tabela é um relacionamento muitos-para-muitos entre as tabelas *Path* e *Link*;
- QUEUE – Armazena os dados relacionados às filas. Nesta arquitetura, a estratégia de fluxo é realizada através de filas simulando diferentes caminhos de rede;
- FORWARDINGEVENT – Armazena os dados relacionados aos novos eventos associados aos caminhos de encaminhamento escolhidos. Essa tabela será populada logo após o módulo *Context-Based Forwarding Management* receber a notificação do módulo *Context Model Management* da existência de novo contexto. Após inserir o dado nessa tabela, o *Context-Based Forwarding Management* fará a escolha, dentre os

caminhos ativos existentes, através do processamento das regras de encaminhamento pré-cadastradas e de acordo com os dados de contexto das aplicações associadas às relações que possuem novo contexto.

Percebeu-se que de todas as regras de encaminhamentos definidas e descritas no **Anexo F** – **Regras de Encaminhamento**, para o tipo de experimento usado na validação, serão necessárias somente estas 7 regras, uma vez que possuem parâmetros úteis para a realização dos testes.

4.5 LINGUAGEM E FERRAMENTAS DE IMPLEMENTAÇÃO

Algumas ferramentas foram essenciais para a implementação desta arquitetura. Para implementação das rotinas tanto de coleta, tratamento e aplicação de novos fluxos foi utilizado a linguagem de programação Python. Na definição das filas simulando os vários caminhos da rede, trabalhamos com o conceito de *Traffic Control*, aliado ao algoritmo HTB e para aplicarmos as mudanças de filas, o módulo Netfilter. O Python é uma linguagem livre e suporta inúmeras plataformas, isso prova que os programas escritos em uma determinada plataforma serão executados sem nenhum problema na maioria das outras plataformas sem nenhuma modificação; O *Traffic Control* trabalha com filas que denominamos *queuing* de forma que quando os pacotes são recebidos eles são tratados e então se torna possível a alteração da prioridade do pacote, da limitação da fila em que ele se encontra e ainda a velocidade de transmissão dentre outros. Estaremos utilizando a ferramenta *Traffic Control*, através do seu comando respectivo chamado tc. Aliado ao *Traffic Control,* temos o algoritmo HTB (*Hierarchical Token Bucket)*, dentre os vários motivos está o fato do HTB ser mais rápido e compreensível.

4.5.1 *Traffic Control* com HTB

No intuito de substituir o *Class Based Queue (CBQ)*, foi proposto o *Hierarchical Token Bucket (HTB)* (LUXIK, 2015), uma vez que o HTB apresenta uma melhor performance. Com isso, diferentes filas podem ser implementadas com suporte a várias classes para o controle de tráfego. Dentre os parâmetros utilizados

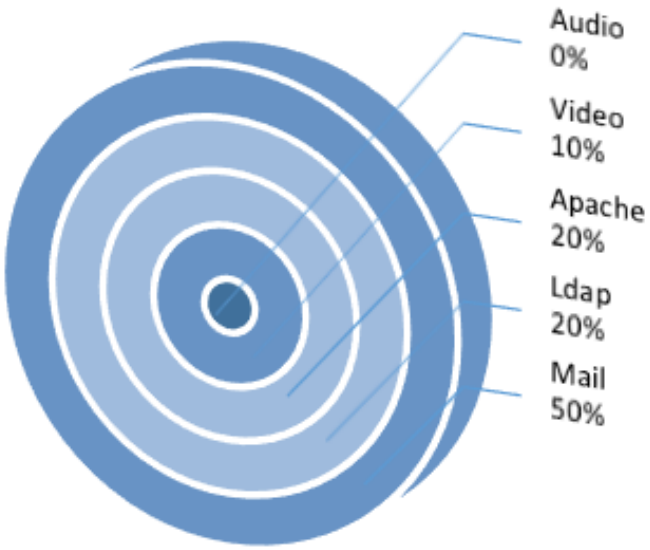
se destacam *rate* e *cell*, onde *rate* é a largura de banda disponível para uso nessa classe e *cell* a largura máxima de banda que uma classe poderá consumir.

Todo esse controle da divisão de banda é feito pela *qdisc* (abreviatura de *Queue discipline* ou disciplina de enfileiramento). Em resumo, refere-se à disciplina de controle de tráfego principal do HTB, ou seja, quando uma determinada classe requisita largura de banda, a ela é disponibilizada um valor de banda mínimo que é especificado em rate. Caso a classe não esteja utilizando sua banda, o *qdisc pai* distribui pelas outras classes conforme solicitado até ao máximo especificado em *cell*.

Uma outra opção é não havendo a necessidade de alocação de valores máximos em *cell*, é possível optar em apenas colocar o mesmo valor de *rate* em *cell*, de forma que aquela classe nunca ultrapasse o valor de banda mantido para a mesma.

A **Erro! Fonte de referência não encontrada.** ilustra um simples exemplo de compartilhamento da banda entre classes de serviços.

Figura 15 - Exemplo de compartilhamento de banda



Observa-se na **Erro! Fonte de referência não encontrada.** que existe uma alocação de banda para cada serviço, entre áudio, vídeo, Apache (APACHE, 2015), Ldap (OPENLDAP, 2015) e correio eletrônico. Porém, a imagem retrata a banda compartilhada sem o valor máximo de banda.

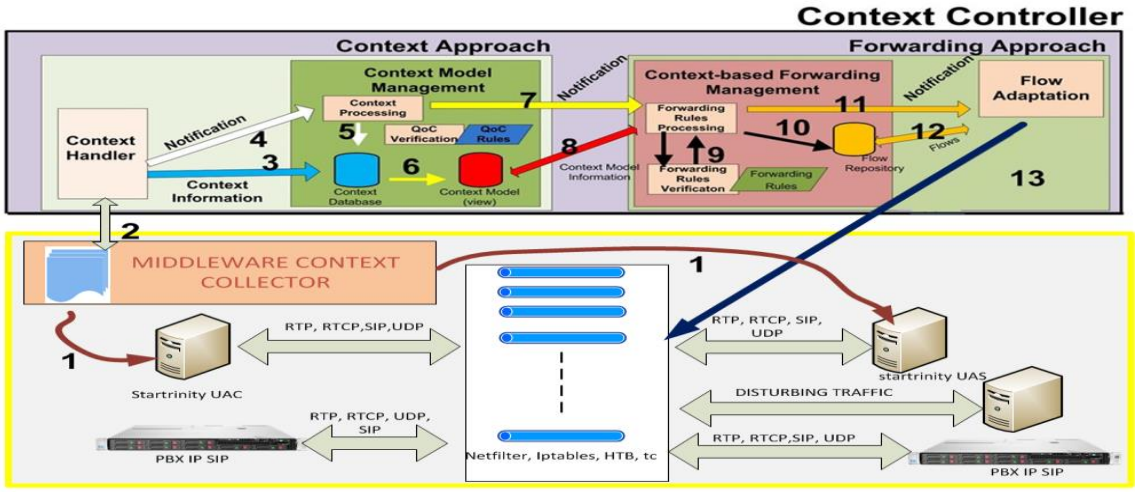
4.6 REQUISITOS DE INSTALAÇÃO

Para validarmos a arquitetura é necessário que um conjunto de requisitos seja respeitado. Segue-se uma lista dos requisitos para instalação e execução do dos cenários:

- Os sistemas operacionais que foram precisos para montar o laboratório:
  - Debian7 Wheezy 32bits;
    - Instalação do python 3.3;
    - Instalação do servidor SSH;
    - Configuração de 3 interfaces de rede;
      - eth0 – Nat;
      - eth1 – bridge – 192.168.100.1
      - eth2 – bridge – 192.168.200.1
    - Configuração das rotas para comunicação de duas redes diferentes;
    - Configuração do sysctl, habilitando o repasse dos pacotes entre as interfaces;
    - Configuração do *Iptables* para compartilhamento da internet;
    - Criação das filas com HTB.
  - Windows7 Ultimate Service Pack 1
    - Configuração da rede em *bridge* com o gateway, para ingressar na rede e consequentemente na internet;
    - Configuração do Ip na faixa 100;
    - Instalação e configuração do Software *Startrinity* como UAC (*User Agent Client*);
  - Windows7 Ultimate Service Pack 1
    - Configuração da rede em bridge com o gateway, para ingressar na rede e consequentemente na internet;
    - Configuração do Ip na faixa 200;
    - Instalação e configuração do Software *Startrinity* como UAS (*User Agent Server*);
  - Windows7 Ultimate Service Pack 1

- Configuração da rede em bridge com o gateway, para ingressar na rede e consequentemente na internet;
- Configuração do Ip para trabalhar em qualquer uma das faixas;
- Injetar trafego UDP, para auxiliar na degradação do MOS.

Figura 16 - Componentes da implementação



A **Erro! Fonte de referência não encontrada.** acima ilustra o esquema de configuração condizente com a arquitetura da implementação.

4.7 DISCUSSÕES E LIÇÕES APREENDIDAS

Estabelecer um parâmetro de simulação para este ambiente, desde a escolha do tipo de tratamento a ser realizado até a decisão pela utilização de filas, foi uma dificuldade.

Essa analogia só pode ser feita, devido ao fato de que tanto o modelo de contexto, quanto o arcabouço se adaptam a qualquer tipo de ambiente, e como não seria possível o acesso a uma rede real, com roteadores e comutadores, foi necessário a adoção de uma abstração simulasse basicamente o enlace (*link*).

A infraestrutura para a realização do experimento é composta por 4 máquinas:

- Servidor Linux Debian 7 wheezy 32 bits;
- Máquina Windows 7 (cliente 1);
- Máquina Windows 7 (cliente 2);
- Máquina Windows 7 (cliente 3).

Para que seja possível uma comunicação entre as maquinas, e além disso simularmos duas redes distintas trocando informações, fez-se necessário um *gateway* de borda com três interfaces, papel esse exercido pelo Servidor Linux.

Ainda na preparação deste ambiente, foi adotada para a primeira interface (eth0), uma configurada baseada em NAT (*Network Address Translator),* conectado a um modem ADSL 10Mb. Para a segunda interface (eth1), configurada como *Bridge* foi associada ao IP 192.168.0.100 e a terceira interface (eth2), também como *bridge* associamos o IP 192.168.1.100. Dessa forma, foi possível separar fisicamente duas faixas de rede, através de configurações especificas nos arquivos e /etc/sysctl.conf e /etc/network/interfaces responsável pelo repasse dos pacotes entre as interfaces e pela configuração estática de endereços IP's e rotas respectivamente, esse tipo de configuração é realizada para que as duas faixas de rede (faixa 0 e 1) tenham acesso a rede externa (Internet) e consequentemente permitam uma comunicação entre as outras estações, configurando assim uma pequena infraestrutura de testes.

Para finalizar a configuração do ambiente, foi necessária configuração de rotas entre as duas faixas de rede, para que as máquinas consigam se comunicar mesmo estando em faixas de redes diferentes. Isto foi feito para que durante a simulação de ligações VOIP de um lado para o outro, o Servidor Linux seja o ponto em comum entre eles e assim permitir a configuração de tráfego por meio de filas entre as chamadas.

Dentre várias possíveis soluções, foi considerado a utilização de filas, com isso surge automaticamente o conceito de *traffic shaper*, que prega a personalização do tráfego de diferentes maneiras, através da largura de banda, através do protocolo, através das portas dos serviços.

Uma outra vantagem nesta solução, é que ela é considerada por muitos administradores de rede como um método de *hardening* (REIS, 2010), justamente pelo fato de proporcionar uma maior qualidade aos serviços. No entanto, como a implementação proposta neste trabalho está relacionada o protocolo SIP, não houve necessidade da criação de filas para cada serviço, o foco principal se deu sobre quantidade de filas e o tamanho das mesmas.

Uma outra dificuldade encontrada foi na escolha do software que seria usado para implementação do *Traffic Shapper*, porem optando pelo módulo Netfilter, nativo do Linux, sem contar a facilidade de execução dos comandos e além disso,

praticamente a maioria dos *front-ends* voltados para esse tipo de solução fazem uso deste recurso de forma embutida.

4.8 CONCLUSÃO

A implementação de uma arquitetura baseada no CAARF, utilizando regras de encaminhamento pré-definidas, permitiu comprovar a eficiência do arcabouço em diferentes abordagens, bem como a escolhida que foi tratamento de filas juntamente com módulo Netfilter, permitiu também avaliar a velocidade e facilidade de implementação de um bando orientado a documentos como é o caso do MongoDB.

O foco que foi dado a esta implementação está mais voltado à estratégia na obtenção de desempenho, que automaticamente comprova ser uma estratégia baseada em contexto.

Com isso, este trabalho evolui até proporcionar uma antecipação de comportamento e demanda, levando em consideração a série histórica, perfil de uso, perfil das aplicações e etc.

5 ESTUDOS DE CASO

5.1 INTRODUÇÃO

Neste capítulo são apresentados alguns estudos de casos que permitem ilustrar o funcionamento da arquitetura proposta para o arcabouço CAARF, proporcionando melhores soluções para os serviços baseados na experiência do usuário. Para facilitar o entendimento sobre o funcionamento do mesmo, o nível de dificuldade dos cenários apresentados neste capítulo foi acrescido deforma gradativa. Como forma de expor as potencialidades da solução, os cenários também foram constituídos de forma que atendessem a diferentes aspectos, tais como: diferentes *codecs*, diferentes tamanhos de filas, diferentes clientes.

A demonstração de estudos de casos com implementações de diferentes aspectos é relevante para a validação da proposta da arquitetura. Entende-se nesse trabalho que essa é a melhor alternativa para demonstrar as suas potencialidades e justifica-las através do êxito nas suas implementações.

Desta forma, nas próximas seções apresentamos cinco estudos de caso ilustrando algumas das possíveis formas de validação da arquitetura proposta para o arcabouço *Context-Aware Adaptative Routing Framework* (CAARF).

5.2 CENÁRIO 1

5.2.1 Cenário de estudo

O cenário considerado para o presente estudo de caso é representado por um determinado sistema de PBX IP, SIP e um fluxo variável de chamadas entre o hospedeiro (host origem) – onde se localizam os clientes chamadores, e o host destino, neste caso o sistema de PBX IP, que é o atendedor das chamadas telefônicas VoIP.

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 cuja banda de largura disponível é variável. Estas filas simulam as rotas disponíveis, cuja métrica é a capacidade de banda alocável.

Nestes 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, onde será empregado o protocolo SIP de sessão, RTP/RTCP para controle dos pacotes de voz e UDP como transporte. Serão desta forma inseridos dois tipos de perturbação, e variáveis, sendo a primeira delas o volume de chamadas, considerando canais de comunicação exclusivos para VoIP, e o segundo caracterizado pela inserção de um fluxo de tráfego concorrente e perturbador, a fim de causar interferência na qualidade.

