Pós-Graduação em Ciência da Computação

ERIC RIBAS MORAES MACHADO

BALANCEAMENTO DE CARGA EM REDES SEM FIO DEFINIDAS POR SOFTWARE

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE
2017

Eric Ribas Moraes Machado

Balanceamento de carga em redes sem fio definidas por software

Este trabalho foi apresentado à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre Profissional em Ciência da Computação.

ORIENTADOR(A): **Prof.. Kelvin Lopes Dias**

RECIFE
2017

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

M149b Machado, Eric Ribas Moraes
Balanceamento de carga em redes sem fio definidas por software / Eric Ribas Moraes Machado. – 2017.
80 f.:il., fig., tab.

Orientador: Kelvin Lopes Dias.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2017.
Inclui referências e apêndices.

1. Redes de computadores. 2. Redes sem fio. 3. Wi-fi. I. Dias, Kelvin Lopes (orientador). II. Título.

004.6 CDD (23. ed.) UFPE- MEI 2017-183

Eric Ribas Moraes Machado

Balanceamento de carga em redes sem fio definidas por software

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre Profissional em 05 de junho de 2017.

Aprovado em: 05/06/2017.

BANCA EXAMINADORA

_____

Prof. Carlos André Guimarães Ferraz
Centro de Informática / UFPE

_____

Prof. Andson Marreiros Balieiro
Universidade de Pernambuco

_____

Prof. Kelvin Lopes Dias
Centro de Informática / UFPE
(Orientador)

Eu dedico este trabalho a minha filha Cecília.

Agradecimentos

Primeiramente quero agradecer a Deus por me permitir chegar ao final dessa jornada que foi o mestrado, sempre iluminando meu caminho e me protegendo. Quero agradecer ao meu orientador Kelvin Lopes Dias por ter acreditado em mim e me guiado do inicio ao fim nesta caminhada.

Agradeço aos meus pais Erealdo e Maricélia por todo esforço que foi me criar e educar, além da fé inabalável que depositam em mim. Agradeço a minha irmã Alyne e meu sobrinho Rafael pelo carinho e amor incondicional. Agradeço aos meus familiares e amigos por todo apoio e orações. Em especial quero agradecer a meus queridos amigos Suellen, Guilherme e Ricardo Sassagawa, eu realmente não teria conseguido sem a ajuda de vocês.

Quero ainda agradecer ao Instituto Federal do Acre e a toda equipe da DSGTI, por segurarem a barra enquanto eu viajava e estudava, em especial a Kellyton, que por várias vezes teve que lidar com toda carga de trabalho sozinho, e Keyla, que por dividir mesa comigo sempre teve que aguentar meu choro quando estava desesperado, além de Jonas, Jair e Giuliano que sempre compartilharam seu conhecimento e experiência comigo quando precisei.

Nunca poderia deixar de agradecer aos amigos que fiz no mestrado em especial aos "Originais" (Jadson, Janderson, Marcelo, Anderson, Giovani e David) que me deram força para seguir em frente nessa jornada.

Por último quero agradecer a minha amada esposa Raíssa que mesmo diante das diversas viagens e noites de estudo sem dormir, me forneceu todo o amor, apoio e força necessária para que eu nunca desistisse dos meus objetivos, mesmo na reta final tendo que carregar nossa filha Cecília na barriga teve paciência e me manteve calmo enquanto me preparava para defesa. Te amo Raíssa!

"Sou um homem afortunado e devo buscar minha fortuna."

—HENRY AVERY, 1694

Resumo

Redes Wi-Fi estão presentes desde as residências aos ambientes corporativos e instituições de ensino e pesquisa. Contudo, a qualidade de serviço (QoS) percebida pelo usuário pode ser degradada quando em ambientes congestionados, como em lugares confinados com grande quantidade de usuários. Para lidar com esse problema várias empresas oferecem soluções verticalmente integradas e fechadas, no entanto, os custos de capital (CAPEX), com a aquisição de equipamentos e licenças, e operacionais (OPEX) decorrente do seu gerenciamento são muito altos. O surgimento das redes definidas por software (SDN) viabilizou o desenvolvimento de varias soluções abertas para melhorar o gerenciamento das redes em geral, possibilitando ao administrador da rede programar rotinas dinâmicas de monitoramento, correção de problemas e otimização do uso dos recursos disponíveis, através de um controlador centralizado com visão geral da rede. Considerando a necessidade de soluções abertas para os Institutos Federais de Educação, Ciência e Tecnologia e o potencial da programabilidade das redes sem fio definidas por software, várias soluções foram desenvolvidas para lidar com o problema da degradação do QoS, entretanto, das inúmeras soluções existentes na literatura para o gerenciamento e balanceamento de carga nestas redes, a maioria delas não visam ambientes confinados com baixa mobilidade e/ou requerem modificações no dispositivo móvel, criando burocracia para o setor de tecnologia da informação e gerando desconforto aos usuários, que muitas vezes possuem dispositivos que não suportam as mudanças necessárias. Diante deste cenário esta dissertação propõe um arcabouço para balanceamento de carga em redes sem fio (Wi-Fi) utilizando a arquitetura SDN. Existem outras propostas que promovem o balanceamento de carga em redes Wi-Fi, mas são, normalmente, orientadas ao usuário. O foco do arcabouço proposto nesta dissertação é prover uma solução orientada à infraestrutura para ambientes de rede sem fio confinados, densos e com baixa mobilidade como auditórios, salas de aula, anfiteatros, dentre outros. O modelo promove a melhoria da QoS das aplicações dos usuários através do monitoramento de dados estatísticos da rede, coletados a partir da controladora SDN e de rádios de baixo custo com firmware modificado (OpenWRT) e da análise da carga gerada por cada usuário, mitigando assim a assimetria de rede por meio da realocação da carga. Para analisar o arcabouço proposto foi desenvolvido um software, utilizado o controlador SDN Ryu, foram configurados pontos de acesso (AP) e definidos cenários de congestionamento e assimetria em um ambiente de redes sem fio. A partir dos experimentos, foi observado que, os usuários tiveram uma melhoria de QoS considerável enquanto o sistema estava ativado.

Palavras-chave: Redes Sem Fio. Wi-Fi. Redes Definidas por Software - SDN. Balanceamento de Carga. OpenWRT. QoS.

Abstract

Wi-Fi networks are present from residences to corporate environments and education and research institutions. However, user perceived quality of service (QoS) can be degraded when in congested environments, such as in confined spaces with large numbers of users. To deal with this problem several companies offer vertically integrated and closed solutions, but, the capital expenditure (CAPEX), with the acquisition of equipment and licenses, and operational expenditure (OPEX) due to its management are very high. The emergence of software-defined networking (SDN) enabled the development of several open solutions to improve network management, enabling the network administrator to program dynamic monitoring routines, problem correction and optimization of the use of available resources through centralized controller with network overview. Considering the need for open solutions for federal institutes and the potential for programmability of SDN, several solutions have been developed to deal with the QoS degradation problem, nevertheless, of the numerous solutions in the literature for managing and balancing the load on these networks, most of them do not target confined environments with low mobility and / or require modifications in the mobile device, creating bureaucracy for the IT sector and generating discomfort to the users, which often have devices that do not support the necessary changes. Considering this scenario, this dissertation proposes a framework for load balancing in wireless networks (Wi-Fi) using the SDN architecture. There are other proposals that promote load balancing on Wi-Fi networks, they are usually user-oriented. In this dissertation the proposed framework focus is to provide an infrastructure-oriented solution for confined, dense, low mobility wireless network environments such as auditoriums, classrooms and amphitheaters. The proposal improves the QoS of the users' applications by the monitoring of statistical data from network, collected through the controller SDN and low cost radios with modified firmware (OpenWRT) and analysis of the load generated by each user, thus mitigating the asymmetry through load reallocation. In order to analyze the proposed framework, a software was developed, using the Ryu SDN controller, configured access points (AP) and defined congestion and asymmetry scenarios in a wireless environment. From the experiments, it was observed that, users had a considerable QoS improvement while the system was activated.

Keywords: Wireless Networks. WI-FI. Software Defined Network - SDN. Load Balancing. OpenWRT. QoS.

Lista de Figuras

Lista de Tabelas

Lista de Acrônimos

Sumário

1 INTRODUÇÃO

Os recentes avanços tecnológicos na área de infraestrutura e comunicação permitiram aos usuários uma nova forma de interagir com serviços em redes locais e na WEB. As redes sem Fio (IEEE 802.11-WiFi) impulsionaram esse crescimento de forma exponencial, possibilitando uma série de dispositivos, como notebooks, tablets e celulares, ingressarem de forma rápida e dinâmica nestes ambientes, não havendo a necessidade de adequação/ampliação lógica ou elétrica nestes locais. Neste contexto, várias instituições de ensino, como o Instituto Federal de Educação, Ciência e Tecnologia do Acre - IFAC, aderiram a uma infraestrutura de rede mista (Cabeada e sem fio) a qual possui um nível de complexidade mais alto e requer um nível de gerenciamento maior.

Em virtude do grande aumento do uso de dispositivos móveis, bem como da necessidade crescente de acesso a redes sem fio, várias pesquisas tem sido realizadas no intuito de prover melhor qualidade de conectividade. Esta qualidade é demandada pela mudança no perfil de acesso dos usuários, anteriormente focados em navegação de sites estáticos e bate papo, e hoje voltados para acesso de vídeo e áudio sob demanda (*streaming*) e aplicações ponto a ponto (P2P) como pode ser observado no relatório apresentado pela empresa Sandvine Incorporated ULC (ULC, 2014) nas **Figuras 1.1** e **1.2**, onde streaming de aúdio e vídeo já correspondem a 67% do uso da banda larga durante o horário de pico na América do Norte, o que requer da rede maior consumo de banda e balanceamento de carga adequado. De acordo com (BRICKLEY; REA; PESCH, 2005) aplicações sensíveis ao atraso (*delay*) como vídeo e áudio sob demanda sobre redes sem fio estão cada vez mais presentes no cotidiano das empresas e instituições de ensino, no entanto o padrão IEEE 802.11b não foi desenhado para suportar esse tipo de QoS e os padrões mais atuais como o 802.11n, mesmo com suporte a QoS (filas, etc.), não resolve todos os problemas.

Um dos grandes problemas da infraestrutura de redes sem fio consiste na falta de um gerenciamento flexível e dinâmico para realizar algumas ações como aumentar e diminuir potência de sinal, alterar canal de comunicação, e controle de ruído. Além destes aspectos, a localização e realocação de clientes entre rádios mais próximos a eles de forma transparente

	Upstream		Downstream		Aggregate	
Rank	Application	Share	Application	Share	Application	Share
1	BitTorrent	25.49%	Netflix	34.89%	Netflix	32.39%
2	Netflix	9.48%	YouTube	14.04%	YouTube	13.25%
3	HTTP	7.18%	HTTP	8.62%	HTTP	8.47%
4	SSL	7.05%	Facebook	2.98%	BitTorrent	5.03%
5	YouTube	6.14%	BitTorrent	2.80%	Facebook	2.94%
6	iCloud	4.41%	iTunes	2.77%	SSL	2.63%
7	Skype	2.77%	MPEG - OTHER	2.66%	iTunes	2.55%
8	Facebook	2.60%	Amazon Video	2.58%	MPEG - OTHER	2.44%
9	FaceTime	2.38%	SSL	2.14%	Amazon Video	2.37%
10	Dropbox	1.48%	Hulu	1.41%	Hulu	1.20%
		68.98%		74.89%		73.28%

sandvine

Figura 1.1: Top 10 aplicações no horário de pico. (ULC, 2014)

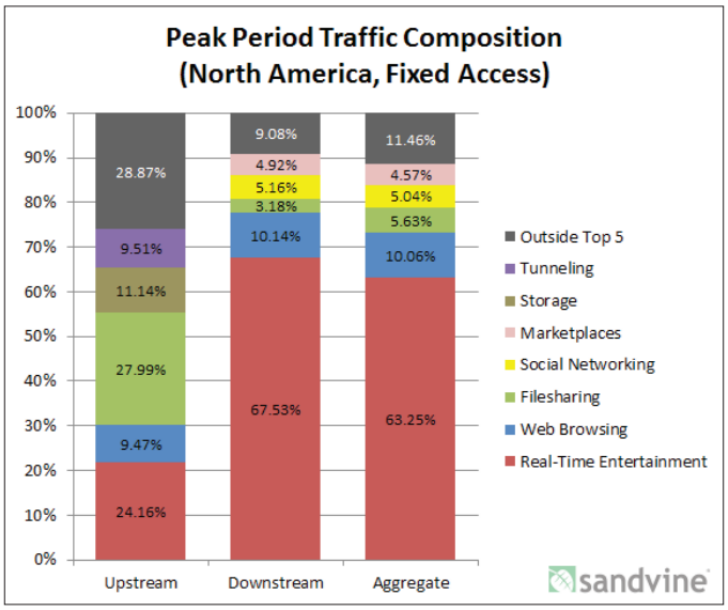


Figura 1.2: Composição do tráfego agregado no horário de pico. (ULC, 2014)

para o usuário (realizar *handoff* na camada L2/L3 [1]) e o balanceamento de carga inteligente são grandes desafios.

As arquiteturas de gerenciamento de redes sem fio locais (WLAN) atuais utilizam controladoras proprietárias, as quais realizam otimizações em toda a rede, como a seleção automática dos melhores canais de operação, ajustes na potência de transmissão de cada pontos de acesso (AP), migração rápida dos clientes entre os APs gerenciados, além de fornecer políticas de qualidade de serviço (QoS) e segurança. No entanto, essas controladoras só gerenciam dispositivos compatíveis e de mesmo fabricante. Além disso, não permitem a livre manipulação de parâmetros e o conjunto (controladora e rádios) ser muito caro (MOURA et al., 2015; DELL, 2015; HP, 2015; CISCO, 2015).

Em março de 2009 o *The Internet Engineering Task Force* (IETF) especificou o protocolo CAPWAP(*Control And Provisioning of Wireless Access Points*) através da RFC 5415 (STANLEY; CALHOUN; MONTEMURRO, 2009), este protocolo permite o controle e o aprovisionamento

[1] Layer 2/3 - protocolos das camadas 2 (enlace) e 3 (rede).

de APs de forma centralizada. Um projeto aberto para utilização do protocolo foi concebido e chamado de OpenCAPWAP, apresentado em (BERNASCHI et al., 2009), mas infelizmente a grande maioria dos APs *small-office-home-office* (SOHO) de baixo custo não suportam o protocolo. Algumas soluções proprietárias utilizando este protocolo já foram apresentadas comercialmente, no entanto, possuem um custo elevado comparado às soluções de prateleira que não suportam o protocolo.

Motivada pela necessidade de inovação e abertura para a programabilidade da rede visando o desenvolvimento de soluções alinhadas com os avanços e demandas para internet do futuro surge o conceito de redes definidas por software (SDN). No contexto de Redes sem fio, SDNs podem ser inseridas para fornecer vários aspectos favoráveis a essa infraestrutura, como segurança, qualidade de serviço, gerencia centralizada dentre outros.

A interação entre infraestrutura de redes sem fio e o potencial de programabilidade da tecnologia de redes definidas por software (SDN) permite uma coleta centralizada de dados estatísticos da rede e possibilita a manipulação livre de componentes dos roteadores, como controle de memória, lógica de roteamento, mecanismos de segurança e até mesmo controle energético através da manipulação de potencia das antenas (CHAUDET; HADDAD, 2013).

As instituições de ensino são repletas de ambientes confinados como salas de aula, auditórios e teatros, tais ambientes são caracterizados por uma concentração grande de usuários de baixa mobilidade, devido à disposição destes ambientes onde os usuários passam a maior parte do tempo sentados. Em virtude da grande densidade de usuários em um espaço limitado, os APs ficam sobrecarregados ocasionando uma queda na QoS tornando a percepção de acesso lenta, inconsistente e em alguns casos negando serviço.

Ao longo dos anos várias propostas foram apresentadas para prover balanceamento de carga em redes sem fio, e com o advento das redes definidas por software novas perspectivas se apresentam para o desenvolvimento de soluções abertas e flexíveis para o balanceamento de carga. Esta dissertação propõe um arcabouço para o balanceamento de carga dinâmico de redes sem fio, chamado SBC-Flow (Sistema de Balanceamento de carga baseado no protocolo OpenFlow), que tem sua concepção totalmente baseada em software aberto e livre, de forma a prover um balanceamento de carga adequado a redes sem fio em ambientes internos (*indoor*), de forma transparente aos usuários e com menor custo, comparado às propostas comerciais presentes no mercado.

1.1 Objetivos

A seguir, serão apresentados os objetivos gerais e específicos da pesquisa desenvolvida nesta dissertação.

1.1.1 Objetivo Geral

O objetivo geral deste projeto é apresentar uma solução para balanceamento de carga em rede sem fio local (WLAN) baseadas em uma arquitetura hibrida de redes legadas (sem suporte a SDN) e definidas por software (SDN) votada para ambientes internos (*indoor*) e com baixa mobilidade dos usuários.

1.1.2 Objetivos Específicos

- Definir um mecanismo de coleta dinâmica de informações baseado em SDN a partir do controlador SDN e dos rádios;
- Prover uma solução de baixo custo;
- Definir parâmetros que identifiquem a sobrecarga dos rádios;
- Desenvolver um algoritmo para analisar os dados coletados com base nos parâmetros de carga obtidos anteriormente;
- Prover o balanceamento de carga dos rádios através do algoritmo desenvolvido sem modificar o dispositivo do usuário;
- Desenvolver interface monitoramento WEB da solução para facilitar a gerência do administrador da rede.

1.2 Organização da Dissertação

A organização deste trabalho segue da seguinte forma. Capítulo 2 aborda a Fundamentação Teórica dos conceitos e tecnologias relacionadas, o capitulo 3 apresenta alguns Trabalhos Relacionados com a proposta, o capitulo 4 descreve o Arcabouço da Solução, o capitulo 5 apresenta a avaliação dos resultados obtidos e por fim o capitulo 6 aborda a conclusão e os trabalhos futuros.

2

FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão apresentados conceitos e tecnologias relacionadas que foram utilizados no projeto, desde o surgimento e evolução de redes sem fio até o paradigma atual das redes definidas por software.