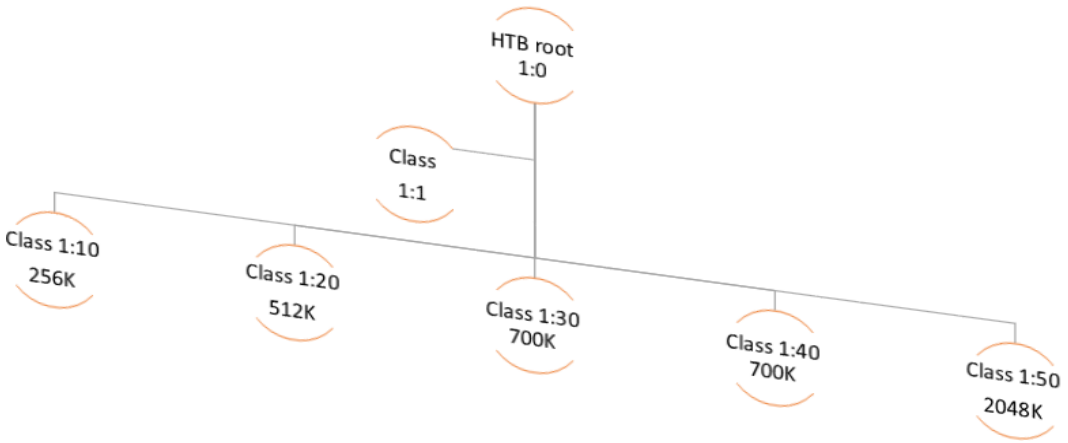
5.2.2 Controle de Tráfego

A seguir alguns parâmetros necessários para criação e validação do 1º cenário de estudo de caso:

- ***rate***:
    a. Banda garantida para classe e sub classes.
- ***ceil*:**
    b. **T**axa máxima que poderá ser emprestada pelo pai (HTB).

Na implementação em questão foi utilizado um enlace (link) de 10mbit (+/- 1MB/s) para *download*, e o tráfego foi separado da seguinte forma, como mostra a Figura 18 a seguir:

Figura 17 - Separação das filas



A seguir uma parte do código usado na implementação de tudo que foi apresentado e que é o script responsável pela criação das filas em nosso experimento, o restante do script encontra-se no **Anexo G** – **Script Controle de Filas/Iptables**:

Figura 18 - Declaração das filas pelo HTB

```
#Define a classe root
tc qdisc add dev $interface root handle 1:0 htb default 90

#Define a taxa total do link da interface
tc class add dev $interface parent 1:0 classid 1:1 htb rate $velol

#Divide o link total da placa em partes
tc class add dev $interface parent 1:1 classid 1:10 htb rate 256kbit ceil 270kbit
tc class add dev $interface parent 1:1 classid 1:20 htb rate 512kbit ceil 530kbit
tc class add dev $interface parent 1:1 classid 1:30 htb rate 700kbit ceil 720kbit
tc class add dev $interface parent 1:1 classid 1:40 htb rate 1024kbit ceil 1040kbit
tc class add dev $interface parent 1:1 classid 1:50 htb rate 2048kbit ceil 2060kbit
```

Desta forma é possível criar 5 tipos de filas de tamanhos diferentes (256k, 512k, 700K, 1024k e 2048K) e a partir destas serão feitos os testes sempre observando os parâmetros de MOS (*Means Opinion Score*).

5.2.3 Sistema de Avaliação

Foi empregado um sistema de simulação baseado no *Startrinity*, uma ferramenta de simulação de chamadas e atendimento por parte do PBX que exerce o papel de chamador e atendedor, aderente ao protocolo SIP, e que compreende um *sniffer* de chamadas embutido específico para VoIP. Desta forma, é possível aferir fim-a-fim o andamento das chamadas, desde o início à sua conclusão e todos os dados referentes à apuração de dados de QoS e QoE são detalhados.

Em síntese, este sistema se comporta como um *sniffer* de chamadas e executa as funções de chamada e atendimento exatamente como um PBX IP padrão. Uma vantagem adicional é a possibilidade de permitir simular chamadas, o que seria impraticável no mundo real. Este sistema é empregado por diversos provedores de serviços de telefonia IP e corporações para aferir e ajustar os parâmetros de rede para tráfego IP de voz.

5.2.4 Sistema de Coleta

O *Context Controller* prevê tarefas distribuídas e, especificamente para aquele referente ao sistema de coleta, esta é realizada pelos agentes de coleta integrantes do Middleware. Para implementar esta tarefa, necessária para todo o experimento, a parte referente à coleta é composta por meio de rotinas escritas em Python que lêem o arquivo de Log de chamadas (CDR) apurado pelo Startrinity e fazem a sua tradução e inserção na base de dados de contexto. Estas rotinas implementam portanto a parte do *Context Agent* e exercem também parte do papel do *Context Handler*. Desta forma, construímos um sistema de geração/simulação de chamadas em ambiente real e um método de coleta, apuração e preenchimento da base de dados de contexto aplicado a este estudo de caso.

5.2.5 Objeto do Experimento

O cenário assim exposto corresponde ao estudo de caso que será objeto de validação por parte do *Context Controller*. 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 *Context Model Management* irá validar alguns dos 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;*
- *concurrent calls;*
- *port range;*
- *application not registered;*
- *Rfactor versus MOS;*
- *User Perception;*
- *codec inválido;*
- *invalid location;*
- *QoS versus MOS.*

Na interpretação dada para este estudo de caso, o *Context Approach* irá validar o QoE. Este será o "*trigger*" do experimento, que desencadeará a notificação ao *Forwarding Approach*. O parâmetro escolhido foi o MOS apurado por uma rotina em Python, que faz a leitura dos parâmetros e os formata em JSON.

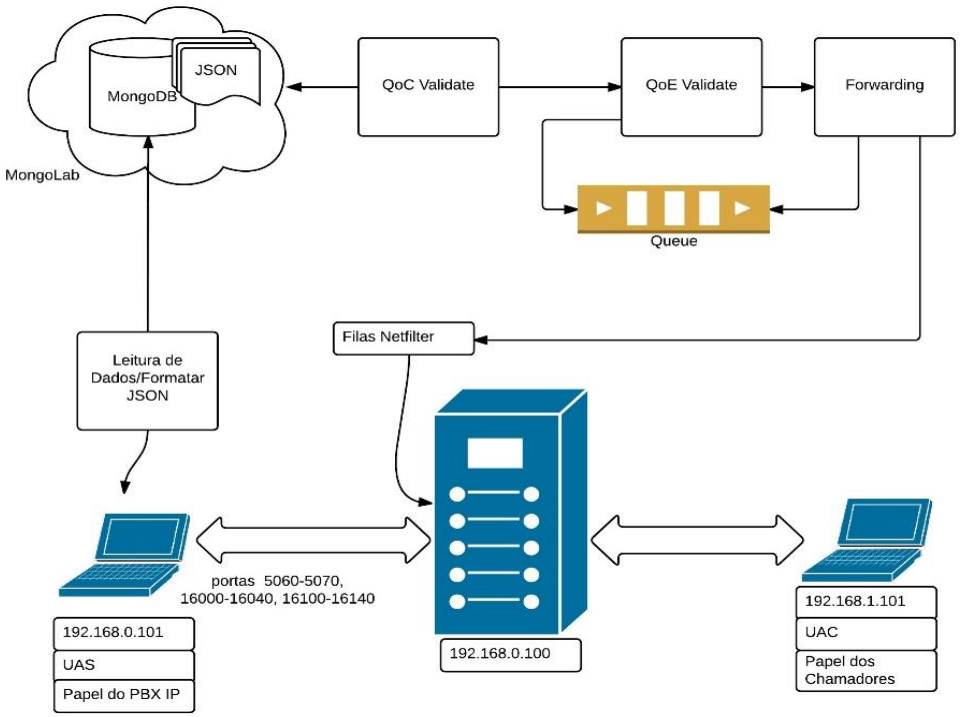
Neste cenário de estudo de caso, o *Host (UAS)* exerce papel do PBX, já o *Host (UAC)* o de grupo de chamadores e o *Host Netfilter* implementa as filas. *Queue* neste caso é a *FIFO* de notificação.

Estes dados, em seguida, são lidos pelo *Context Handler*, implementado por uma rotina em Python que agrega os dados e preenche a base de dados de *Collections* de contexto. O *Context Management* realiza a validação de QoC (omitida até o momento) e também faz a validação do MOS.

Ocorrendo a violação de MOS, uma notificação é gerada para o *Forwarding Approach* – "*queue*" na **Erro! Fonte de referência não encontrada.**, que é uma FIFO de notificações. O módulo de *Forwarding* é então acionado para determinar uma nova fila no *Netfilter* cujas métricas correspondam aos requisitos necessários para acomodar as novas condições apresentadas no Contexto.

Neste primeiro experimento 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.

Figura 19 - Cenário do Estudo de Caso



São realizadas passagens experimentais com o Codec G.711 e G.729 para apurar as diferenças de uso e sensibilidade para diferentes métodos de compressão de voz.

5.2.6 Dinâmica do experimento

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

Para tanto, foram criadas nas rotinas pontos de parada para que possa ser dado 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, o *screenshot* do banco de dados a cada rodada e a nova posição, sem prejuízo da dinâmica real. 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.

Assim sendo, a cada 25 chamadas realizadas, efetiva-se a apuração e validação do MOS. O *Forwarding* é informado no caso de violação do MOS definido no SLA daquele usuário. O Forwarding, neste caso específico, calcula a banda necessária com base em "*concurrent calls*" e no Codec empregado naquele momento, e determina 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 *Forward* e em seguida aplicado o respectivo comando sobre o *Netfilter (Iptables)*.

A seguir são apresentadas as telas obtidas a cada rodada, destacando-se os campos de MOS e R-factor, bem como *concurrent calls*. 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 lugar e em máquinas e redes distintas. O *Context Controller* pode ser executado em uma determinada localidade, assim como o *Forwarding* 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 cliente permite visualizar o estado do banco e suas respectivas variáveis a qualquer tempo, a partir de qualquer localidade. Ao final de cada passo realizamos um "*refresh view*" e é feita a verificação de mudança de estado das variáveis.

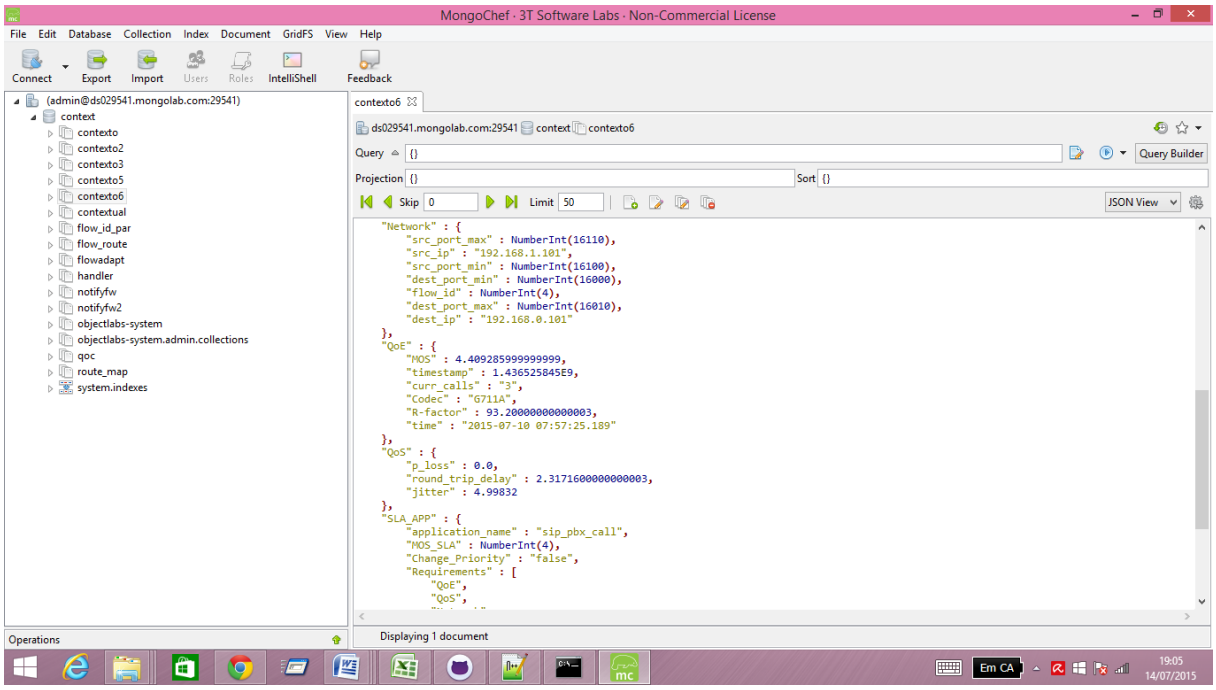
A **Erro! Fonte de referência não encontrada.** apresenta o MongoChef e a primeira tela, onde o MOS apurado é satisfatório e conforme e portanto não gera notificação. Nas demais telas apresentamos a sequência do experimento realizado.

A Tabela 2 apresenta os parâmetros testados.

Tabela 2 - Resultados obtidos em cada passagem

Tamanho da Fila	Chamadas Simultâneas	MOS	Ação
256K	3	4,409	Manter
256K	6	3,106	Notifica Fwd
700K	6	4,304	Manter
700K	12	2,560	Notifica Fwd
1200K	12	4,409	Manter
1200K	18	4,19	Manter

Figura 20 - Tela do MongoChef, software cliente de visualização de banco de dados MongoDB



A **Erro! Fonte de referência não encontrada.**2 apresenta a primeira rodada de testes dando ênfase ao valor de MOS, maior que 4.

Figura 21 - Rodada 1, MOS 4,40 > 4, não gera notificação. 3 chamadas/fila 256


```
    },
    "QoE" : {
        "MOS" : 4.409285999999999,
        "timestamp" : 1.436525845E9,
        "curr_calls" : "3",
        "Codec" : "G711A",
        "R-factor" : 93.20000000000003,
        "time" : "2015-07-10 07:57:25.189"
    },
    "QoS" : {
        "p_loss" : 0.0,
        "round_trip_delay" : 2.3171600000000003,
        "jitter" : 4.99832
    },
    "SLA_APP" : {
        "application_name" : "sip_pbx_call",
        "MOS_SLA" : NumberInt(4),
```


A **Erro! Fonte de referência não encontrada.**3 apresenta a segunda rodada de testes dando ênfase ao valor de MOS, menor que 4.

Figura 22 - Rodada 2, MOS 3,106 < 4, gera notificação, 6 chamadas, /fila256


```
},
"QoE" : {
    "MOS" : 3.1061686666666666,
    "timestamp" : 1.436526309E9,
    "curr_calls" : "6",
    "Codec" : "G711A",
    "R-factor" : 61.38556416666668,
    "time" : "2015-07-10 08:05:09.982"
},
"QoS" : {
    "p_loss" : 20.398098666666666,
    "round_trip_delay" : 2872.8833199999995,
    "jitter" : 10.59796
},
"SLA_APP" : {
    "application_name" : "sip_pbx_call",
    "MOS_SLA" : NumberInt(4),
    "Change_Priority" : "false",
    "Requirements" : [
        "QoE",
        "QoS",
```

A **Erro! Fonte de referência não encontrada.**4 apresenta a terceira rodada de testes dando ênfase ao valor de MOS, maior que 4.

Figura 23 - Rodada 3, MOS 4,304 > 4, não gera notificação. 6 chamadas, fila/700


```
},
"QoE" : {
    "MOS" : 4.304107079999999,
    "timestamp" : 1.436526645E9,
    "curr_calls" : "6,833333333",
    "Codec" : "G711A",
    "R-factor" : 90.53906600000002,
    "time" : "2015-07-10 08:10:45.651"
},
"QoS" : {
    "p_loss" : 0.76826836,
    "round_trip_delay" : 2.8044399999999996,
    "jitter" : 6.3136399999999995
},
"SLA_APP" : {
    "application_name" : "sip_pbx_call",
    "MOS_SLA" : NumberInt(4),
    "Change_Priority" : "false",
    "Requirements" : [
```


A **Erro! Fonte de referência não encontrada.**5 apresenta a quarta rodada de testes dando ênfase ao valor de MOS, menor que 4.

Figura 24 - Rodada 4, MOS 2,560 < 4, gera notificação. 12 chamadas/fila700

```
},
"QoE" : {
    "MOS" : 2.5601433750000004,
    "timestamp" : 1.43652742E9,
    "curr_calls" : "9,444444444",
    "Codec" : "G711A",
    "R-factor" : 49.66154083333334,
    "time" : "2015-07-10 08:23:40.470"
},
"QoS" : {
    "p_loss" : 29.276237916666663,
    "round_trip_delay" : 2139.82196,
    "jitter" : 11.12948
},
"SLA_APP" : {
    "application_name" : "sip_pbx_call",
    "MOS_SLA" : NumberInt(4),
    "Change Priority" : "false",
```


A **Erro! Fonte de referência não encontrada.**6 apresenta a quinta rodada de testes dando ênfase ao valor de MOS, maior de 4.

Figura 25 - Rodada 5, MOS 4.40 > 4, não gera notificação. 12 chamadas/fila1200


```
},
"QoE" : {
    "MOS" : 4.409285999999998,
    "timestamp" : 1.436527505E9,
    "curr_calls" : "9,875",
    "Codec" : "G711A",
    "R-factor" : 93.2,
    "time" : "2015-07-10 08:25:05.422"
},
"QoS" : {
    "p_loss" : 0.0,
    "round_trip_delay" : 3.0147999999999997,
    "jitter" : 3.9161999999999995
},
"SLA_APP" : {
    "application_name" : "sip_pbx_call",
    "MOS_SLA" : NumberInt(4),
```


A **Erro! Fonte de referência não encontrada.**7 apresenta a sexta rodada de testes dando ênfase ao valor de MOS, maior que 4.

Figura 26 - Rodada 6, MOS 4,19 > 4. Não gera notificação. 18 chamadas/fila 1200


```
},
"QoE" : {
    "MOS" : 4.19417332,
    "timestamp" : 1.436527583E9,
    "curr_calls" : "9,875",
    "Codec" : "G711A",
    "R-factor" : 87.90667760000002,
    "time" : "2015-07-10 08:26:23.465"
},
"QoS" : {
    "p_loss" : 1.7685852800000001,
    "round_trip_delay" : 132.69368,
    "jitter" : 7.488
},
"SLA_APP" : {
    "application_name" : "sip_pbx_call",
    "MOS_SLA" : NumberInt(4),
    "Change_Priority" : "false",
```


Este experimento permite visualizar a coleta dos dados, a modificação dos dados de contexto e a geração de notificações (por meio de filas no MongoDB). O módulo *Forwarding*, após o cálculo de adequação, deve escolher dentre as filas

disponíveis, aquela que irá comportar a quantidade de chamadas simultâneas demandada.

Neste caso, pode ser visualizado na **Erro! Fonte de referência não encontrada.** através do parâmetro "*curr_calls*", que o mesmo atinge o valor máximo de 9,875. Os critérios adotados neste caso em particular, determinam a escolha de primeira fila capaz de acomodar cerca de 10 chamadas (arredondando 9,875) para o número mais próximo, com uma margem de 20%. Estes são critérios que devem ser determinados pelo administrador, mas para efeito de demonstração vamos considerar que a nova fila deva acomodar 10 chamadas simultâneas, com mais 20% de folga, ou seja 12 chamadas.

Com base neste número, é calculada a banda requerida. Como o G.711 pode consumir até 80Kbps, no pior caso, por chamada simultânea, chegamos o valor de 12 x 80Kbps, ou seja. 960Kbps.

O sistema procura então a próxima fila capaz de acomodar 960Kbps. Esta fila, tal como configurada no Netfilter é a fila 1200, ou seja 1,2Mbps. A execução da rodada 6, ilustrada pela **Erro! Fonte de referência não encontrada.**, mostra o resultado em termos de MOS da execução da mesma quantidade de chamadas simultâneas realizada na rodada anterior e que havia gerado uma notificação de MOS. Nesta rodada o fator de MOS se recupera a atinge 4,19, não gerando mais notificações.

5.3 CENÁRIO 2

O

CENÁRIO 2 é semelhante ao CENÁRIO 1, no entanto, neste cenário é empregado o Codec G.729, cujo consumo efetivo de largura de banda é cerca de 31.2kbps, de acordo com a Tabela 3 a seguir. De acordo com a Tabela 3, o Codec G.729 apresenta uma taxa de compreensão efetiva bem superior ao G.711, motivo pelo qual é bastante empregado em situações onde a banda seja estreita, notadamente na WAN. Em redes corporativas, contudo, 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 e *Round Trip Delay*.

Tabela 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 está aferida na especificação G.107, que atribui ao uso do Codec G.729 um fator de *impairment*, ou seja, degradação, que é refletia em uma pontuação inferior de MOS em relação ao Codec G.711 (ITU, 2015), mesmo quando a qualidade do link é a ideal em ambos os casos.

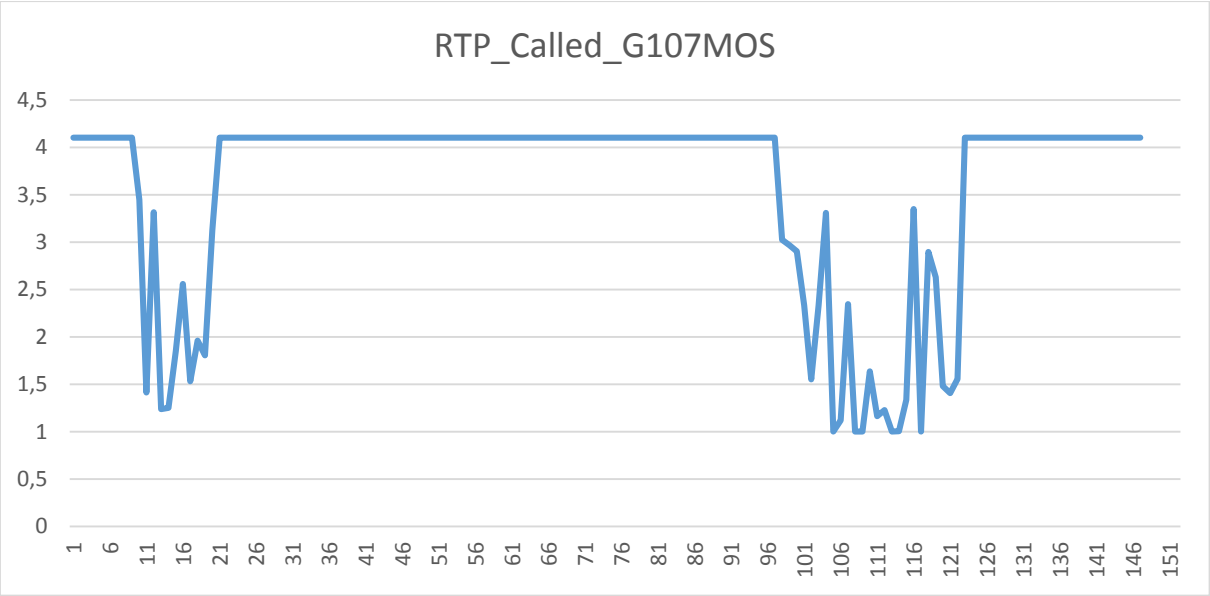
Neste experimento foram realizadas 5 rodadas sucessivas aplicando variações referentes aos parâmetros *quantidade de chamadas simultâneas* e *tamanho de largura de banda*, de acordo com a Tabela 4.

Tabela 4 - Quantidade de banda X Chamadas simultâneas

Banda (Kbps)	Chamadas Simultâneas
128	5
300	5,8,12, 18
512	12, 18

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 *Context Controller* e resultam na notificação ao *Forwarding Module*, que neste cenário requer a mudança de fila.

Figura 27 - Comportamento do sistema e sua recuperação a partir da mudança de fila para uma banda mais larga



Neste caso o primeiro parâmetro alterado foi o Codec G.729, que consome uma banda máxima média em torno de 30Kbps. Na Tabela 4, 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. O *Context Handler* apura esta queda e o *Context Management* sinaliza ao *Forward Adapter*, que por meio de um novo cálculo de banda necessária aponta para a fila de 300Kbps. Nas interações seguintes pode ser observado que o MOS se mantém constante em nível máximo, isto é, 4.1.

Deve também ser notado que no caso do MOS com Codec G.729 o MOS máximo é 4.1, em contraposição ao MOS 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 capacidade de ligações simultâneas mais do que dobra, sobre um mesmo canal, o que pode ser visualizado ao compararmos os gráficos desta passagem com os gráficos da passagem anterior, isto é, aquela que utiliza Codec G.711.

Na interação acima, foram criadas 5 chamadas simultâneas com Codec G.729 sobre uma fila de 128Kbps. Mesmo com baixo consumo, o sistema não foi capaz suportar 5 chamadas com este Codec em G.729. Em seguida são introduzidas 8, 12 e 18 ligações simultâneas. Com 18 ligações simultâneas o *Context Handler* apura uma

nova queda de MOS, o *Context Model Management* aciona o *Forwarding Management*, que por sua vez apura a quantidade de ligações simultâneas em curso e notifica o *Flow Adapter* para atribuir a este fluxo uma fila de banda 700Kbps. Ao realizar a adaptação, o sistema recupera sua capacidade de transitar dados de VoiP com qualidade de MOS máxima, ou seja, 4.1.

Desta forma, visualizamos que ao detectar a perda de MOS, o sistema é sempre capaz de notificar o *Flow Adapter* e este busca uma nova alocação de fila.

5.4 CENÁRIO 3

Neste cenário foi introduzido um elemento de perturbação. Os CENÁRIO 1 e

CENÁRIO 2 refletem as situações onde o canal de voz possui uma banda exclusiva e são demonstrados os casos onde o fator de perturbação é a ocupação excessiva da fila por um excesso de ligações simultâneas.

Basicamente, a diferença entre o CENÁRIO 1 e

CENÁRIO 2 reside na diferença dos *Codecs* e a pontuação de partida do MOS apurado. O

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

Neste cenário será introduzida uma fonte de tráfego de perturbação por meio de um UDP injector (UDP Tester). A sequência de testes é apresentada na Tabela 5.

Tabela 5 - Sequência de testes com injeção de tráfego perturbador

Função	Endereço IP	Porta
UDP Tester	192.168.0.101	13013 – 13014
UAS	192.168.0.102	5061-16XXX
UAC	192.168.1.101	5063-16XXX
Netfilter	192.168.0.100/192.168.1.100	

A cadência dos testes segue a ordem e a sequência, de acordo com a seguinte nomenclatura:

Figura 28 - Fluxo dos testes

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

Figura 29 - Fluxo dos testes com injeção de tráfego UDP

A sequência acima descrita (**Erro! Fonte de referência não encontrada.**) é realizada e os dados apurados pelo *Context Handler* embasam a tomada de decisões do módulo de *forwarding*, conforme demonstrado nos gráficos de degradação e retomada do MOS registrados a seguir.

Figura 30 - Degradação do R-Factor

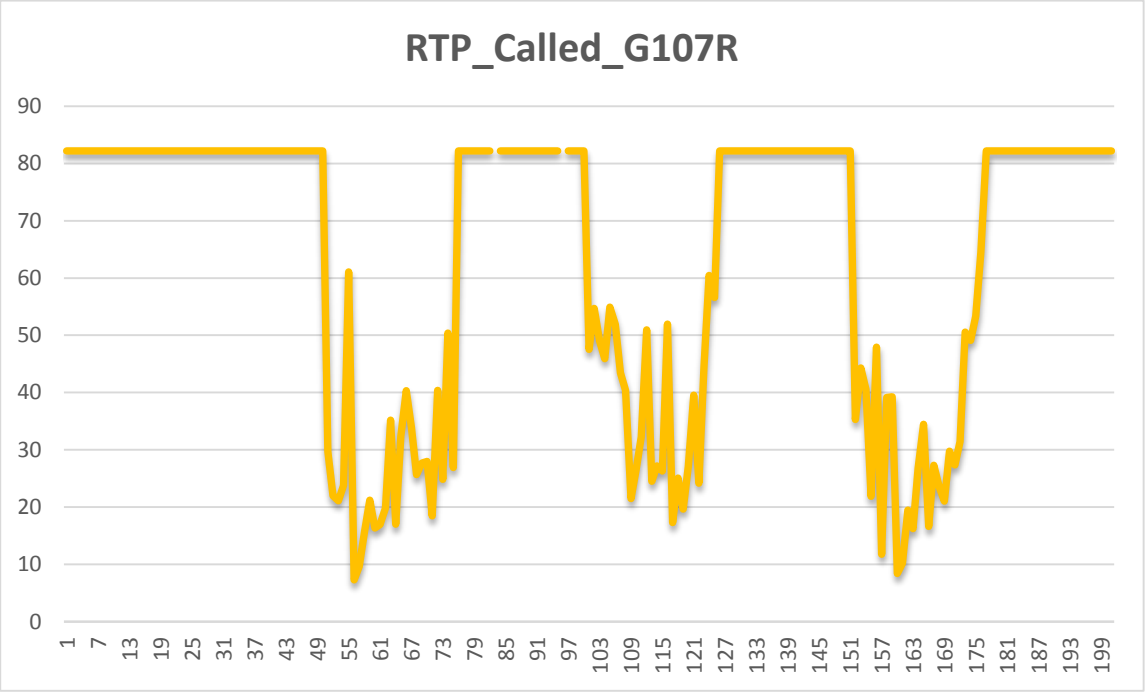
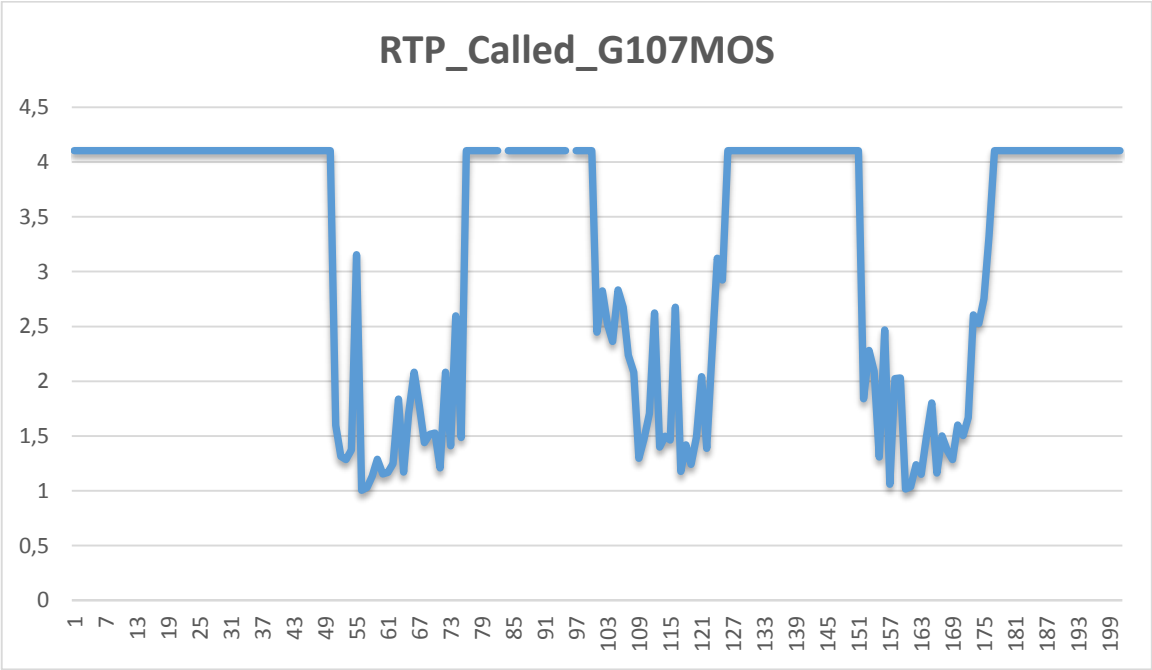


Figura 31 - Degradação do MOS



Os gráficos ilustrados nas Figura 31 - Degradação do MOS

e **Erro! Fonte de referência não encontrada.** apresentam o MOS e o Rfactor, onde visualizamos a correlação entre estes dois fatores. O MOS é derivado do R-Factor, e desta forma, um perfil de tráfego deve acompanhar o outro como pode

ser visualizado acima. Neste cenário, foi introduzido um tráfego perturbador, injetando tráfego UDP concorrente ao tráfego RTP.