2.1 Redes sem Fio

Em 1999 o *Institute of Electrical and Electronics Engineers* (IEEE) publicou os padrões 802.11a e 802.11b operando nas frequências 2.4GHz e 5GHz atingindo taxas nominais de 11 a 54Mbps. Naquele mesmo ano surgiu a *Wireless Ethernet Compatibility Alliance* (WECA), uma organização sem fins lucrativos que foi criada para garantir que houvesse interoperabilidade entre dispositivos de diferentes fabricantes, e mais tarde passaria a se chamar *Wi-Fi Alliance* (WFA) e criaria o selo Wireless Fidelity (Wi-Fi) para verificar a adesão de fabricantes ás especificações da tecnologia, mas devido a sua popularização acabou se tornando um sinônimo para o padrão IEEE 802.11.(IEEE, 2012; NUNES et al., 2014; PFAFF; DAVIE, 2013)

O padrão 802.11 é um conjunto de especificações que foi inicialmente desenvolvido para prover conectividade sem fio em curto alcance a um grupo reduzido de usuários por meio de espectro de frequência não licenciado (IEEE, 2012), mas com o passar dos anos esse padrão vem evoluindo e atualmente está ficando cada vez mais robusto, seguro e veloz, como pode ser visto no padrão IEEE 802.11ac (IEEE, 2013) ilustrado na figura 2.1. As redes sem fio popularizam e proliferaram em praticamente todos os setores da sociedade como industria e comércio além de viver uma época de franca expansão no setor residencial, impulsionado por tendencias como internet das coisas (IoT) e Casas Inteligentes (Smart Homes), as quais tem como base a intercomunicação de vários dispositivos dentro da residência, criando um leque de benefícios ao usuário.

Dos vários benefícios advindos das Redes sem fio, destaca-se como um dos primordiais a economicidade deste tipo de infraestrutura, onde a facilidade da instalação de novos clientes obtém vantagens tanto financeiras quanto da redução no tempo de implantação. A possibilidade da adição de novos dispositivos em rede sem a necessidade de modificações na estrutura física

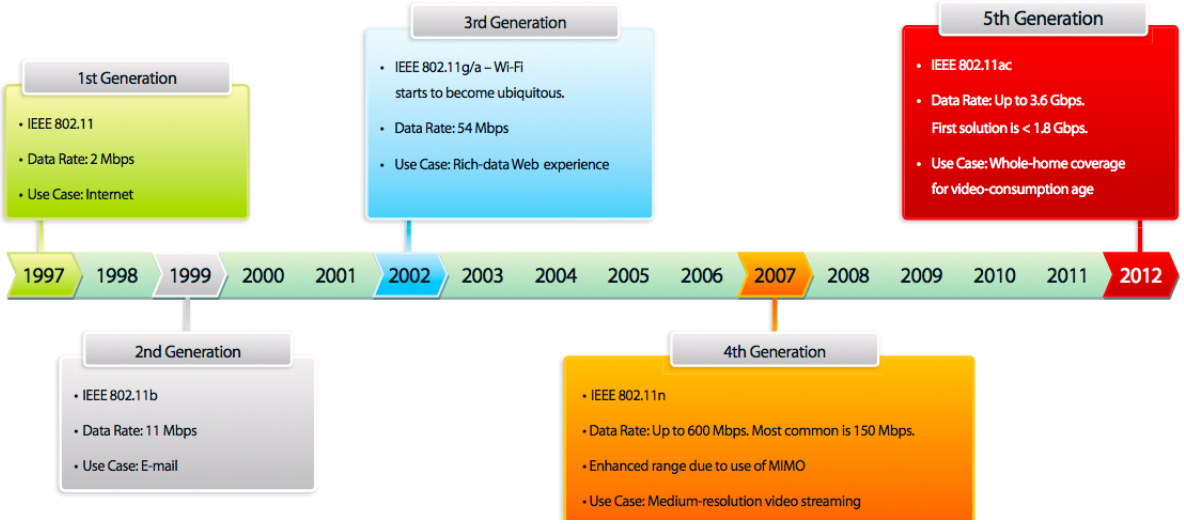


Figura 2.1: Evolução WIFI (SHIMPI, 2015)

do ambiente de trabalho impulsiona cada vez mais a utilização desta tecnologia.

Em 27 de Março de 2008 a Anatel, através de resolução, destinou a faixa de radiofrequências de 2.400 MHz a 2.483,5 MHz para uso, em caráter secundário, por sistemas do Serviço Limitado Privado, regulamentando 11 canais de 22MHz, 3 destes canais sem sobreposição (1,6,11) e restringindo a 1W de potência como pode ser observado na imagem abaixo:

Jurisdição	Faixa (GHz)	Canais Distintos	Canais sem sobreposição	Restrições de potência
EUA, Canadá	2,401 – 2,4835	11	3	1W irradiado
	5,15 – 5,35	8	8	50-250 mW irradiado 3
Brasil	2,400 – 2,4835	11	3	1W
	5,15 – 5,35	8	8	EIRP £ 200 mW 5
	5,47 – 5,725	11	11	250 mW e EIRP £ 1W
	5,725 – 5,850	5	5	1W
ETSI 1 (Europa)	2,401–2,4835	13	4	100mW ERP até 1 W 4
	5,15 – 5,35; 5,47 – 5,725	19	19	
Japão	2,4 – 2,495	14	4	10mW/MHz 2
	4,9-5;5,15-5,25	8	8	10mW/MHz

Figura 2.2: Alocação de Frequências para WLAN (TELECO, 2016)

Conforme essa infraestrutura cresce vários desafios e oportunidades vão surgindo, dentre eles, vários aspectos são observados como abrangência, desempenho, segurança e qualidade de serviço. A possibilidade de se conectar a redes sem fio a partir de qualquer posição geográfica onde haja abrangência de sinal permite que estes clientes fiquem dispersos fisicamente na área de cobertura destas redes.

Redes sem fio de médio e grande porte já utilizam tecnologias de gerencia distribuída de sinal através de controladoras e roteadores distribuídos. Estes controladores são responsáveis por autenticar os clientes, balancear a carga entre os roteadores, gerir o tráfego dentre outras funções, permitindo que o mesmo ou até vários SSIDs sejam propagados em toda área de cobertura do sinal.

2.2 Redes Definidas por Software - SDN

A Rede Definida por Software (Software-Defined Networking - SDN) é uma arquitetura emergente caracterizada por ser dinâmica, gerenciável e adaptável. Esta arquitetura separa o controle da rede da função de encaminhamento, permitindo que o controle da rede seja programável e a infraestrutura física seja abstraída para os serviços de rede e aplicações (ONF, 2017). A **Figura 2.3** ilustra a arquitetura SDN:

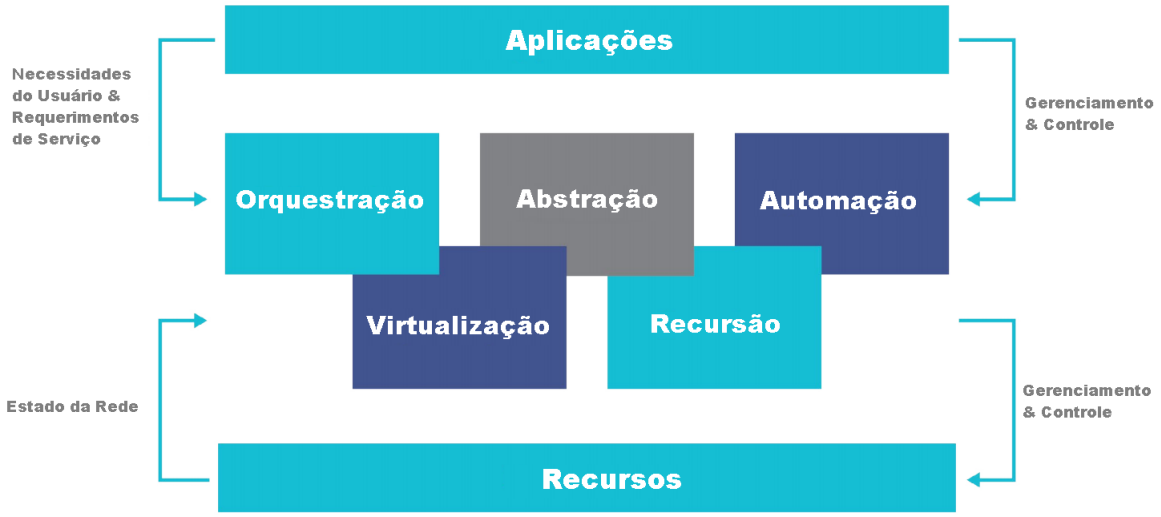


Figura 2.3: A Arquitetura de Redes Definidas por Software (ONF, 2017)

Podemos observar na **Figura 2.3** que na arquitetura SDN o controle da rede é diretamente programável (aplicações). A separação do controle de abstração do encaminhamento no plano de dados, permite que os administradores da rede ajustem dinamicamente o fluxo de tráfego da rede de acordo com as necessidades do negócio. Toda a inteligência da rede fica centralizada em controladores SDN, o qual mantém uma visão global da rede e funciona como um ponto de referência (lógico) único para aplicações e servidores de políticas de acesso mesmo que haja vários controladores. A arquitetura SDN permite que os administradores de rede possam configurar, gerenciar, assegurar e otimizar os recursos de rede de forma rápida e dinâmica através de aplicações SDN, podendo ser escritos por eles mesmos, pois não há dependência de nenhum software proprietário, já que foi implementado sobre padrões abertos, o que simplifica o uso e a modelagem da rede, pois as instruções são executadas por controladores SDN, não dependendo de dispositivos e protocolos proprietários específicos (ONF, 2017).