A dinâmica deste cenário é similar às anteriores, porém com um novo elemento de elemento de perturbação. Nos gráficos das Figura 31 - Degradação do MOS e **Erro! Fonte de referência não encontrada.** é possível observar que ao final da apuração de um bloco de chamadas, o *Context Management* realiza o cálculo da média do MOS daquele bloco de ligações que embasa a decisão de sinalizar, isto é, notificar o *Flow Adapter* acerca da necessidade de mudança de fila. Nos momentos 75, 125 e 175 é tomada a decisão de alocar uma nova fila de encaminhamento e o MOS é então recuperado.

Neste experimento, a fim de ilustrar o mecanismo de funcionamento de uma forma mais clara, foi estabelecida a apuração da média de 25 ligações. Havendo violação de MOS, ocorre então a tomada de decisão. A violação de MOS não ocorreu, para fins de experimentação, com apenas uma ligação, mas após a apuração de uma média de ligações. Esta decisão é tomada pelo administrador de sistema, que pode definir um período de tolerância para observar a queda de MOS, ou uma média de chamadas. Desta forma, a decisão pode ser tomada pela duração do tempo de violação ou por uma quantidade de chamadas estabelecidas.

5.5 CENÁRIO 4

O CENÁRIO 4 é similar ao CENÁRIO 3, contudo as regras aplicadas ao gerenciamento de QoE são efetivadas através do controle de portas. O *Forwarding Module* recebe, tal como nos cenários anteriores, as notificações emitidas pelo *Context Controller*. As decisões de adaptação do fluxo de chamadas são efetivadas neste experimento com base nos seguintes parâmetros adquiridos pelo *Context Controller*:

- Quantidade de Chamadas Concorrentes apuradas para um dado fluxo;
- Codec utilizado;
- Portas utilizadas;
- Protocolo.

Para tanto, no *Iptables* foi usado o módulo “**multiport**” que permite que sejam especificadas múltiplas portas para um alvo. Podem ser especificadas até 15 portas

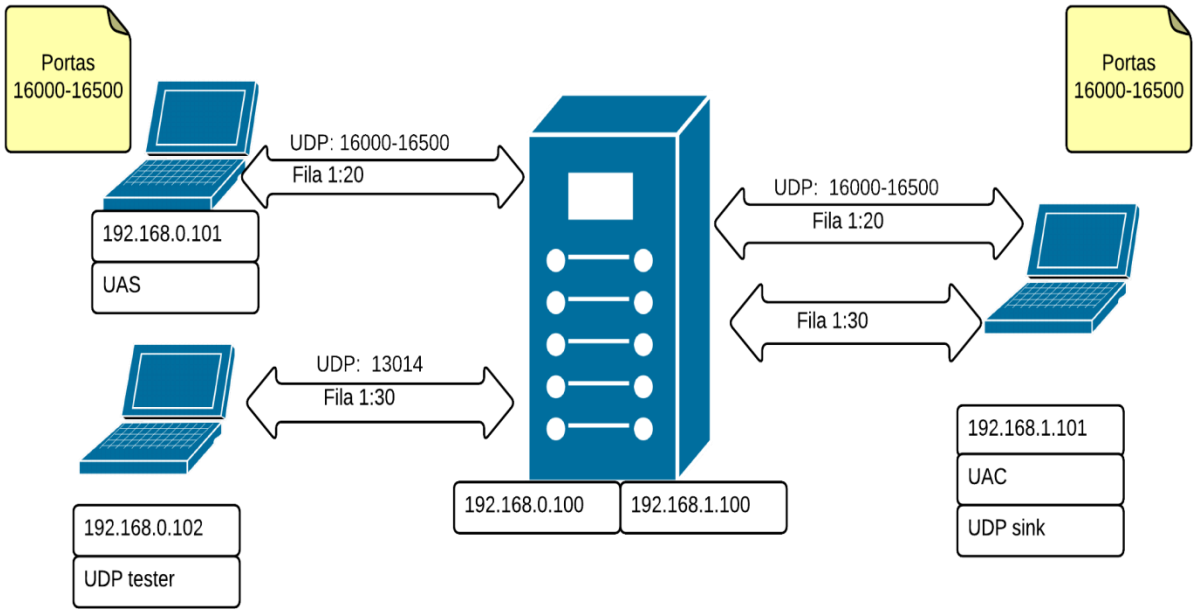
em um único parâmetro e basta que uma porta seja validada para que a regra entre em ação, pois a comparação é feita usando condições "**or**".

O parâmetro “**multiport”** deve ser acompanhado de um dos argumentos abaixo:

- --source-port [porta1, porta2...] - Permite que a regra confira se a porta de origem estiver presente entre as portas especificadas;
- --destination-port [porta1, porta2...] - Permite que a regra confira se a porta de destino estiver presente entre as portas especificadas, e;
- --port [porta1, porta2...] – Permite que regra confira se a porta de origem ou destino estiver presente no parâmetro.

Este módulo pode eliminar muitas regras de firewall que fazem o mesmo tratamento de pacotes para diversas portas diferentes.

Figura 32 - Arquitetura utilizada no Cenário 4



A **Erro! Fonte de referência não encontrada.**, ilustra a arquitetura de testes usada na realização do CENÁRIO 4, deixando claro a separação de filas, enfatizando o tratamento de portas dos serviços.

Os gráficos ilustrados nas

Figura 34 - Eixo x, rodadas. Eixo Y, degradação e recuperação de MOS

,

Figura 35 - Perda de pacotes

e **Erro! Fonte de referência não encontrada.** demonstram o comportamento do sistema a cada rodada de ligações, e o respectivo detalhamento do R-Factor, MOS e um parâmetro de QoS especificamente orientado a este experimento, referente à taxa de perda de pacotes (p_loss).

Figura 33 - Eixo X, rodadas, Eixo Y, degradação e recuperação do R-Factor

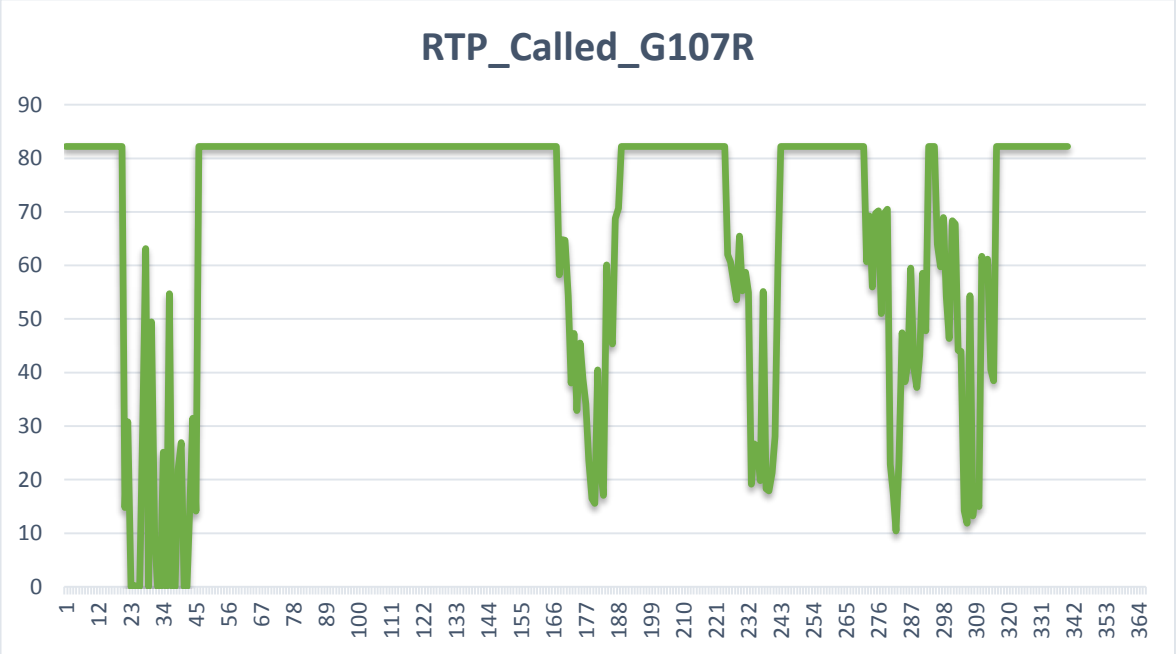


Figura 34 - Eixo x, rodadas. Eixo Y, degradação e recuperação de MOS

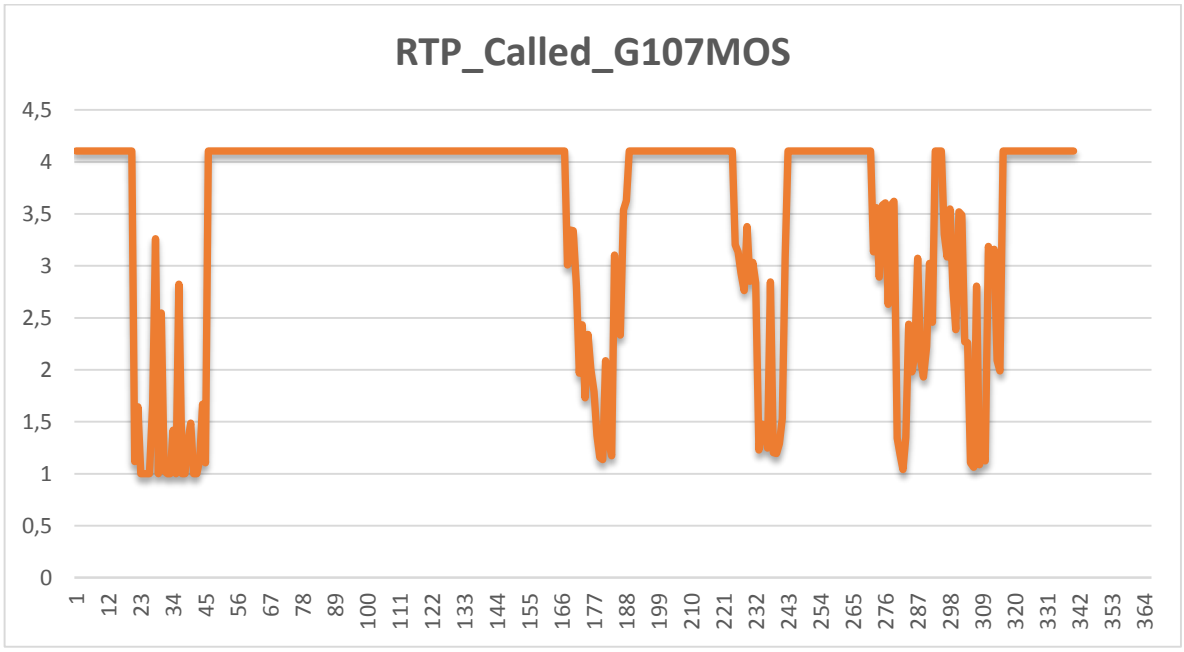
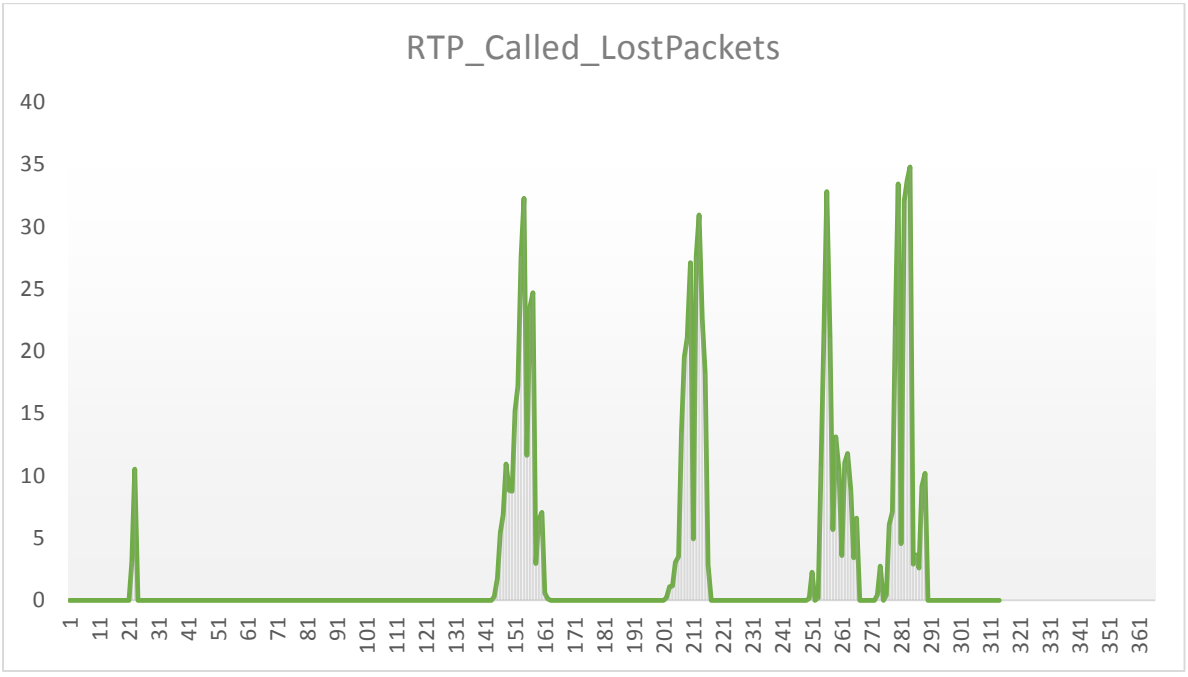


Figura 35 - Perda de pacotes



Os parâmetros apurados pelo *Context Controller* são confrontados com a base de rotas disponíveis, que no caso do *Context Forwarding*, neste experimento, é emulado pelas filas do Netfilter. Ao final do processamento dos dados de contexto, um comando *iptables* é emitido se houver violação de MOS, caso contrário, nada acontece. Dessa forma, fica demonstrada a eficiência do controle por portas sip e rtp, conforme discutido a seguir.