A arquitetura SDN tem como premissa a redução do custo e a melhoria da experiência do usuário através da automação dos serviços de rede, os princípios que promovem esta premissa são a separação do plano de controle do plano de dados, a centralização do controle e a possibilidade das aplicações desenvolvidas interagirem diretamente com o controle da rede(ONF, 2017).

2.2.1 Openflow

O protocolo OpenFlow é o elemento base para construção de soluções SDN. Ele foi desenvolvido pela Universidade de Stanford como um padrão aberto que implementa os conceitos SDN (MCKEOWN et al., 2008). O OpenFlow é a primeira interface de comunicação padrão definida entre as camadas de controle e infraestrutura de uma arquitetura SDN. Esse protocolo permite o acesso direto e a manipulação do plano de dados dos dispositivos de rede tanto físicos como virtuais, permitindo que os gestores de rede ajustem, de forma dinâmica, a rede de acordo com as necessidades de negócio reduzindo significativamente a quantidades de ações necessárias e a complexidade do gerenciamento(ONF, 2017).

Na **Figura 2.4** podemos observar o posicionamento lógico do protocolo OpenFlow e as principais características de cada camada da Arquitetura de Redes Definidas por Software.

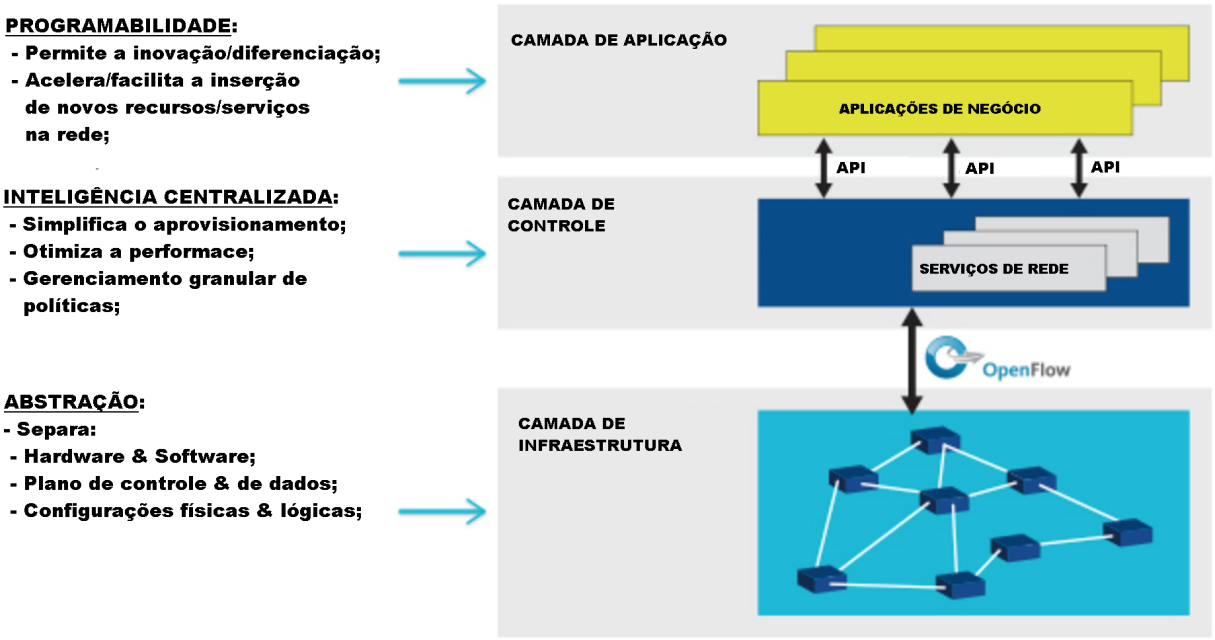


Figura 2.4: Camadas SDN (ONF, 2017)

O protocolo OpenFlow iniciou em 2008 com a versão 1.0, mas atualmente possui várias versões. Na **Figura 2.5** podemos observar uma tabela comparativa entre as especificações das versões lançadas até o momento, é importante ressaltar que alguns controladores de Redes Definidas por Software não possuem suporte às versões mais recentes e que a versão utilizada no projeto é a 1.3, pelo fato de possuir a gama de especificações necessárias para o modelo e ser compatível com todos os equipamentos, switchs virtuais (Open vSwitch - OVS) e controlador SDN utilizado.

Funções presentes	Especificações OpenFlow					
	OF 1.0	OF 1.1	OF 1.2	OF 1.3	OF 1.4	OF 1.5
Flow Table	Única	Múltiplas	Múltiplas	Múltiplas	Múltiplas	Múltiplas
MPLS	NÃO	SIM	SIM	SIM	SIM	SIM
IPv6	NÃO	NÃO	SIM(Básico)	SIM	SIM	SIM
Comunicação simultânea com múltiplos controladores	NÃO	NÃO	SIM	SIM	SIM	SIM
GroupTable	NÃO	NÃO	NÃO	SIM	SIM	SIM
Meter	NÃO	NÃO	NÃO	SIM	SIM	SIM
Suporte a interfaces de Fibra óptica	NÃO	NÃO	NÃO	NÃO	SIM	SIM
Egress Tables	NÃO	NÃO	NÃO	NÃO	NÃO	SIM

Figura 2.5: Tabela Comparativa das versões do protocolo Openflow (JÚNIOR, 2016)

2.2.2 Controlador SDN

Segundo (LINS, 2015), por gerir de forma centralizada a comunicação entre todos os elementos da rede, além de prover uma visão unificada dela, o controlador SDN é considerado o componente principal da arquitetura de redes definidas por software. Na **Figura 2.6** podemos observar as interfaces de comunicação base de um controlador SDN:

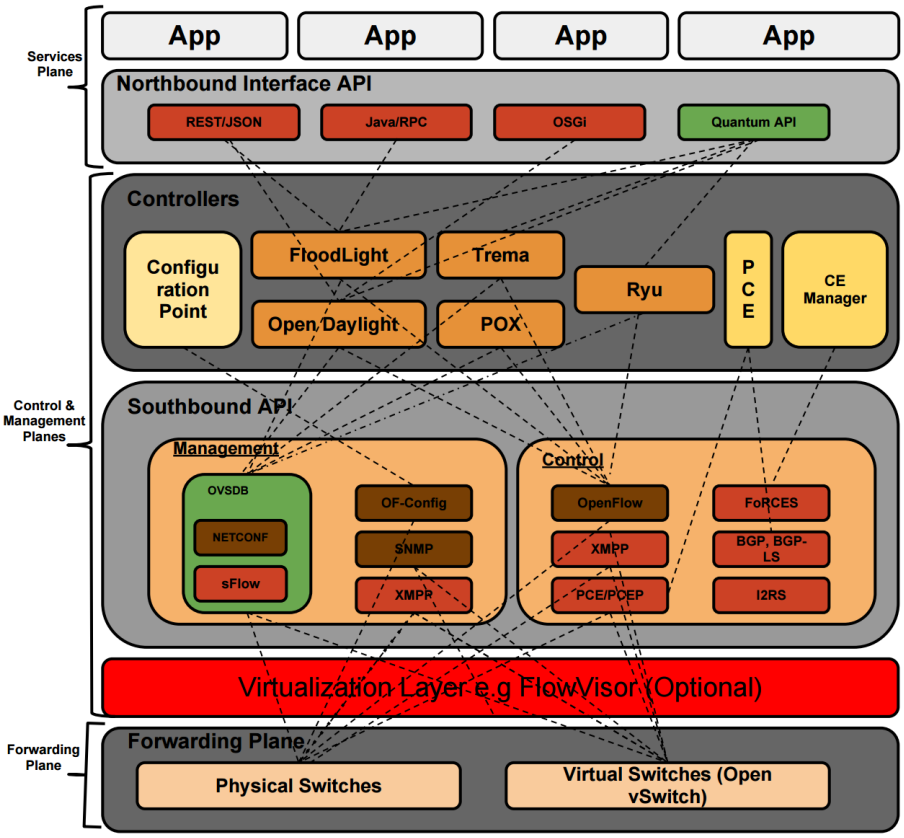


Figura 2.6: Interfaces de Comunicação dos controladores SDN (KHONDOKER et al., 2014)

De acordo com (GUEDES et al., 2012), os controladores disponibilizam um arcabouço de

programação para o desenvolvimento de aplicações. Este ambiente permite o acesso e interação com eventos gerados pela rede, além de permitir o controle dinâmico da lógica de roteamento. Através desse ambiente é possível implementar estruturas de monitoramento para perfis de tráfego mais sofisticado além de facilitar a implementação de políticas de segurança baseadas em níveis de abstração.

Existem diversos controladores para o paradigma SDN como pode ser observado na tabela comparativa 2.1.

Aspectos	POX	RYU	TREMA	FLOODLIGHT	OPENDAYLIGHT
Interface	SB(OpenFlow)	SB(OpenFlow) + SB(OVSDB JSON) + NB(REST API)	SB(OpenFlow)	SB(OpenFlow) + NB(Java + REST)	SB (OpenFlow e outros Protocolos) + NB(REST e Java RPC)
Virtualização	Mininet e Open vSwitch	Mininet e Open vSwitch	Ferramenta integrada de Emulação virtual	Mininet e Open vSwitch	Mininet e Open vSwitch
GUI	Sim	Sim	Não	Sim (WEB UI REST)	Sim
Rest API	Não	Sim	Não	Sim	SIM
Produtividade	Média	Média	Alta	Média	Média
Open Source	Sim	Sim	Sim	Sim	Sim
Documentação	Pouca	Média	Média	Boa	Boa
Suporte a Linguagem	Python	Python	C / Ruby	Java	Java
Modularidade	Média	Média	Média	Alta	Alta
Suporte a Plataforma	Linux, Mac OS e Windows	Linux	Linux	Linux, Mac OS e Windows	Linux
Suporte a TLS	Sim	Sim	Sim	Sim	Sim
Integração com OpenStack	Não	Forte	Fraca	Média	Média

Tabela 2.1: Comparativo Controladores SDN (KHONDOKER et al., 2014)

2.2.3 Ryu

O controlador utilizado nesse projeto foi o Ryu. Na **Tabela 2.1** o Ryu é apresentado como uma ferramenta de porte médio comparado com Floodlight e OpenDaylight, entretanto, apresenta todos os aspectos necessários para o projeto e possui a mesma linguagem de programação do módulo de controle (Python), facilitando a integração. Todo o seu código é gratuito disponibilizado sobre o licenciamento Apache 2.0, o qual tem uma vasta documentação na internet e indicada para desenvolvimento de aplicações móveis e WEB. O Ryu provê componentes de software com APIs bem definidas que facilitam o desenvolvimento de aplicações para controle e gerência da rede. O Ryu suporta vários protocolos para gerenciamento de dispositivos de rede, como o OpenFlow, Netconf, OF-Config, etc. A versão atual do Ryu suporta até a versão 1.5 do OpenFlow e extensões Nicira.(COMMUNITY, 2014)

Na **Figura 2.7** podemos observar a arquitetura do controlador Ryu, de cima para baixo temos a Northbound que permite a comunicação das aplicações externas (Operadores, Openstack, aplicações desenvolvidas por usuários, etc.) e o controlador Ryu. No arcabouço SDN do Ryu temos vários componentes como as bibliotecas internas, o distribuidor de eventos (Ryu Event Dispatcher), o analizador/serializador OpenFlow, o suporte aos protocolos adicionais de comunicação e as aplicações internas (Apps embutidos como descoberta de topologia, firewall, etc.) e os desenvolvidos pelo administrador da rede para automatizar uma série de funções. Logo abaixo temos a Southbound onde atua o protocolo OpenFlow e realiza a comunicação com os

switchs OpenFlow e outros dispositivos de rede.

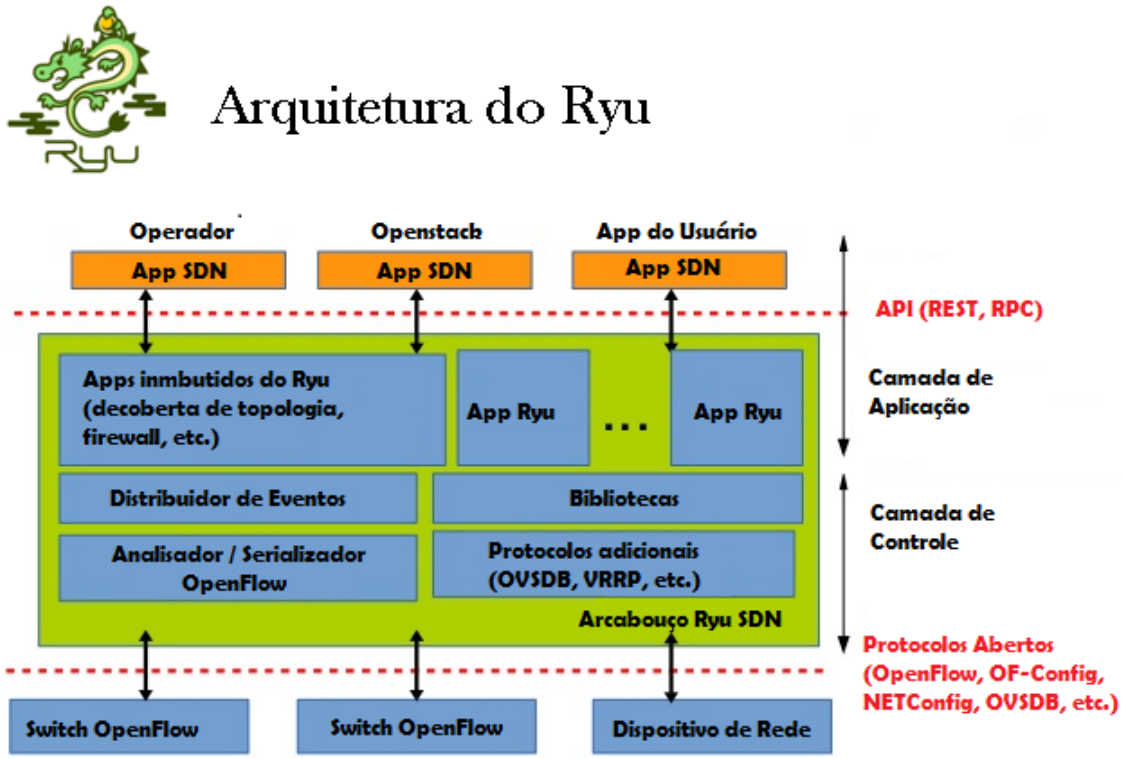


Figura 2.7: Arquitetura Controlador Ryu (YAMAHATA, 2013)

2.2.4 OpenWRT

O OpenWRT é uma distribuição de Linux para dispositivos embarcados, que possui um sistema de arquivos completamente customizável com um gerenciador de pacotes. Através do gerenciador de pacotes é possível customizar o dispositivo para atender às demandas de serviço, instalando vários serviços que antes não eram possível devido às restrições do software proprietário original do dispositivo. O OpenWRT é um projeto inteiramente livre e de código aberto licenciado pela GPL e possui uma comunidade aberta a novos contribuidores, possuem várias versões disponíveis e suportam uma vasta gama de dispositivos, o que possibilita a utilização de equipamentos de diversas marcas e modelos em conjunto sem as restrições da arquitetura dos softwares proprietários originais (OPENWRT, 2004).

2.2.5 Open vSwitch

O Open vSwitch - OVS é um projeto que tem como produto um switch virtual multicamada com qualidade de produção, ele está licenciado sobre o código aberto Apache 2.0 e foi desenhado para possibilitar e suportar automação massiva da rede através da programabilidade

da arquitetura SDN através de diversos protocolos (OpenFlow, NetFlow, sFlow, IPFIX, RSPAN, CLI, LACP, 802.1ag)(OVS, 2016).

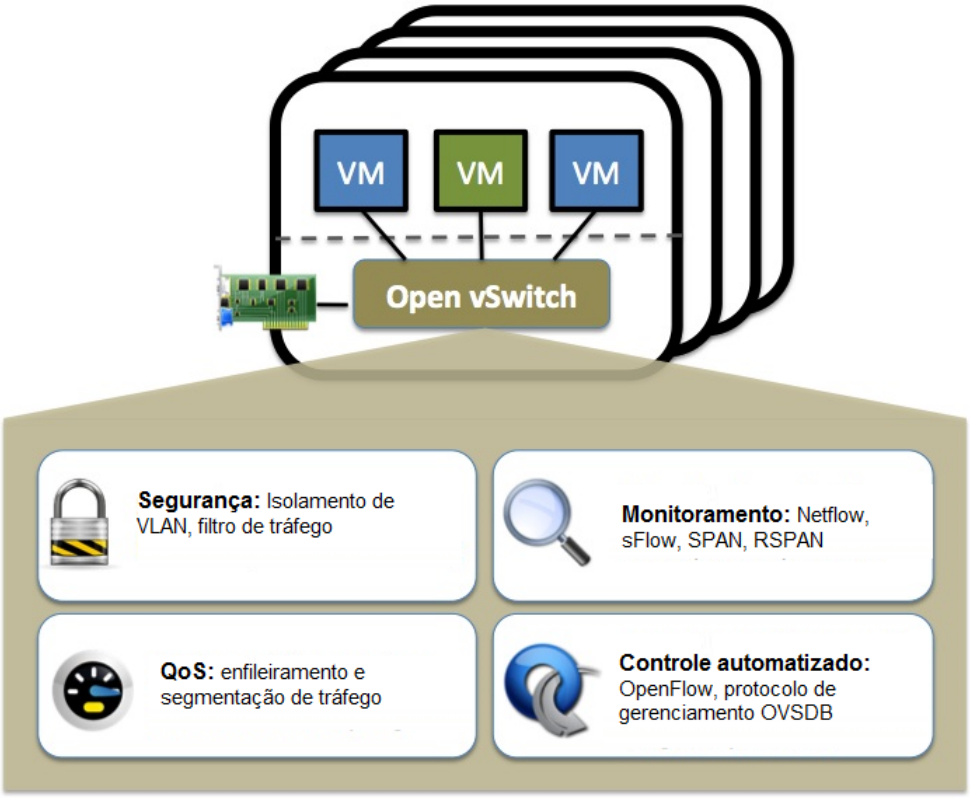


Figura 2.8: Arquitetura Open vSwitch (OVS, 2016)

Através da ferramenta OpenWRT é possível instalar o OVS no espaço do *kernel* de um rádio SOHO, possibilitando a sua integração em uma rede definida por software, permitindo desta forma, a automatização de uma série de funções e serviços de rede.