As

Figura 34 - Eixo x, rodadas. Eixo Y, degradação e recuperação de MOS

e

Figura 35 - Perda de pacotes

mostram a dinâmica dos fatores MOS e *R-factor* apurados. Uma vantagem desta abordagem é que apurar este único parâmetro torna mais simples, eficaz e rápido determinar os momentos de degradação. A política aqui adotada segue aquelas adotadas anteriormente, i.e, são apurados blocos de 25 ligações e calculada a média do MOS em cada bloco. O mesmo é realizado para R-factor. Desta forma, a um período de degradação de MOS/R-Factor segue-se sempre uma retomada do nível de MOS conforme a métrica definida de acordo com o valor limite inferido, i.e, (4.0) para MOS. Todos os instantes de subida e recuperação que podem ser visualizados nos gráficos acima estão associados com uma tomada de decisão de mudança de fila.

Já a **Erro! Fonte de referência não encontrada.** ilustra o forte acoplamento existente entre a perda de pacotes e a degradação da MOS e R-Factor.

Na **Erro! Fonte de referência não encontrada.**, o sistema utiliza inicialmente a fila 0 referente a 256Kbps, ressaltada no código. Após a constatação de violação de MOS, o sistema de *forwarding* assume uma nova fila, fila 1, de 512Kbps, que é suficiente para acomodar o tráfego referente à extensão de portas UDP para tráfego VoIP. Sucessivos ajustes de filas, a fim de acomodar a quantidade de chamadas simultâneas, são realizados em função da degradação do MOS/F-factor.

Figura 36 - Representação das filas no banco de dados MongoDB

```
{
    "_id" : ObjectId("558851cae4b04eaf19cff593"),
    "flow_id" : NumberInt(4),
    "path" : {
        "pathid" : NumberInt(4),
        "fwrid_ini" : NumberInt(4),
        "fwrid_fin" : NumberInt(4),
        "hop_count" : NumberInt(1)
    },
    "link" : {
        "linkid" : "int",
        "fwrid_ini" : "int",
        "fwrid_fin" : "int",
        "status" : "string"
    },
    "link_parameter" : {
        "linkid" : "int",
        "parm_id" : "int",
        "curvalue" : "int"
    },
    "pathlink" : [
        {
            "pathid" : "int",
            "linkid" : "int"
        }
    ],
    "queue" : [
        {
            "queueid" : NumberInt(0),
            "description" : NumberInt(256),
            "class" : "1:10"
        },
        {
            "queueid" : NumberInt(1),
            "description" : NumberInt(512),
            "class" : "1:20"
        },
        {
            "queueid" : NumberInt(2),
            "description" : NumberInt(700),
            "class" : "1:30"
        },
        {
            "queueid" : NumberInt(3),
            "description" : NumberInt(1024),
            "class" : "1:40"
        },
        {
            "queueid" : NumberInt(4),
            "description" : NumberInt(1250),
            "class" : "1:50"
        },
        {
            "queueid" : NumberInt(5),
            "description" : NumberInt(1500),
            "class" : "1:60"
        },
        {
            "queueid" : NumberInt(6),
            "description" : NumberInt(2048),
            "class" : "1:70"
        }
    ],
    "eventforwarding" : {
        "pathid" : NumberInt(4),
        "queueid" : NumberInt(1),
        "evtfrwid" : NumberInt(1),
        "evtid" : NumberInt(1)
    }
}
```

Na **Erro! Fonte de referência não encontrada.** são ilustradas as filas, indicadas no subdocumento pelo elemento “queue”:{(..)}.

No instante inicial, a fila alocada no *Iptables* e representada no MongoDB, possui um número de identificação (0), uma descrição (256), que nos permite indicar a capacidade de banda desta fila. Neste caso trata-se de uma fila de 25Kbps

associada a classe (1:10) definida no *Iptables*. Já o campo “eventforwarding”:{(...)} indica a ocorrência de um evento relacionado à mudança do queueid.

Por fim, esta decisão será traduzida por um comando em Python no *Iptables*, como ilustrado na **Erro! Fonte de referência não encontrada.**.

Figura 37 - Comando Python-Iptables reprogramando as filas do Netfilter

```
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)
```

Após a aplicação do comando, o campo *status* de *eventforwarding* é reinicializado e o *Forwarding Controller* prossegue seu atendimento a notificações. Esta última diretiva atribui uma nova fila ao tráfego cujas portas origem estejam na faixa 16000-16500 (ou outra faixa estabelecida) permitindo uma melhoria de MOS. Em todos os momentos em que se pode verificar a ascensão vertical no perfil das linhas dos gráficos referentes às

Figura 34 - Eixo x, rodadas. Eixo Y, degradação e recuperação de MOS

e

Figura 35 - Perda de pacotes

, ocorreu um envio de comando python conforme **Erro! Fonte de referência não encontrada.**.

5.6 Cenário 5

No

Cenário 5, o Codec G.711 foi empregado. A seguir, a captura de tela referente ao arquivo “*firewall*” é apresentada na **Erro! Fonte de referência não encontrada.**, definindo as filas empregadas. As rodadas aplicadas neste caso foram (Tabela 6):

Tabela 6 - Ciclo de testes do Cenário 5

Tamanho da fila (Kbps)	Chamadas simultâneas	Notificação	Fila no Netfilter
300	3	N	1:20
300	6	S	1:20
512	6	N	1:30
512	12	S	1:30
1024	12	N	1:50
1024	18	S	1:50
1500	18	N	1:60

Figura 38 - Parte do script de Firewall, enfatizando a configuração das filas

```
#!/bin/bash

interface=eth1
velol=10mbit

function iniciar()
{
        #Define a classe root
        tc qdisc add dev $interface root handle 1:0 htb default 90

        #Define a taxa total do link da interface
        tc class add dev $interface parent 1:0 classid 1:1 htb rate $velol

        #Divide o link total da placa em partes
        tc class add dev $interface parent 1:1 classid 1:10 htb rate 300kbit ceil 350kbit
        tc class add dev $interface parent 1:1 classid 1:20 htb rate 300kbit ceil 350kbit
        tc class add dev $interface parent 1:1 classid 1:30 htb rate 512kbit ceil 550kbit
        tc class add dev $interface parent 1:1 classid 1:40 htb rate 700Kbit ceil 750kbit
        tc class add dev $interface parent 1:1 classid 1:50 htb rate 1024Kbit ceil 1096Kbit
        tc class add dev $interface parent 1:1 classid 1:60 htb rate 1500kbit ceil 1550kbit
        tc class add dev $interface parent 1:1 classid 1:70 htb rate 128kbit ceil 148Kbit

### REGRAS BASEADAS EM PORTAS ###
iptables -t mangle -A FORWARD -o $interface -p udp -m multiport --sport 5060:5070 -j CLASSIFY --set-class 1:10
iptables -t mangle -A FORWARD -o $interface -p udp -m multiport --sport 16000:16500 -j CLASSIFY --set-class 1:60
### REGRAS PARA AS PORTAS DO UDP TESTER, TRAFFIC INJECTION ###
iptables -t mangle -A FORWARD -o $interface -p udp -m multiport --sport 13013:13014 -j CLASSIFY --set-class 1:70
}
```

A seguir, as

Figura 40 - Variação do MOS com Codec G107

,

Figura 41 - Variação do *Round Trip Delay*

,

Figura 42 - Variação do *Jitter*

,

Figura 43 - Variação da perda de pacotes

e **Erro! Fonte de referência não encontrada.** comprovam a eficácia do Cenário 5 e do experimento como um todo.

Figura 39 - Variação do R-Factor com Codec G107

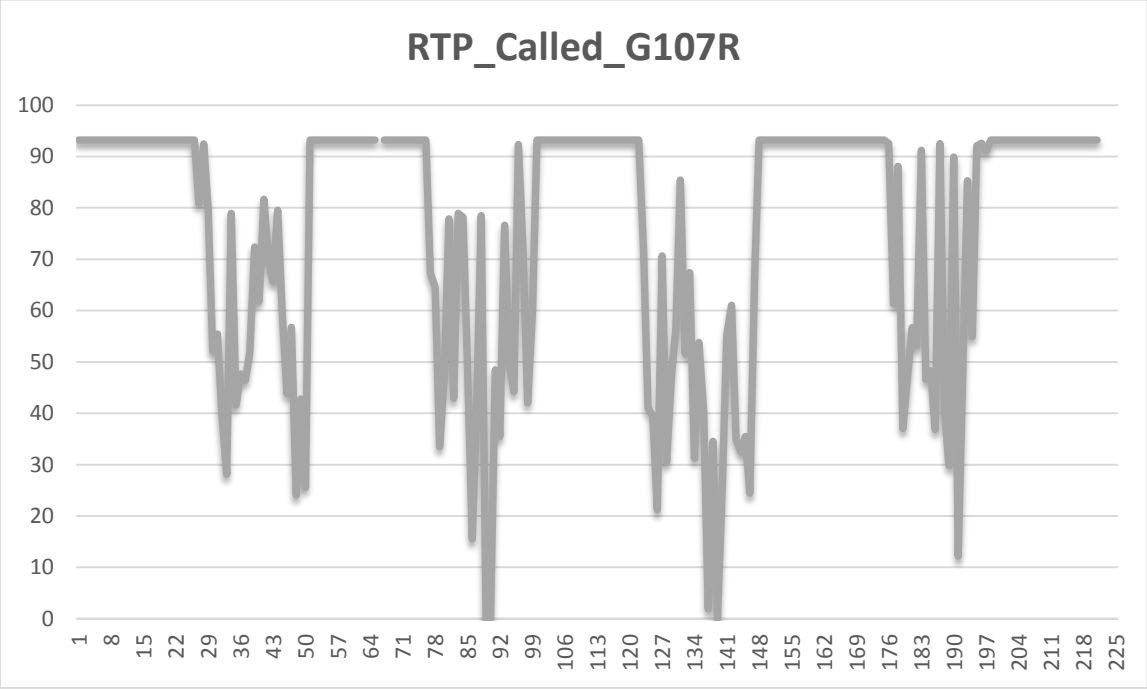


Figura 40 - Variação do MOS com Codec G107

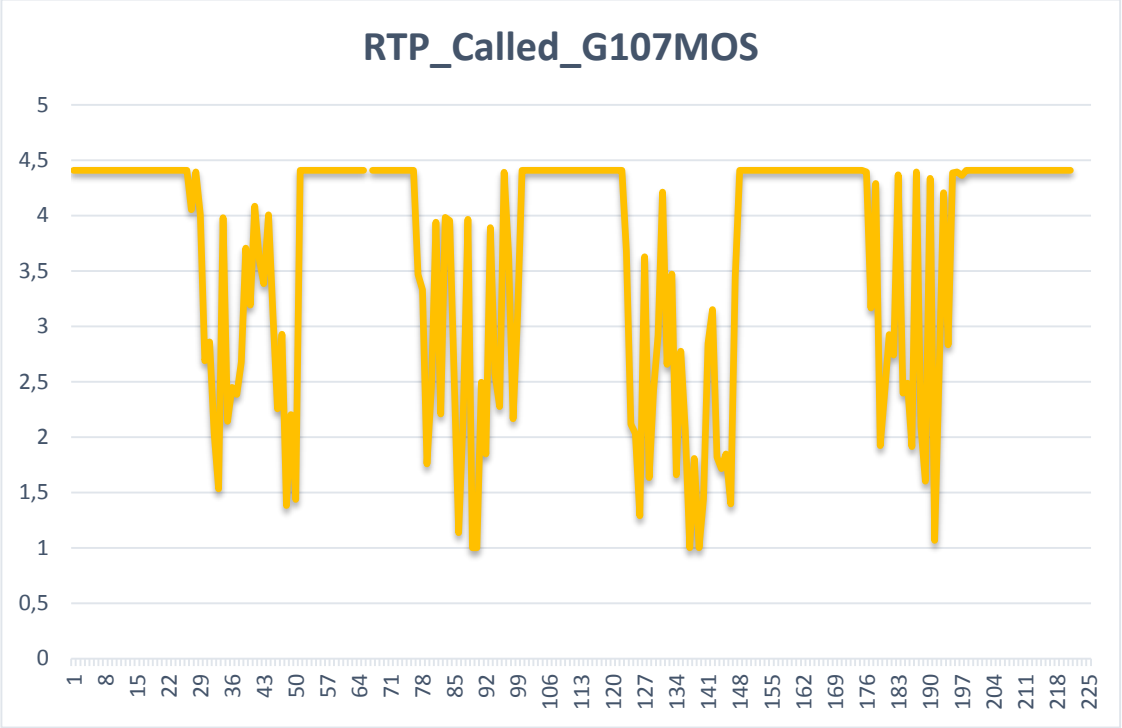


Figura 41 - Variação do *Round Trip Delay*

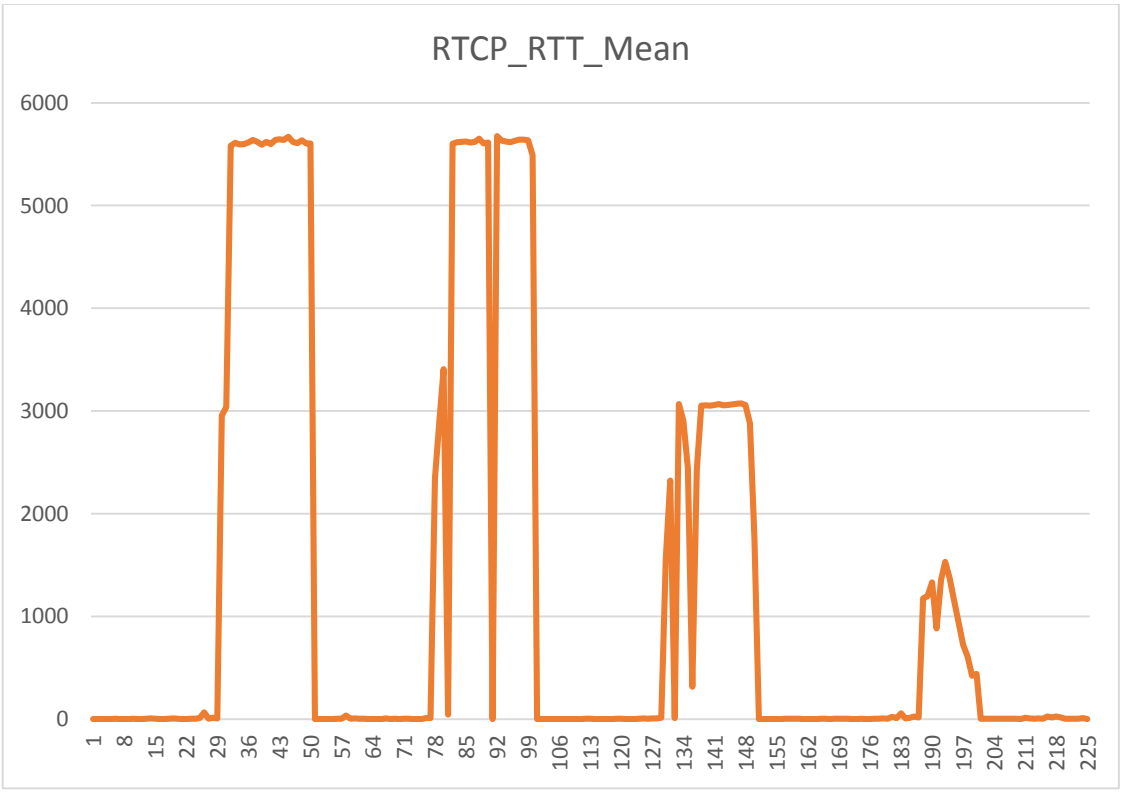


Figura 42 - Variação do *Jitter*

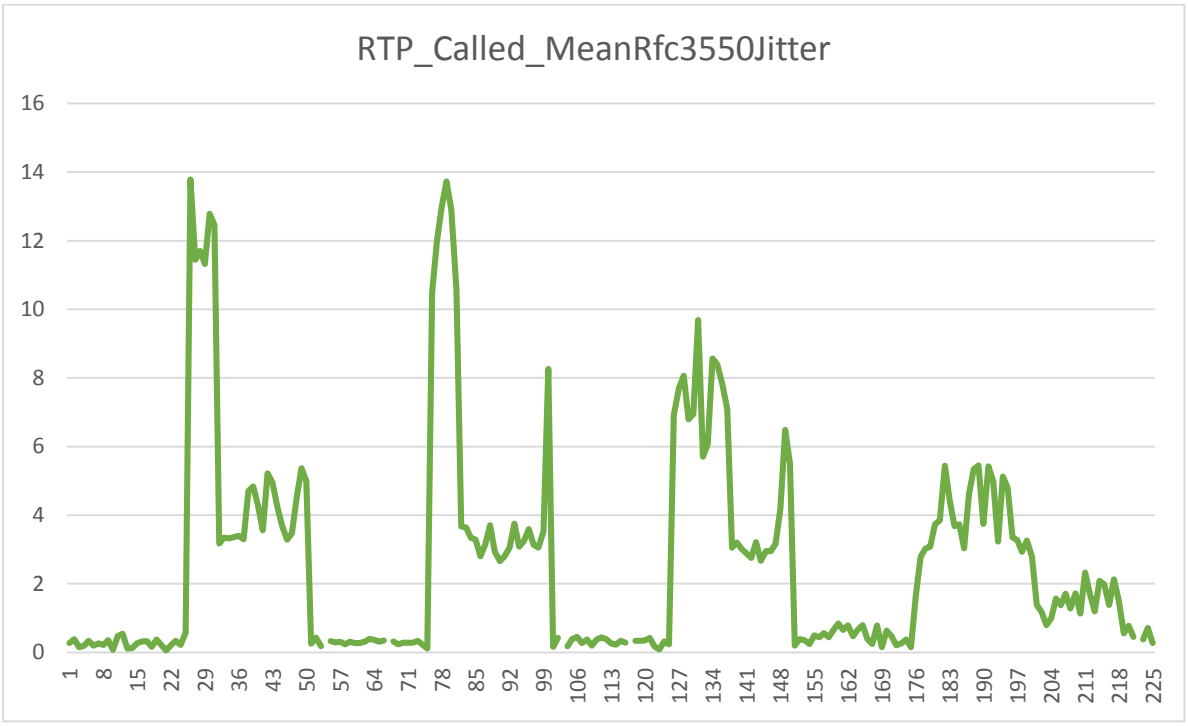
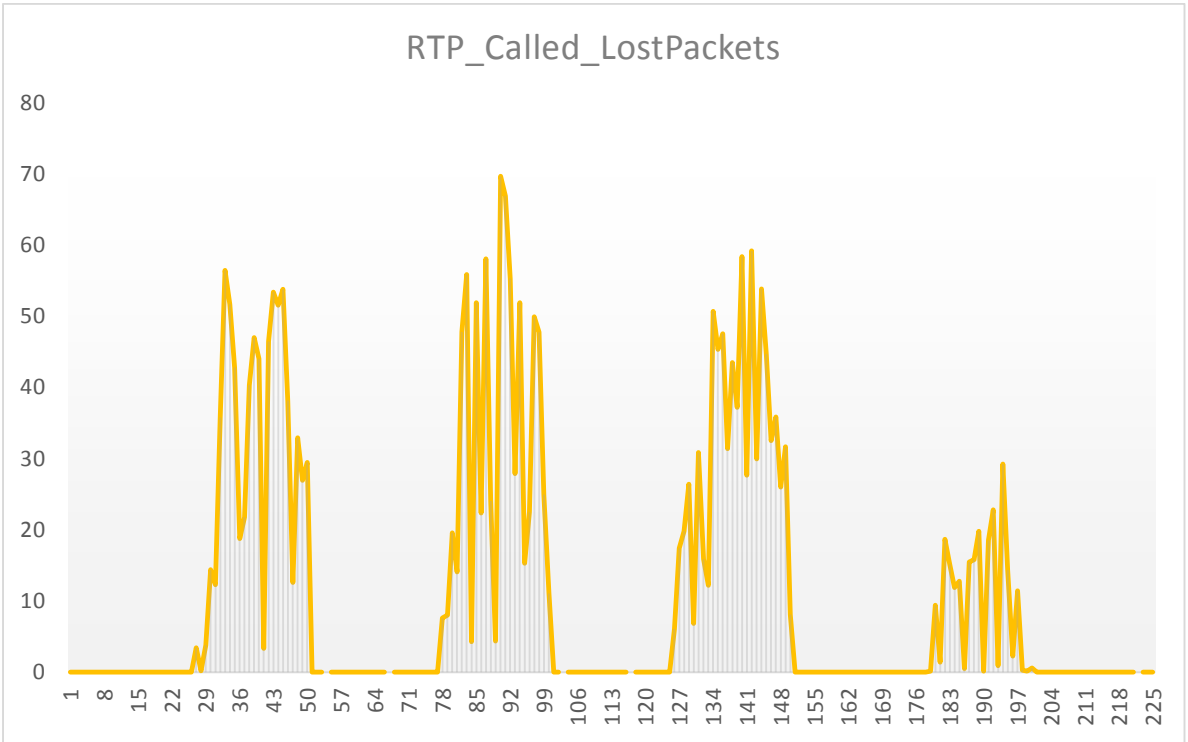


Figura 43 - Variação da perda de pacotes



As

Figura 40 - Variação do MOS com Codec G107

a **Erro! Fonte de referência não encontrada.** apresentam os dados coletados referentes respectivamente a *R-Factor*, MOS, *Round Trip Delay, Jitter* e *Packet loss* para uma mesma série de ligações. O parâmetro *R-Factor* é calculado a partir dos demais fatores, do qual é derivado o MOS. A disposição dos gráficos favorece a análise imediata da correlação direta entre os fatores de degradação destes quesitos de QoS e sua influência sobre o MOS.

Sabe-se que o parâmetro *Round Trip Delay,* é o atraso do pacote no caminho de ida e volta (origem - destino). O parâmetro *Packet Loss* representa a perda de pacotes que ocorre no caminho, no caso, na fila de encaminhamento no NetFilter, e o *Jitter* refere-se à variação do atraso. Pode ser verificada claramente a correlação inversa entre estes fatores. O MOS também é derivado do tipo de Codec utilizado. Além disso, o parâmetro *Round Trip Delay* é obtido a partir dos parâmetros coletados no protocolo RTCP. As métricas acima foram obtidas sobre o comportamento do tráfego RTP (*media stream*).

As figuras ilustram que tão logo ocorre a aplicação da regra de encaminhamento, ou seja, o cálculo de banda necessária para acomodar a nova situação de fluxo, e sua aplicação no Netfilter, ocorre uma subida vertical no gráfico, refletindo a pronta recuperação da qualidade do sistema.

5.7 DISCUSSÕES

Os experimentos realizados no capítulo 5 demonstram a efetividade da solução proposta. Especificamente no caso analisado, foi possível obter a recuperação do desempenho do sistema sempre que foi apurada a queda de qualidade da conexão (pela avaliação da queda de qualidade da ligação). Neste caso, esta queda foi apurada por parâmetros de MOS, mas poderia ter sido apurada também por meio da leitura de outros parâmetros, como perda de pacotes *e round trip delay*, isto é, parâmetros de rede, ou parâmetros fornecidos pelo próprio usuário.

Ficou demonstrado que o *Flow Adapter* recebeu as notificações, e foi capaz de se comunicar com o *Context Controller*, por meio de notificações emitidas por uma fila de prioridades de notificação. Sempre que recebeu as notificações, o módulo de encaminhamento procedeu com a recuperação da qualidade do sistema.

Nesta experimentação esta recuperação se deu pelo aumento da capacidade das filas do Netfilter, por meio de comandos enviados ao *Iptables*. Sempre que ocorreu esta sequência de operações, após a apuração realizada pelo *Context Management*, o módulo de encaminhamento foi capaz de consultar a base de dados de contexto, localizada na nuvem, e realizar os cálculos necessários para alocar uma nova banda de comunicação. As medições subsequentes mostraram a recuperação do valor de MOS em todos os casos.

A integração com o *Context Controller* se deu por meio de aplicações e bases de informação distribuídas. Uma das abordagens do trabalho diz respeito à possibilidade de este sistema (o *Context Controller*) ser uma Gerência geral acima de outras gerências. Considerando que existem diversos tipos de gerenciamento e controle de redes, já implementados em diversas plataformas de monitoramento e em variados níveis de profundidade.

5.8 CONCLUSÃO

Neste capitulo foi comprovado que o sistema implementado é capaz de receber notificações a partir de um módulo de coleta, gerência e inferência, e tomar decisões de roteamento acerca de uma otimização que possa ser encaminhada para um nível inferior da estrutura.

No caso específico da implementação por filas, os experimentos transcorreram apresentando resultados efetivos, possibilitando a viabilidade de atuação em outras estruturas.

O sistema funciona tanto como um atuador direto (execução do comando *Iptables*) ou como um gestor (sobre estruturas mais complexas existentes). A forma de implementação adotada é extremamente flexível e de altíssima legibilidade. Estas facilidades objetivam dotar os futuros pesquisadores de maior produtividade e rapidez de aprendizado na solução proposta. Um dos maiores motivos para o fracasso na continuidade de projetos mais complexos se dá justamente na transição entre um projetista/pesquisador e outro. Esta foi uma questão que foi dada especial atenção neste trabalho, visando uma continuidade permanente.

6 CONCLUSÕES E PERSPECTIVAS FUTURAS

Nesta dissertação foi apresentada uma solução que objetiva a implementação de uma arquitetura para o arcabouço *Context-Aware Adaptative Routing Framework* (CAARF), mais especificamente do módulo funcional de gerência de encaminhamento.

Nesta seção são apresentadas as conclusões gerais sobre o trabalho realizado, onde são contemplados um resumo do trabalho realizado, as principais contribuições relativas a esta dissertação e algumas características da solução proposta. Para finalizar, são apresentadas sugestões para continuidade deste trabalho.

6.1 CONCLUSÕES

Esta dissertação propôs a discussão da eficácia da aplicação integrada de Qualidade de Experiência (QoE), Qualidade de Serviço (QoS) e Qualidade de Dispositivo (QoD), como solução para viabilizar o encaminhamento otimizado de tráfego sensível ao tempo relacionados às aplicações convergentes.

De forma a proporcionar uma melhor experiência do usuário dessas aplicações, foram definidos, um modelo e um arcabouço para o tratamento de contexto denominado, respectivamente, *Context Model* e *Context-Aware Adaptative Routing Framework (CAARF)*. O principal objetivo do arcabouço CAARF é apoiar as tomadas de decisões de encaminhamento de dados baseados em contexto. Com a definição das regras de encaminhamento, é possível reavaliar, de tempos em tempos, se os caminhos definidos para determinados eventos são os mais adequados, proporcionando assim uma melhor condição de uso da rede por parte dos usuários em geral.

Para tanto, a implementação de uma arquitetura, que utilize estas regras de encaminhamento pré-definidas, permitiu comprovar a eficácia do arcabouço em diferentes abordagens, neste caso em particular no tratamento de filas junto ao módulo Netfilter. Com isso, foi possível validar as estratégias utilizadas para obtenção de melhor desempenho para o encaminhamento baseado em contexto. No entanto, devido à impossibilidade de coletar eventos de Qualidade de Dispositivo através de um *middleware* que pudesse interagir com os sensores dos dispositivos

(equipamentos de comunicação reais), a validação deste trabalho foi realizada apenas através da integração dos parâmetros de QoS e QoE.

Dessa forma, alguns objetivos principais foram definidos e seguidos ao longo deste trabalho. Um deles era determinar se seria possível obter com eficácia a resolução do encaminhamento adaptativo a partir de um único parâmetro, neste caso o MOS. Este tipo de abordagem permitiu integrar os módulos de aquisição e coleta com um grande número de aplicações já existentes. Os resultados obtidos nos experimentos respaldaram os objetivos iniciais estabelecidos. Este enfoque permite também incorporar os dados de contexto referente à percepção do usuário, que também pode ser traduzido em MOS. Caso as informações baseadas em QoE não estiverem presentes, usa como considera-se como base os dados de QoS, como *jitter*, *packet loss* e RTT, caso contrário a tomada de decisão é feita preferencialmente pelo QoE.

Os estudos de caso apresentados neste trabalho são baseados em uma rede particular, descrita através de cenários que contemplam características de um ambiente real. Durante os experimentos realizados, de forma a dar ênfase ao módulo de encaminhamento, foi considerado que o contexto do ambiente computacional já tinha sido modificado anteriormente. Cabendo a esse módulo apenas a aplicação das regras de encaminhamento para a definição dos melhores caminhos e a modificação das tabelas de fluxos (através da alocação das respectivas filas de encaminhamento).

Os cenários implementados evoluíram de forma incremental através de acréscimos aos experimentos, incluindo variáveis de contexto e percepções dos usuários mais detalhadas. Com isso, é possível aumentar a quantidade de parâmetros adquiridos pelo agente *middleware*, módulo que está atualmente em desenvolvimento, e será integrado como fornecedor de variáveis de contexto mais ricos, como a localização e dados relacionados com a qualidade dos dispositivos.

Da mesma forma, também foi comprovada a eficácia da utilização de um sistema para a representação dos dados de contexto baseado em NoSQL. Isto é, por meio de documentos, o que torna este sistema extremamente flexível, permitindo adicionar indefinidamente novos parâmetros e variáveis de contexto. Esta expressividade habilitará os futuros desenvolvedores a adicionar mais facilmente extensões a este trabalho.

Na seção a seguir, temos as perspectivas futuras para continuação desse trabalho apresentado.

6.2 TRABALHOS FUTUROS

Esta seção identifica alguns trabalhos futuros tanto para dar continuidade à pesquisa aqui apresentada, de forma a propor alternativas às limitações citadas. Segue uma lista não restritiva de sugestões para trabalhos futuros:

- O modelo de contexto proposto e arquitetura devem ser validados através da execução de cenários diferentes e mais complexos;
- Definição e implementação de protocolos de encaminhamento sensíveis ao contexto, baseados no modelo de contexto proposto;
- Extensão ao *midleware* em desenvolvimento, com vários tipos de coletores diferentes;
- Utilização de diferentes tipos de aplicações, permitindo a coleta de diferentes parâmetros de contexto;
- Implementação e validação da noção de Qualidade de dispositivo, permitindo a representação e validação de cenários com diferentes métricas, como por exemplo, aplicações médicas voltadas ao monitoramento de pacientes, e;
- Aplicação da solução proposta à cenários de redes SDN, com construções de adaptadores de fluxo dinâmicos, de forma simulado ou em uma rede real.