2.3 Trabalhos Relacionados

Neste capitulo serão apresentados trabalhos e conceitos relacionados a proposta, ligados à área de balanceamento de carga, redes sem fio e redes definidas por software. Ao longo dos anos várias soluções foram apresentadas para balanceamento de carga em redes sem fio, e com o advento das redes definidas por software uma série de novos trabalhos foram desenvolvidos.

2.3.1 Balanceamento de Carga em Redes Sem Fio

Segundo (YEN; YEH; CHI, 2009) as abordagens tradicionais podem ser categorizadas basicamente entre as orientadas ao usuário e as orientadas à infraestrutura, de acordo com o local onde as decisões de gerência são tomadas. Foi observado que as propostas se dividem ainda em preventivas (evitando condições de assimetria) ou reativas (corrigindo condições de assimetria,

sendo deste tipo a proposta apresentada neste trabalho), todas as propostas reativas compartilham basicamente os mesmos aspectos: um módulo de monitoramento de dados da rede, um algoritmo para detecção da assimetria e por fim, um mecanismo para realocar a carga excedente entre os pontos de conexão, provendo assim uma melhoria no QoS dos clientes.

Dos vários trabalhos publicados na área de balanceamento de carga em redes sem fio um dos que se destaca é o apresentado por (VELAYOS; ALEO; KARLSSON, 2004), esta é uma abordagem orientada à infraestrutura, onde foi proposto um esquema de balanceamento de carga entre rádios (Pontos de Acesso - APs) de uma rede sem fio local (*Wireless LAN*). Nesta proposta agentes (software cliente) foram instalados em cada AP, que periodicamente determinam a carga média de cada AP categorizando-os em sobrecarregado, balanceado e subcarregado. A métrica utilizada foi a taxa de tráfego média do AP. Os APs marcados como sobrecarregados não recebem novos clientes, permitindo somente os APs subcarregados a receber novos clientes. Esse esquema de balanceamento permite que a Taxa de tráfego média da rede seja aumentada e o atraso nos APs sejam reduzidos. Neste trabalho foram apresentados aspectos importantes como o modelo comparativo na verificação de situações de assimetria entre os pontos de acesso da topologia. No entanto este modelo é baseado em usuários que estão em movimento contínuo, transitando de um rádio para outro dentro da área de cobertura. O nosso modelo utiliza alguns aspectos deste trabalho, mas difere deste no sentido em que trabalha de forma reativa às condições de assimetria, não utiliza agentes nos rádios pois as analises dos dados e ações de contingencia são realizadas direto no controlador SDN e é voltado para ambientes confinados.

Outro trabalho relevante orientado à infraestrutura é o apresentado por (BRICKLEY; REA; PESCH, 2005) no qual é realizado um estudo sobre o padrão 802.11e buscando melhorar o QoS de usuários com perfil de áudio e vídeo sob demanda (*streaming*) balanceando a carga entre vários APs enquanto a área de cobertura deles é alterada através de uma técnica conhecida como "*Cell Breathing*". Esta técnica consiste na capacidade do AP de reconfigurar sua área de cobertura através da alteração de sua potência de transmissão, onde a redução de sua potência de transmissão implicará no encolhimento de sua área de cobertura e o aumento da potência de transmissão na ampliação da área de cobertura. Como observado no estudo, a utilização desta técnica em redes WLAN (wireless Local Area Network) permitiu a rede se adaptar às condições de assimetria de carga, onde os rádios sobrecarregados reduzem sua área de cobertura para diminuir a quantidade de usuários conectados a eles, diminuindo assim o número de colisões e a quantidade de banda necessária para as retransmissões, aumentando assim a taxa média de transferência da rede. Nossa porposta difere desse pois utilizamos algoritmos para detecção da assimetria e realocamos os usuários de acordo com sua carga média e a relação de sinal entre o AP e o usuário.

No artigo apresentado por (BALACHANDRAN; BAHL; VOELKER, 2002) são apresentadas duas abordagens, ambas orientadas à infraestrutura, para balanceamento de carga em "*Hot-spots*" públicos de redes sem fio através de algoritmos adaptativos de balanceamento de carga, com o foco na acomodação de mais usuários de forma dinâmica provendo banda onde ela

é necessária e quando ela for necessária, fazendo uso eficiente dos recursos da rede e garantindo ao menos um quantidade de banda mínima para cada usuário. Nesta proposta onde há alta mobilidade (usuários em carros, ônibus, etc.) por parte dos usuários, um AP sobrecarregado pode em questão de segundos ficar subcarregado em virtude do deslocamento físico dos clientes e como consequência deste fato, os usuários, podem vir a sobrecarregar um AP vizinho. A nossa proposta difere pois é uma solução para ambientes confinados como salas de aula em uma arquitetura de redes definidas por software e em um ambiente de pouca mobilidade dos usuários (logo que conectam ao AP permanecem longos períodos naquele local), a área em questão é coberta por todos os rádios da célula de balanceamento respeitando a ortogonalidade dos canais, que é realizada estaticamente no inicio da configuração dos rádios.

Em (BEJERANO; HAN; LI, 2004) foi implementada uma solução algorítmica, para determinar as associações entre usuários e APs que garantam em toda a rede uma alocação de banda justa aos usuários, onde este objetivo é alcançado através da utilização de algoritmos sofisticados que balanceiam a carga entre os APs. Neste esquema cada dispositivo móvel é equipado com um software cliente para monitoramento da qualidade do canal sem fio da percepção do usuário para cada um dos APs próximos a ele. O cliente repassa essa informação para o centro de controle da rede (Network Control Center - NOC), o qual, determina as associações entre usuários e APs e repassa aos clientes as decisões, desta forma os clientes se associam aos APs aos quais foram designados.

Outro trabalho orientado ao usuário foi o apresentado por (YEN; YEH, 2006) onde tanto os APs como os clientes rodam agentes SNMP e os usuários coletam de um servidor SNMP dados estatísticos da rede. O servidor SNMP armazena informações de quantidade de usuários conectados e banda residual de cada AP e mantem os usuários informados sobre o atual nível de congestionamento, através dessa informação os usuários calculam a capacidade remanescente de banda de cada AP e decidem se podem associar-se a outro AP.

No trabalho, também orientado ao usuário, apresentado por (NICHOLSON et al., 2006) é instalado em cada usuário uma aplicação a qual faz com que ele se conecte em cada AP dentro do seu alcance. Após conectado ele realiza uma bateria de testes para determinar os valores de banda e RTT (Round-Time-Trip) através de servidores TCP/UDP na internet, o resultado deste teste é armazenado em um banco de dados local dentro do AP, assim o usuário pode escolher o AP com o melhor performance.

O nosso trabalho difere destes trabalhos orientados ao usuário pois coletamos toda informação e tomamos todas as decisões a partir do Controlador SDN (orientado à infraestrutura) e não é realizado nenhum tipo de modificação nos clientes, além do fato de que utilizamos uma arquitetura de Redes Definidas por Software como base para monitoramento e coleta de informações da rede.

2.3.2 SDN em Redes Sem Fio

O advento das Redes Definidas por Software (SDN) promoveu uma série de possibilidades no intuito de tornar as redes de computadores mais dinâmicas, eficientes e seguras. Com o crescimento da popularidade do paradigma SDN na comunidade acadêmica e industria, grandes empresas como Google, HP, Dell, Microsoft já apresentam suas primeiras soluções dentro desta infraestrutura, além disso, vários trabalhos referentes a utilização de SDN no padrão 802.11 já aparecem no meio acadêmico. Seguem abaixo alguns dos trabalhos mais relevantes publicados até o momento.

Em (DELY et al., 2012) foi apresentada a Arquitetura CloudMAC para redes sem fio locais utilizando SDN. A proposta da arquitetura utiliza rádios com o projeto OpenWRT (OPENWRT, 2004), switches Openflow, o controlador SDN POX e um servidor com máquinas virtuais executando instancias de Pontos de Acesso - APs no padrão 802.11 como serviço na nuvem. O projeto consiste na retirada de parte das funcionalidades da camada MAC dos rádios físicos e inseri-las em rádios virtualizados em um servidor de dentro da rede, desta forma, tarefas como controle de cabeçalho e criptografia, que normalmente são executadas na camada MAC de cada rádio, neste modelo passam a ser executadas nesses rádios virtuais, enquanto isso os rádios físicos ficam dedicados a poucas tarefas como retransmissão de quadros e confirmações de recebimento(ACKs). Um dos pontos fortes observados no modelo foi a possibilidade de acomodar usuários em diferentes canais no mesmo AP, desta forma, um usuário com duas placas de rede conectados ao mesmo AP poderia migrar de um canal congestionado para outro sem precisar se re-associar. Outro ponto forte foi a redução da transmissão de quadros de "beacon"(beacon frames) sobre o meio sem fio através de uma abordagem sob demanda, desta forma, os quadros de beacon são transmitidos quando os usuários enviam solicitações de sondagem (probe requests). Observou-se que o CloudMAC proporcionou uma boa integração entre os elementos com e sem fio da arquitetura SDN, pelo fato de não requerer a instalação de agentes adicionais nos APs e nenhuma API para lidar com a estrutura de redes sem fio.