REFERÊNCIAS

AGBOMA, F.; LIOTTA, A. QoE-aware QoS Management. In: INT. CONF. ON ADVANCES IN MOBILE COMPUTING AND MULTIMEDIA, 6., 2008. **Proceedings…** 2008.

ALAHMARI, S.; ZALUSKA, E.; ROURE, D. C. D. A Metrics Framework for Evaluating SOA Service Granularity. In: IEEE INTERNATIONAL CONFERENCE ON SERVICES COMPUTING, 2011, Washington.. **Proceedings…** Washington, DC, USA: IEEE Computer Society, 2011. (SCC '11), p. 512–519. ISBN 978-0-7695-4462-5. Disponível em: <http://dx.doi.org/10.1109/SCC-.2011.98>. Acesso em: 20 jan. 2012.

AL-ALI, R. J. et al. **G-QoSM**: Grid Service Discovery Using QoS Properties. Computers and Artificial Intelligence 21. [S.l.]: [s.n.], 2002.

ALRESHOODI, M.; WOODS, J. **Survey on QoE\QoS Correlation Models for Multimedia Services**. arXiv preprint arXiv:1306.0221. [S.l.]: [s.n.], 2013.

APACHE. 2015. Disponivel em: <http://apache.org>. Acesso em: 28 ago. 2015.

AWDUCHE, D. et al. Requirements for Traffic Engineering over MPLS. **Internet RFC,** n. 2702, sep.1999.

AWS. 2015. Disponível em: <https://aws.amazon.com/pt>. Acesso em: 28 ago. 2015.

AZURE. 2015. Disponível em: <http://azure.microsoft.com/pt-br>. Acesso em: 28 ago. 2015.

BLACK, D. An Architecture for Differentiated Services. **Internet RFC,** n. 2475, dec. 1998.

BRADEN, R.; CLARK, D. ; SHENKER, S. Integrated Services in the Internet Architecture: an Overview. **Internet RFC,** n. 1633, jun 1994.

BUCHHOLZ, T.; KÜPPER, A.; SCHIFFERS, M. Quality of Context: What It Is and Why We Need It. In: INTERNATIONAL WORKSHOP OF THE HP OPENVIEW UNIVERSITY ASSOCIATION (HPOVUA), 10., 2003. **Proceedings...** p.1-14, 2003.

BUCHHOLZ, T.; SCHIFFERS, M. Quality of Context: What It Is And Why We Need It. In: WORKSHOP OF THE OPEN VIEW UNIVERSITY ASSOCIATION (OVUA'03), 10., 2003. **Paper**… 2003.

CHEN, G.; KOTZ, D. **A Survey of Context-Aware Mobile Computing Research**. Hanover, NH, USA.; [s.n.], 2000.

CHEN, K. T. et al. Quadrant of Euphoria: a Crowdsourcing Platform for QoE Assessment. **Network, IEEE**, v.24, n.2, p.28-35, 2010.

CHOWDHURY, R.; BHANDARKAR, P.; PARASHAR, M. Adaptive QoS Management for Collaboration in Heterogeneous Environments. Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS 2002), pp. 90-100 communication ecosystem. **Communications Magazine**, IEEE, v.50, n.4), p.58-65. 2002.

CISCO. 2015. Disponível em: <http://www.cisco.com/c/en/us/td/docs/net_mgmt/ip_solution_center/60/traffic_management/user/guide/tem.html>. Acesso em: 28 ago. 2015.

CLOUDMARKET. 2015. Disponível em: <https://www.cloudmark.com/>. Acesso em: 28 ago. 2015.

DEORA, V. et al. A Quality of Service Management framework Based on User Expectations. In: INTERNACIONAL CONFERENCE ON SERVICE ORIENTED COMPUTING (ICSOC03), 5., 2003. **Proceedings…** Springer, Heidelberg, 2003.

DEY, A. K. **Providing Architectural Support for Building Context-Aware Applications.** Georgia: Georgia Institute of Technology, 2000.

DEY, A. K.; ABOWD, G. D.; SALBER, D. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. **Journal Human-Computer Interaction archive**, v.16, n.2, p.97-166, 2001.

El-GENDY, M. A.; BOSE, A.; SHIN, K.G. Evolution of the Internet QoS and Support for Soft Real-Time Applications. **Proceedings of the IEEE,** v.91, n. 7, 2003.

EMMANOUILIDIS, C.; KOUTSIAMANIS, R.-A.; TASIDOU, A. Mobile guides: Taxonomy of architectures, context awareness, technologies and applications. **Journal of Network and Computer Applications**, 2012.

FISCHER, S.; KAMMENHUBER, N.; FELDMANN, AREPLEX: Dynamic Traffic Engineering Based on Wardrop Routing Policies. In: ACM CONEXT CONFERENCE, 2006. **Proceedings**… 2006

GAMMA, E. et al. **Design Patterns**: Elements of Reusable Object-Oriented Software. Boston, MA, USA: Addison-Wesley, 1995.

GLAZER, D. W.; TROPPER, C. A new metric for dynamic routing algorithms. **IEEE Transactions on Communications**, v.38, n.3, mar. 1990.

GOOGLE. 2015. Disponível em: <http://google.com>. Acesso em: 28 ago. 2015.

HONG, D. W.; HONG, C. S. A QoS Management Framework for Distributed Multimidea System. **Int J. Network Mgmt,** n.13, p.115–127, 2003; (DOI: 10.1002/nem.465).

HOßFELD, T.; TRAN-GIA, P.; FIEDLER, M. Quantification of quality of experience for edge-based applications. In: MANAGING Traffic Performance in Converged Networks (pp. 361-373). Springer Berlin Heidelberg, 2007.

HTB, 2015. Disponivel em: <http://luxik.cdi.cz/~devik/qos/htb/manual/userg.htm>. Acesso em: 28 ago. 2015.

ITUT.I.350 - INTERNATIONAL TELECOMMUNICATIONS UNION. General Aspects of Quality of Service and Network Performance In: DIGITAL networks, Including ISDNs". Recommendation I. 350, Revision 1. [S.l.]: [s.n.], nov. 1993.

JOHNSON, R. E. Frameworks = (components + patterns). **Comm. of the ACM**, v. 40, n.10, p.39–42, 1997.

JSON. 2015. Disponível em: <https://www.mongodb.org/json_and_bson>. Acesso em: 7 ago. 2015.

KANDULA, S. et al. Walking the tightrope: Responsive yet stable traffic engineering. **ACM SIGCOMM Computer Communication Review,** v. 35, n. 4, p. 253-264,aug. 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, Y.; LEE, K. A Quality Measurement Method of Context Information in Ubiquitous Environments. In: INTERNATIONAL CONFERENCE ON HYBRID INFORMATION. 2006. **Proceedings…** 2006.

KRAUSE, M.; HOCHSTATTER, I. 7 Challenges in Modelling and Using Quality of Context (QoC). Mobility Aware Technologies and Applications: **LNCS,** n.3744, p.324–333, 2005.

KUROSE, J. F.; ROSS, K. W. **Redes de computadores e a internet:** uma abordagem top-down. 3. ed. São Paulo: Person Education, 2009.

KUSMIEREK, E; CHOI, B.Y. DUAN, Z; ZHANG, Z.L. An Integrated Network Resource and QoS Management Framework. In: IEEE WORKSHOP ON IP OPERATIONS AND MANAGEMENT (IPOM), 2002, Dallas, USA. **Proceedings…** 2002. p. 68-72.

LAGHARI, K. U. R.; CONNELLY, K. Toward Total Quality of Experience: A QoE Model in a Communication Ecosystem. **Communications Magazine**, IEEE, v.50, n.4, p.58-65, 2012.

LEE, D. S. et al. Performance Analysis of a Thershold-based Dynamic Routing Algorith. In: INTERNATIONAL TELETRAFFIC CONGRESS, ITC 14., 1994. **Proceedings…** 1994.

LIN, Hwa-Chun; WANG, Sheng-Wei; TSAI, Chung-Peng. Traffic intensity based fixed-alternate routing in all-optical WDM networks. In: COMMUNICATIONS, 2006. ICC'06. IEEE INTERNATIONAL CONFERENCE ON. IEEE, 2006. **Proceedings…** 2006. p. 2439-2446.

LU, G. **Communication and Computing for Distributed Multimedia Systems**. Norwood: Artech House, 1996.

LUO REN, L.; JON TONG SENG, Q. Towards Context Information Refinement for Proximity Mobile Service using Quality of Context. In: INTERNATIONAL CONFERENCE ON MOBILE, 6., 2009. **Proceedings...** 2009.

MANZOOR, A.; TRUONG, H.; DUSTDAR, S. Quality Aware Context Information Aggregation System for Pervasive Environments. In: ADVANCED INFORMATION NETWORKING AND APPLICATIONS WORKSHOPS, WAINA '09. 2009. **Proceedings...** 2009.

MASCOLO, C.; MUSOLESI, M. SCAR: Context aware Adaptive Routing in Delay Tolerant Mobile Sensor Networks. In: INTERNATIONAL CONFERENCE ON WIRELESS COMMUNICATIONS AND MOBILE COMPUTING. IWCMC '06, 2006. **Proceedings...** 2006. p. 533-538.

MATTSSON, M. **Evolution and Composition of Object-Oriented Frameworks**. 2000. Tese (Doutorado) - University of Karlskrona, Ronneby, 2000.

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

MONGODB. 2015. Disponível em: <https:// www.mongodb.org/>. Acesso em: 7 ago. 2015.

MONGOLAB. 2015. Disponível em: <https://www.mongodb.org/>. Acesso em: 7 ago. 2015.

MUAKAD, F. **Simlação de gerência de encaminhamento para o roteamento adaptativo baseado em contexto**. 2015. Dissertação (Mestrado)- UNIFACS Universidade Salvador, Salvador, 2015.

NAZARIO, D. C.; DANTAS, M. A. R.; TODESCO, J. L. Taxonomia das publicações sobre Qualidade de Contexto. **Sustentable Business International Journal**, n.20, 2012.

NETFILTER. 2015. Disponível em <http://netfilter.org/>. Acesso em: 28 ago. 2015.

OLIVEIRA, André. **Mecanismo de notificação baseado em contexto para encaminhamento adaptativo.** 2015. Dissertação (Mestrado)- UNIFACS Universidade Salvador, Salvador, 2015.

OPENLDAP. 2015. Disponível em: <http://openldap.org>. Acesso em: 28 ago. 2015.

PETZ, A. et al. An Architecture for Context-Aware Adaptation of Routing in Delay-Tolerant Networks. In: EXTREME CONFERENCE ON COMMUNICATION, 4., 2012. Zurich, Switzerland. **Proceedings…** 2012.

PFLEEGER, S.L. **Engenharia de software:** teoria e prática. 2. ed. São Paulo: PrenticeHall, 2004.

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.

PYTHON. 2015. Disponível em: <https://www.python.org/>. Acesso em: 28 ago. 2015.

REIS, Flavio. Hardening em Sistemas Operacionais GNU/LINUX. **Revista Eletrônica da Faculdade Metodista Granbery,** Juiz de Fora – MG, 2010.

ROSEN, E. et al. **Multiprotocol Label Switching Architecture.** Disponível em: <draft-ietf-mpls-arch-05.txt>. Acesso em: 28 ago. 1999.

SCHILIT, B.; ADAMS,N.; WANT, R. Context-aware computing applications. In: WORKSHOP ON MOBILE COMPUTING SYSTEMS AND APPLICATIONS (WMCSA '94), 5., 1994. **Proceedings…**1994. p.85-90.

SCHIMUNECK, M. A. K.; MACHADO, S. R.; MAROTTA, M. A. Análise da qualidade de experiência em redes sem fio. In: SEMINÁRIO DE INICIAÇÃO CIENTÍFICA, 2014. **Anais…** 2014.

SCHUSTER, D.; ROSI, A.; MAMEI, M. et al. Pervasive Social Context-Taxonomy and Survey. **ACM Transactions on Intelligent Systems and Technology**, v.9,n. 4, p. 1-22, 2012.

SHAIKH, J.; FIEDLER, M.; COLLANGE, D. Quality of Experience from User and Network Perspectives. **Annals of telecommunications-annales des télécommunications,** v.65, n.1-2, p.47-7, 2010.

SILVA, J. P. S. et al. Context-Aware Framework for Adaptive Routing. In: INTERNATIONAL WORKSHOP ON ADVANCES IN ICT INFRASTRUCTURES AND SERVICES, (ADVANCE 2014), 2014, Miami, USA. **Proceedings…** 2014.

SKORIN-KAPOV, L.; VARELA, M. A Multi-Dimensional View of QoE: the ARCU model. In: MIPRO, 2012 INTERNATIONAL CONVENTION, 35., 2012. **Proceedings…** 2012. p. 662-666.

SOMMERVILLE, I. **Engenharia de software**. 8. ed. São Paulo: Pearson Addison Wesley, 2007.

TANENBAUM, Andrew S. **Redes de computadores.** 4. ed. São Paulo: Campus, 2006.

VIANA, W. et al. A semantic approach and a web tool for contextual annotation of photos using camera phones. In: WISE. 2007. **Proceedings…** 2007. p.225-236.

VIEIRA, V.; SALGADO, A. C.; TEDESCO, P. C. A. R. Modelos e Processos para o Desenvolvimento de Sistemas Sensíveis ao Contexto. In: JORNADAS DE ATUALIZAÇÃO EM INFORMÁTICA, 28., Bento Gonçalves, RS. 2009. **Proceedings…** 2009.

VOGEL, A. et al. Distributed Multimedia and QoS: a Survey. IEEE. **Multimedia,** v. 2, n. 2, p. 10-18, 1995.

WEISER, M. The Computer for the 21st Century.Published. **ACM SIGMOBILE Mobile Computing and Communications Review - Special**, v.3, n.3, p. 3-11. 1999.

WENNING, B.; TIMM-GIEL, A.; GÖRG, C. **A generic framework for context-aware routing and its implementation in wireless sensor networks**. ITG-Fachbericht-Mobilkommunikation-Technologien und Anwendungen, 2009.

XIE, H. et al. On Self Adaptive Routing in Dynamic Environments-an-evaluation and Design using a Simple, Probabilistic Scheme. In: NETWORK PROTOCOLS, 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. **Networking and Applications,** v. 4, n.1, 37-49, 2010.

YERIMA, S. Implementation and Evaluation of Measurement Based Admission Control Schemes within a Converged Networks QoS Management Framework. **International Journal of Computer Networks & Communications**, v. 3, n. 4, p. 137-152, 2011.

YOUTUBE. 2015. Disponivel em: <http://www.youtube.com/>. Accesso em 28 ago. 2015.