Outro projeto interessante e bem parecido com o CloudMAC foi o Odin, introduzido por (SURESH et al., 2012), onde ele apresenta uma plataforma SDN programável para redes sem fio locais (WLANS), eles definem o Odin como uma arquitetura de rede sem fio distribuída de MAC-dividido. Esta arquitetura foi definida desta forma pois os APs físicos executam algumas operações da camada 2 enquanto o controlador SDN executa outras operações. Sua arquitetura consiste em rádios modificados com o projeto OpenWRT (OPENWRT, 2004) e um controlador SDN FloodLight. Esta arquitetura é dividida em dois módulos (softwares), o modulo Master, o qual é executado dentro do controlador e o módulo agente que roda em cada AP. Este modelo tem como ponto central o que o autor chamou de Light Virtual Access Point (LVAP), onde cada usuário sem fio se conecta a um BSSID único enquanto permanecer dentro da área de sobreposição do sinal dos rádios da arquitetura, o que possibilita que os usuários migrem de um AP físico para outro de forma transparente sem necessidade de re-associação

e re-autenticação. No entanto é interessante ressaltar algumas limitações da arquitetura, para cada usuário conectado é acrescido um overhead devido a utilização de rádio SOHO, o qual possui limitações de hardware, o modelo pode não ser escalável. Outro ponto é que para que os usuários possam migrar de forma transparente todos os rádios devem estar configurados no mesmo canal 802.11, caso isso não seja feito, é necessária uma adaptação do 802.11h criado para espectros de 5Ghz de frequência.

É importante ressaltar o trabalho realizado por (SCHULZ-ZANDER et al., 2014), onde é utilizado como base o projeto Odin, apresentado por (SURESH et al., 2012), e é introduzido sobre a proposta um algoritmo reativo de migração de usuários para redes sem fio em sobreposição de canais. O algoritmo coleta de cada AP os usuários conectados e seus respectivos SNRs (Signal-Noise-Ratio), através destas informações é montado, para cada usuário, um mapa de APs candidatos. As decisões de migração são baseadas em análise realizada em cada usuário, onde o AP que possuir a maior percepção de sinal para aquele usuário e ao mesmo tempo acomodar sua carga atual após a migração se torna um candidato. De acordo com o autor enquanto o sistema estava ativo, observou-se que 50% dos clientes obteve uma melhoria na taxa de transferência comparado com os resultados anteriores apresentados apenas pela solução do Odin.

Em (MOURA et al., 2015) foi apresentado o projeto Ethanol, o qual consiste em uma arquitetura de Redes Definidas por Software para redes sem fio. O modelo é composto por um controlador SDN (na proposta foi utilizado o POX) e rádios comerciais de prateleira (SOHO) com uma distribuição do software PANTOU instalada no espaço do usuário. A proposta tem como foco prover gerenciamento dos usuários, segurança, mobilidade e QoS através análise e programabilidade de QoS dos fluxos geridos pelo controlador, isso foi possível a partir de um software adicional instalado em cada AP da arquitetura. Este software se comunica com o controlador SDN possibilitando a coleta e o gerenciamento dinâmico de politicas de QoS dos usuários conectados aos APs, tanto pelo meio cabeado quanto pelo meio sem fio. A proposta também apresenta uma estrutura reativa de balanceamento de carga, onde o controlador monitora eventos sinalizados pelos APs, por meio da aplicação adicional instalada, no intuito de detectar a assimetria na rede. O controlador SDN monitora os eventos de autenticação e associação de novos usuários e define políticas de controle de acesso permitindo a conexão ou negando e redirecionando o usuário para outro AP, utilizando como métrica a quantidade de usuários em cada AP em relação a um número pré-definido de usuários por AP, no entanto, em caso de uma negativa de conexão em um AP sobrecarregado, o controlador não define para qual AP o usuário deve ir, deixando o usuário livre para decidir entre os APs remanescentes. Como adicional a proposta apresentou melhorias quanto a redução de tráfego ARP no meio sem fio, através de um algoritmo de otimização dentro do controlador.

Um dos trabalhos mais recentes e relevantes dentro deste escopo de orientação a infraestrutura é o apresentado por (LUENGO, 2016) onde é apresentado um Sistema de Gerenciamento de Usuários Sem Fio (Wireless User Management System - WUMS) dentro de uma arquitetura de Redes Definidas por Software. O WUNS é dividido em dois componentes, o

primeiro é o Controlador Sem Fio Definido por Software (Software Defined Wireless Controller - SDWNC) o qual fica acima do controlador SDN e coleta informações estatísticas da rede a partir da sua interface "*Northbound API*", o segundo componente é a Aplicação de Usuário Sem Fio (Wireless User Application - WUA) o qual é instalado no dispositivo de cada usuário. O SDWNC analisa a simetria da rede através de uma formula de equidade, classificando cada AP de acordo com sua carga (sobrecarregado, balanceado e subcarregado) e repassa essas informações e decisões de migração ao usuário através do WUA, o que auxilia o usuário a migrar para o AP subcarregado mais próximo dele. Um aspecto importante sobre o algoritmo de decisão de migração deste trabalho é que ele analisa a carga média que cada usuário está gerando e prioriza para migração as conexões mais recentes (novos usuários) mantendo os antigos usuários nos APs sobrecarregados que serão balanceados. Nosso trabalho assemelha-se a esse por se tratar de uma solução de balanceamento de carga para redes sem fio definidas por software, entretanto, difere dele em vários aspectos, são eles: em (LUENGO, 2016) não é uma solução para ambientes confinados; é desenvolvido uma aplicação em java que comunica (monitorando dados estatísticos) com a *Northbound API* do controlador SDN, não sendo utilizado os benefícios de uma aplicação openflow nativa; em seu AP é instalado uma versão do software Pantou (YIAKOUMIS; SCHULZ-ZANDER; ZHU, 2012), o qual roda no espaço do usuário (no capítulo 5 é apresentado uma avaliação de desempenho sobre espaço do usuário e espaço do *kernel*) suportando somente openflow 1.0, comporta menos usuários e limita a vazão do AP; é necessário a modificação dos dispositivos de todos os usuários para usufruir da solução; utiliza como parâmetros no algoritmo de migração a vazão média de cada usuário e o tempo que ele está conectado àquele AP. Nossa proposta é voltada para ambientes confinados e a inteligência de monitoramento roda dentro do próprio controlador através de uma aplicação openflow, a qual ainda monitora os APs e realiza grande parte dos cálculos (análise e tratamento de fluxos, vazão média, relação de sinal entre AP e cliente); em nosso AP utilizamos o OpenWRT junto com o OVS no espaço do *kernel* suportando até a versão 1.5 do openflow e obtendo melhor desempenho dos APs; para usufruir da nossa proposta não é necessário a modificação do dispositivo dos usuários sendo transparente para eles; em nossa proposta utilizamos a vazão média de cada usuário e a relação do sinal entre o AP e o usuário como parâmetro do algoritmo de migração.

Os trabalhos acima inspiraram o nosso modelo, mas diferem dele de várias formas, alguns por não tratarem os ambientes confinados com concentração de usuários, outros por levarem somente em consideração a quantidade de usuários conectados e não a carga de tráfego que cada um está efetivamente gerando, há ainda outros que dependem de modificações ou instalações de módulos específicos nos rádios e usuários, deixando espaço para o desenvolvimento de mais modelos como o proposto neste trabalho, na **Tabela 2.1** é apresentado um comparativo das propostas.

Proposta	Balanceamento de Carga	Orientação	Ambiente (Indoor - Outdoor)	Perfil Usuário	Modificações no cliente	Abordagem	SDN	Aplicação OpenFlow	Controlador SDN
VELAYOS; ALEO; KARLSSON, 2004	Sim, evitando novas conexões em pontos de sobrecarga.	Usuário	Indoor / Outdoor	Alta mobilidade	Não	Perventiva	X	X	X
BRICKLEY; REA; PESCH, 2005	Sim, reduzindo a área de cobertura de APs sobrecarregados.	Infraestrutura	Indoor / Outdoor	Alta mobilidade	Não	Reativa	X	X	X
BALACHANDRAN; BAHL; VOELKER, 2002	Sim, através de técnicas de QoS.	Infraestrutura	Outdoor	Alta mobilidade	Não	Reativa	X	X	X
BEJERANO; HAN; LI, 2004	Sim, através de um controle distribuído de associação entre usuários e APs.	Infraestrutura	Indoor / Outdoor	Alta mobilidade	Sim	Reativa	X	X	X
YEN; YEH, 2006	Sim, evitando novas conexões em pontos de sobrecarga.	Usuário	Indoor / Outdoor	Alta mobilidade	Sim	Preventiva	X	X	X
NICHOLSON et al., 2006	Sim, evitando novas conexões em pontos de sobrecarga.	Usuário	Indoor / Outdoor	Alta mobilidade	Sim	Preventiva	X	X	X
DELY et al., 2012	Sim, através da acomodação de usuarios com duas placas de rede em diferentes canais no mesmo AP.	Infraestrutura	Indoor / Outdoor	Alta mobilidade	Sim	Reativa	Sim	Sim	POX
SURESH et al., 2012	X	Infraestrutura	Indoor / Outdoor	Alta mobilidade	Não	X	Sim	Sim	FloodLight
SCHULZ-ZANDER et al., 2014	Sim, ele modifica o trabalho de SURESH et al., 2012 inserindo um algoritmo de migração.	Infraestrutura	Indoor / Outdoor	Alta mobilidade	Não	Reativa	Sim	Sim	FloodLight
MOURA et al., 2015	Sim, realocando a carga excedente através de handover.	Infraestrutura	Indoor / Outdoor	Alta mobilidade	Não	Reativa	Sim	Sim	POX
LUENGO, 2016	Sim, realocando a carga excedente através de handover.	Infraestrutura	Indoor / Outdoor	Alta mobilidade	Sim	Reativa	Sim	Não	OpenDaylight
Nossa proposta	Sim, realocando a carga excedente através de handover.	Infraestrutura	Indoor (confinado)	Baixa mobilidade	Não	Reativa	Sim	Sim	Ryu

Tabela 2.2: Comparativo das propostas

3

ARCABOUÇO SBC-FLOW

O Modelo proposto é baseado em um Sistema de Balanceamento de Carga (SBC-Flow) para Redes Sem Fio em uma arquitetura de Redes Definidas por Software (SDN). O SBC-Flow ficará encarregado da inteligência do modelo, ele será o responsável por detectar a assimetria de carga na rede a partir da coleta e análise de dados oriundos do controlador SDN, e através de *handovers* horizontais migrar estes clientes, que estão gerando essa carga excedente nos rádios sobrecarregados, para rádios que estejam com a carga inferior a média da rede, corrigindo assim condições de desbalanceamento.