ZHAO, W.; OLSHESKI, D.; SCHULZRINNE, H. **Internet Quality of Service**: an overview. Relatório Técnico CUCS-003-00. Columbia: Columbia University, 2000

ZIMMERMANN, A.; LORENZ, A.; OPPERMANN, R. An Operational Definition of Context. In: INTERNATIONAL AND INTERDISCIPLINARY CONFERENCE ON MODELING AND USING CONTEXT (CONTEXT'07), 6., 2007. **Proceedings**… 2004. p.558-571.

APÊNDICE A - Publicações do autor

SILVA, J. P. S.; OLIVEIRA, A.; MUAKAD, F.; SPINOLA, S. ***Context-Aware Framework for Adaptive Routing***. In: INTERNATIONAL WORKSHOP ON ADVANCES IN ICT INFRASTRUCTURES AND SERVICES, (ADVANCE 2014), Miami, USA. September 2014.

SILVA, J. P. S.; OLIVEIRA, A.; MUAKAD, F.; SPINOLA, S. ***Providing Adaptive Traffic Routing Based on User and Network Context***. Submetido para publicação.

SILVA, J. P. S.; OLIVEIRA, A.; MUAKAD, F.; SPINOLA, S. ***Adaptive Traffic Routing Based on Context***. Submetido para publicação.

ANEXO A - Mapa de Filas (Bson/Json E Mongodb)


```
{
    "_id": {
        "$oid": "558851cae4b04eaf19cff593"
    },
    "flow_id": 4,
    "path": {
        "pathid": 4,
        "fwrid_ini": 4,
        "fwrid_fin": 4,
        "hop_count": 1
    },
    "link": {
        "linkid": "int",
        "fwrid_ini": "int",
        "fwrid_fin": "int",
        "status": "string"
    },
    "link_parameter": {
        "linkid": "int",
        "parm_id": "int",
        "curvalue": "int"
    },
    "pathlink": [
        {
            "pathid": "int",
            "linkid": "int"
        }
    ],
    "queue": [
        {
```

```
    "queueid": 0,
    "description": 300,
    "class": "1:10"
},
{
    "queueid": 1,
    "description": 200,
    "class": "1:20"
},
{
    "queueid": 2,
    "description": 512,
    "class": "1:40"
},
{
    "queueid": 4,
    "description": 1024,
    "class": "1:50"
},
{
    "queueid": 5,
    "description": 2048,
    "class": "1:60"
},
{
    "queueid": 7,
    "description": 4096,
    "class": "disabled"
},
{
    "queueid": 8,
    "description": 128,
    "class": "none:70"
}
```

```
    ],
    "eventforwarding": {
        "pathid": 4,
        "queueid": 4,
        "evtfrwid": 0,
        "evtid": 0
    },
    "eventlog": [
        {
            "timestamp": 1440531067.592,
            "newqueue": 2
        },
        {
            "timestamp": 1440532330.7,
            "newqueue": 2
        },
        {
            "timestamp": 1440532549.576,
            "newqueue": 3
        },
        {
            "timestamp": 1440532619.052,
            "newqueue": 3
        },
        {
            "timestamp": 1440532808.117,
            "newqueue": 4
        }
    ]
}
```

ANEXO B – Fila Notification (Bson/Json Mongodb)


```
{
   "_id": {
      "$oid": "55a2b429e4b0b251e71411d8"
   },
   "notification": "true",
   "session": "true",
   "flowidqueue": [4,5,9,11]
}
```

ANEXO C – Forward Management (Python)

```
import pymongo
import sys
import time
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

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 (2.0)

        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:
                print 'Solicitado update de fila a o Flow Adapter'

        flowadapt.update({"_id":obf},{"$set":{'eventforwarding':{'evtfrwid':1,'evtid':1,'pat
hid':flow_to_treat,'queueid':i}}},upsert=True)

        ##########################
        # espera pelo comando ser aplicado ao iptables
        # o módulo 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(2)

        # zera o event id, e retorna para o loop geral
            print 'zerando o eventid, na fila:',i
            flowadapt.update({"_id":obf},{"$set":{'eventforwarding':{'evtfrwid':0,'evtid'
:0,'pathid':flow_to_treat,'queueid':i}}},upsert=True)
            t2=time.time()
            flowadapt.update({"_id":obf},{"$push":{'eventlog':{'timestamp':t2,
'newqueue':i}}},upsert=True)
```

```
#-----------------------------------------------------------------
        return()
find()
```

ANEXO D – Flow Adaptor (Python / Linux)

```
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()
```

ANEXO E – Eventos do CAARF

Context Handler

- Recebimento ativação de Agente;
- Recebimento de desligamento normal de Agente;
- Recebimento de *Keep Alive*;
- Recebimento Nova Relação;
- Recebimento Atualização de Relação por coleta periódica;
- Recebimento Atualização de Relação pelo Usuário;
- Recebimento Encerramento de Relação;
- Recebimento Atualização do estado de caminho;
- Recebimento Atualização de parâmetros de caminho;

Notificação

- Gera notificação de *Keep Alive* das relações ativas para *Model*;
- Gera notificação de desligamento normal de Agente;
- Gera notificação de Nova relação;
- Gera notificação de Atualização de relação, seja por usuário ou outra;
- Gera notificação de Encerramento de relação;
- Gera notificação de Atualização de estado do caminho ou mudança de parâmetros do caminho;
- Notificação da inclusão de novas notificações a serem tratadas;

Outros Eventos

- Gravação dos Eventos relevantes no *Context Database*;
- Notificação de *Overload* para o *Context Model Mangement*
- Leitura das informações contextuais notificadas pelos *Handlers* nas bases *Context Database locais*;
- Salva no *Context Model Database;*
- Aplicação dos Filtros de QoC
- Mudança de Status dos eventos no *Context Model Database;*
- Desativação do registro de contexto anteriormente utilizado;

Context Model Management

- Eventos de Notificação;
    - Notificação da inclusão de novas notificações a serem tratadas;
- Outros Eventos;
    - Leitura das informações contextuais notificadas pelos *Handlers* nas bases *Context Database* locais;
    - Salva no *Context Model Database*;
    - Aplicação dos Filtros de QoC
    - Mudança de Status dos eventos no *Context Model Database;*
    - Desativação do registro de contexto anteriormente utilizado.

Context-based Forwarding Management

- Eventos de Recebimento;
    - Recebimento da inclusão de novas notificações a serem tratadas;
- Eventos de Notificação;
    - Notificação de novos fluxos para o *Flow Adaptation;*
- Outros Eventos;
    - Leitura das informações de contexto atualizadas no *Context Model Database*;
    - Processamento das regras de fluxo pré-definidas;
    - Salva das novas regras de encaminhamento no *Flow Repository*;

Flow Adaptation

- Eventos de Recebimento;
    - Recebimento da inclusão de novas regras de encaminhamento;
- Outros Eventos;
    - Consulta ao *Flow Repository;*
    - Aplicação das novas regras de encaminhamento nos comutadores.

ANEXO F – Regras de Encaminhamento

Tabela 7 - Regras de Encaminhamento

TABELA	DESCRIÇÃO DA TABELA	OBS
PARAMETERGROUP	ARMAZENA OS GRUPOS DE PARAMETROS DE QOD, QOE e QOS	IDENTIFICAÇÃO ÚNICA DO GRUPO DE PARÂMETROS
PARAMETER	ARMAZENA OS PARÂMETROS UTILIZADOS NA ARQUITETURA	IDENTIFICAÇÃO ÚNICA DOS PARÂMETROS
DEVICE	ARMAZENA TODOS OS DISPOSITIVOS DENTRO DO DOMÍNIO DO CONTROLLER SEJAM DISPOSITIVOS DE ENCAMINHAMENTO OU NÃO	IDENTIFICAÇÃO ÚNICA DOS DISPOSITIVOS
DEVICEPARAMETER	ARMAZENA TODOS OS PARÂMETROS COLETADOS DOS DISPOSITIVOS QUE SÃO AVALIADOS DENTRO DO CONTROLLER	IDENTIFICAÇÃO DO DEVICE (PARTE 1 DA CHAVE)
INDIVIDUALITY	ARMAZENA TODAS AS INDIVIDUALIDADES QUE PERTENCEM AO DOMÍNIO DA ARQUITETURA	IDENTIFICAÇÃO DO USUÁRIO
INDIVIDUALITYPARAMETER	ARMAZENA OS DADOS DE PARÂMETROS COLETADOS DA INDIVIDUALIDADE	IDENTIFICAÇÃO DA INDIVIDUALIDADE
APPLICATIONGROUP	ARMAZENA O GRUPO DE APLICAÇÕES TRATADOS NA ARQUITETURA	IDENTIFICAÇÃO DO GRUPO DE APLICAÇÕES
APPLICATION	ARMAZENA AS APLICAÇÕES QUE SÃO TRATADAS NA ARQUITETURA	IDENTIFICAÇÃO DA APLICAÇÃO
APPLICATIONPARAMETER	ARMAZENA OS PARÂMETROS BÁSICOS DE AVALIAÇÃO DAS APLICAÇÕES	IDENTIFICAÇÃO DA APLICAÇÃO (PARTE 1 DA CHAVE)
RELATION	ARMAZENA OS DADOS DAS RELAÇÕES QUE USAM O CONTEXTO	IDENTIFICAÇÃO DA RELAÇÃO
ACTIVITY	ARMAZENA DADOS ATUAIS DE CADA APLICAÇÃO USADA EM CADA RELAÇÃO	IDENTIFICAÇÃO DA RELAÇÃO
PATH	ARMAZENA OS DADOS DE CAMINHO ENTRE ROTEADOR ORIGEM E DESTINO	IDENTIFICAÇÃO DO CAMINHO
LINK	ARMAZENA OS DADOS DE LINK ENTRE ROTEADOR ORIGEM E DESTINO	IDENTIFICAÇÃO DO LINK
LINKPARAMETER	ARMAZENA OS DADOS DE PARÂMETROS COLETADOS DO LINK	IDENTIFICAÇÃO DO LINK
PATHLINK	ARMAZENA OS DADOS DE CAMINHO E LINK ASSOCIADOS	IDENTIFICAÇÃO DO CAMINHO
QUEUE	ARMAZENA OS DADOS DE TIPOS DE FILA	IDENTIFICAÇÃO DO TIPO DA FILA
FORWARDINGEVENT	ARMAZENA OS DADOS DE EVENTOS E A ESCOLHA DO NOVO CAMINHO	IDENTIFICAÇÃO DO EVENTO DE ENCAMINHAMENTO
RULES	ARMAZENA TODAS AS REGRAS DE QOC E ENCAMINHAMENTO	RULES (1 - QOC, 2 - FORWARDING) (PARTE 1 DA CHAVE)

ANEXO G – Script Controle de Filas/Iptables

```
#!/bin/bash
interface=eth1
velol=10mbit

function iniciar()
{
    #Define a classe root
    tc qdisc add dev $interface root handle 1:0 htb default 90

    #Define a taxa total do link da interface
    tc class add dev $interface parent 1:0 classid 1:1 htb rate $velol

    #Divide o link total da placa em partes
    tc class add dev $interface parent 1:1 classid 1:10 htb rate 300kbit ceil
550kbit
    tc class add dev $interface parent 1:1 classid 1:20 htb rate 200kbit ceil
250kbit
    tc class add dev $interface parent 1:1 classid 1:30 htb rate 512kbit ceil
550kbit
    tc class add dev $interface parent 1:1 classid 1:40 htb rate 512Kbit ceil
550kbit
    tc class add dev $interface parent 1:1 classid 1:50 htb rate 1024kbit ceil
1050Kbit
    tc class add dev $interface parent 1:1 classid 1:60 htb rate 2048kbit ceil
2098kbit
    tc class add dev $interface parent 1:1 classid 1:70 htb rate 128kbit ceil
148Kbit

    #Faz com que as taxas das subclasses sejam divididas por igual entre os
hosts que estiverem naquela classe
    tc qdisc add dev $interface parent 1:20 handle 20: sfq perturb 5
    tc qdisc add dev $interface parent 1:30 handle 30: sfq perturb 5
```

```
tc qdisc add dev $interface parent 1:90 handle 90: sfq perturb 5

### REGRAS BASEADAS EM PORTAS ###
iptables -t mangle -A FORWARD -o $interface -p udp -m multiport --sport 5060:5070 -j CLASSIFY --set-class 1:10
iptables -t mangle -A FORWARD -o $interface -p udp -m multiport --sport 16000:16500 -j CLASSIFY --set-class 1:20

### REGRAS PARA AS PORTAS DO UDP TESTER, TRAFFIC INJECTION
iptables -t mangle -A FORWARD -o $interface -p udp -m multiport --sport 13013:13014 -j CLASSIFY --set-class 1:70
#TESTANDO INTERNET NA ILA 1:30 PORTAS 80 E 443
iptables -t mangle -A FORWARD -o $interface -p tcp -m multiport --sport 80,443,53 -j CLASSIFY --set-class 1:30

###   FILAS DE 256K ####
iptables -t mangle -A FORWARD -s 192.168.0.101 -j CLASSIFY --set-class 1:30
iptables -t mangle -A FORWARD -d 192.168.0.101 -j CLASSIFY --set-class 1:30
iptables -t mangle -A FORWARD -d 192.168.0.101 -j CLASSIFY --set-class 1:20
iptables -t mangle -A FORWARD -s 192.168.0.101 -j CLASSIFY --set-class 1:30
###   FILAS DE 512K ###
iptables -t mangle -A FORWARD -d 192.168.0.102 -j CLASSIFY --set-class 1:20
iptables -t mangle -A FORWARD -d 192.168.0.101 -j CLASSIFY --set-class 1:20
iptables -t mangle -A FORWARD -d 192.168.0.102 -j CLASSIFY --set-class 1:20
###   FILAS DE 700k ###
iptables -t mangle -A FORWARD -d 192.168.0.102 -j CLASSIFY --set-class 1:30
```

```
        iptables -t mangle -A FORWARD -d 192.168.0.101 -j CLASSIFY --set-class
1:30
        iptables -t mangle -A FORWARD -d 192.168.0.102 -j CLASSIFY --set-class
1:30
        ###   FILAS DE 2M ###
        iptables -t mangle -A FORWARD -d 192.168.0.102 -j CLASSIFY --set-class
1:40
        iptables -t mangle -A FORWARD -d 192.168.0.101 -j CLASSIFY --set-class
1:40
        iptables -t mangle -A FORWARD -d 192.168.0.102 -j CLASSIFY --set-class
1:40
        iptables -t mangle -A FORWARD -d 192.168.0.102 -j CLASSIFY --set-class
1:20
        iptables -t mangle -A FORWARD -d 192.168.0.101 -j CLASSIFY --set-class
1:20
        #iptables -t mangle -A FORWARD -d 192.168.1.101 -j CLASSIFY --set-class
1:20
        iptables -t mangle -A FORWARD -o eth0 -p tcp -m multiport --sport 110,25,143
-j CLASSIFY --set-class 1:2    0

        }
        function parar()
        {
                tc qdisc del dev $interface root 2> /dev/null > /dev/null
                iptables -F -t mangle
                iptables -X -t mangle
        }

        function reiniciar()
        {
                parar
                iniciar
        }
```

```
function status()
{
        watch -n1 tc -d -s class show dev $interface
}

case "$1" in
        'start')
        iniciar
        ;;
        'restart')
        reiniciar
        ;;
        'stop')
        parar
        ;;
        'status')
         status
;;
        *)
        echo "usage: start | restart | stop"
esac
```