3.1 Arquitetura do Modelo SBC-Flow

Este modelo utiliza o conceito de células de balanceamento, onde cada célula é formada por um grupo de rádios na mesma frequência com tratamento estático de canais ortogonais cobrindo a mesma área (Overlapping WLAN coverage) tendo como foco o acomodação da carga de tráfego naquela região. O SBC-Flow é dividido em dois módulos; o de **monitoramento** e o de **controle**, como pode ser observado na **Figura 3.1**. O módulo de monitoramento é uma aplicação OpenFlow que roda integrado ao Controlador SDN, esta aplicação é responsável por criar dinamicamente os fluxos de cada cliente a medida que se conectam à rede, coleta estatísticas de fluxos, calcula o tráfego médio dos Rádios e clientes e repassa os dados para o módulo de Controle.

O módulo de controle analisa as informações proporcionadas pelo Módulo de Monitoramento, procura por situações onde um Rádio entra em condição de sobrecarga comparada com a carga média da Célula controlada e de forma reativa define a quantidade de carga que deve ser migrada alocando os clientes responsáveis por ela em outro rádio da Célula que não esteja sobrecarregado. O processo é composto por quatro fases que coletam e analisam dados específicos da rede, executando ações pré-determinadas no intuito de corrigir a assimetria, são elas as fases: Fase de Monitoramento, Fase de Análise de Carga, Fase de Análise e Seleção de Usuários e Fase de Migração e Avaliação.

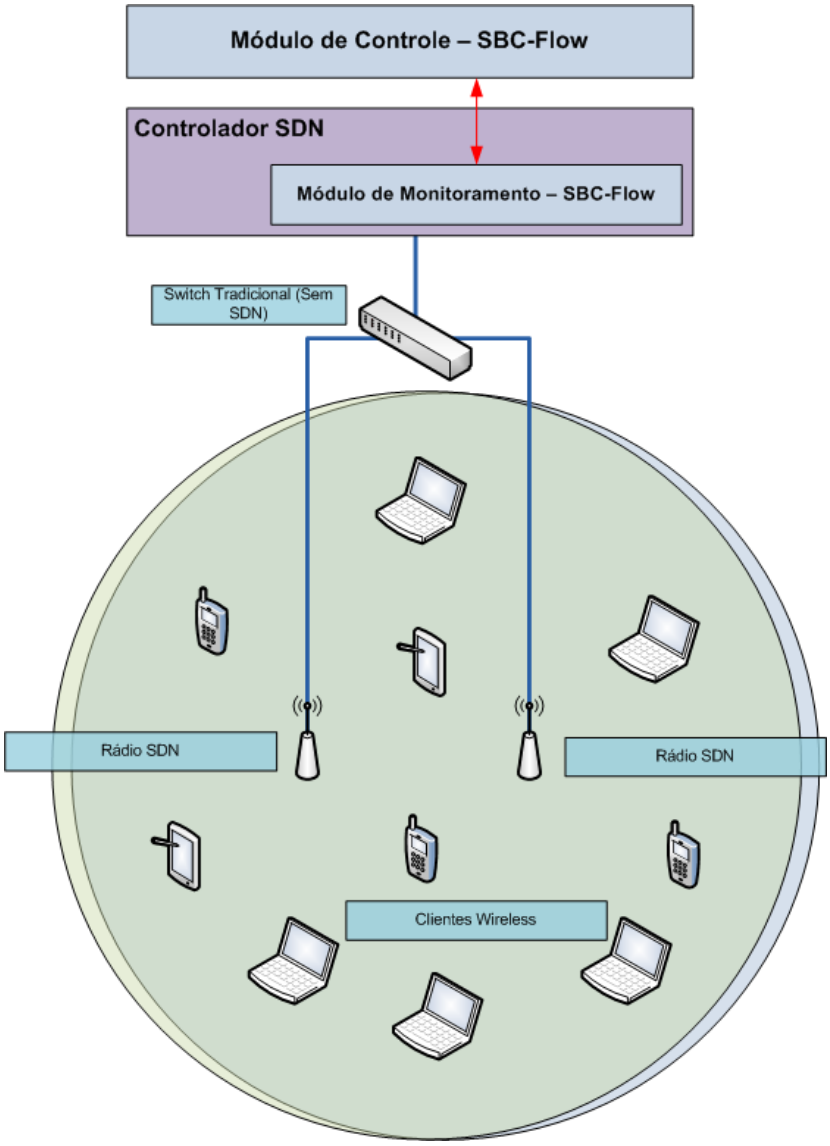


Figura 3.1: Modelo do Projeto

3.2 Fases do Algoritmo de Balanceamento

Seguem abaixo as quatro fases do algoritmo de balanceamento de carga, são elas: **Fase de Monitoramento, Fase de Análise de Carga, Fase de Análise e Seleção de Usuários e Fase de Migração**.

3.2.1 Fase de Monitoramento

Esta é a primeira fase do modelo SBC-Flow, todas etapas desta fase são executadas dentro do módulo de Monitoramento. O módulo de Monitoramento é uma aplicação OpenFlow(sbcflow_monitor.py) que está integrada com o controlador SDN (Ryu). Esta aplicação é orientada a eventos, dentro de períodos pré-determinados são enviadas requisições de dados estatísticos ao controlador e quando o evento de resposta é sinalizado, ela captura e trata os dados

coletados. O sbcflow_monitor interage dinamicamente com o controlador, os rádios cadastrados nas células e com o banco de dados do módulo de Controle coletando informações ou executando ações.

Esta aplicação é responsável por vários serviços, são eles: Criação dinâmica de fluxos no controlador, conforme os clientes vão se conectando aos rádios os fluxos são criados no controlador; Monitoramento contínuo de estatísticas do controlador, enviando requisições de fluxo e capturando os eventos de resposta; Calculo de trafego médio dos Clientes e Rádios, tomando como base de calculo os fluxos de cada cliente; Verificação de usuários conectados e coleta da percepção do sinal de cada um a partir dos rádios; Encaminhamento do resultado dos cálculos para o Banco de dados no Módulo de Controle.

Inicialmente são cadastradas as células e seus respectivos rádios (tanto células como rádios são índices numéricos, ex.: célula 1, rádio 1, rádio 2, etc.) diretamente na interface WEB do Módulo de Controle, com base nas informações cadastradas o SBC-Flow registra dinamicamente para monitoramento no controlador os rádios(switches openflow) da arquitetura. Em seguida envia requisições de fluxo de todos os clientes conectados em cada rádio em monitoramento. Nesta fase são calculadas a Taxa de Tráfego Média de cada cliente e de cada rádio, para isto, são tomados para análise os fluxos dos clientes no sentido *downlink*. Para os cálculos serão coletadas sempre 4 (quatro) amostras ($T_{u,a}$ **- Fórmula 3.1**) para maior precisão dos dados gerados, dos fluxos são extraídos a quantidade de dados transmitidos em bytes (*byte_count*) e o tempo de duração do fluxo em segundos (*duration_sec*), de posse de quatro amostras sequenciais destes dois dados é calculada a Taxa de Tráfego Média dos clientes (T_u **- Fórmula 3.2**). De posse da Taxa de Tráfego Média de todos os clientes (T_u) é realizado o cálculo da Taxa de Tráfego Média dos rádios (T_r **- Fórmula 3.3**), onde por rádios são somados as T_u de todos os clientes conectados a ele, obtendo-se assim a T_r dele naquele momento. Todas as informações coletadas e calculadas nesta fase são armazenadas no banco de dados do Módulo de Controle para uso nas próximas fases. O fluxograma dessa fase pode ser observado na **Figura 3.2**.

FÓRMULAS:

$$T_{u,a} = \frac{(byte\_count_{atual} - byte\_count_{inicial}) \times 8}{duration\_sec_{atual} - duration\_sec_{inicial}} bit/s \tag{3.1}$$

$$T_u = \frac{(\sum_{a=1}^{a=A} T_{u,a})}{A} bit/s \tag{3.2}$$

$$T_r = \sum_{a=1}^{a=U} T_{u,U} bit/s \tag{3.3}$$

LEGENDA

$T_{u,a}$ = Taxa de Tráfego de Amostra do Cliente
T_u = Taxa de Tráfego Média do Cliente
T_r = Taxa de Tráfego Média do Rádio
A = Amostra
$T_{u,U}$ = Taxa de Tráfego Média do Clientes do Rádio

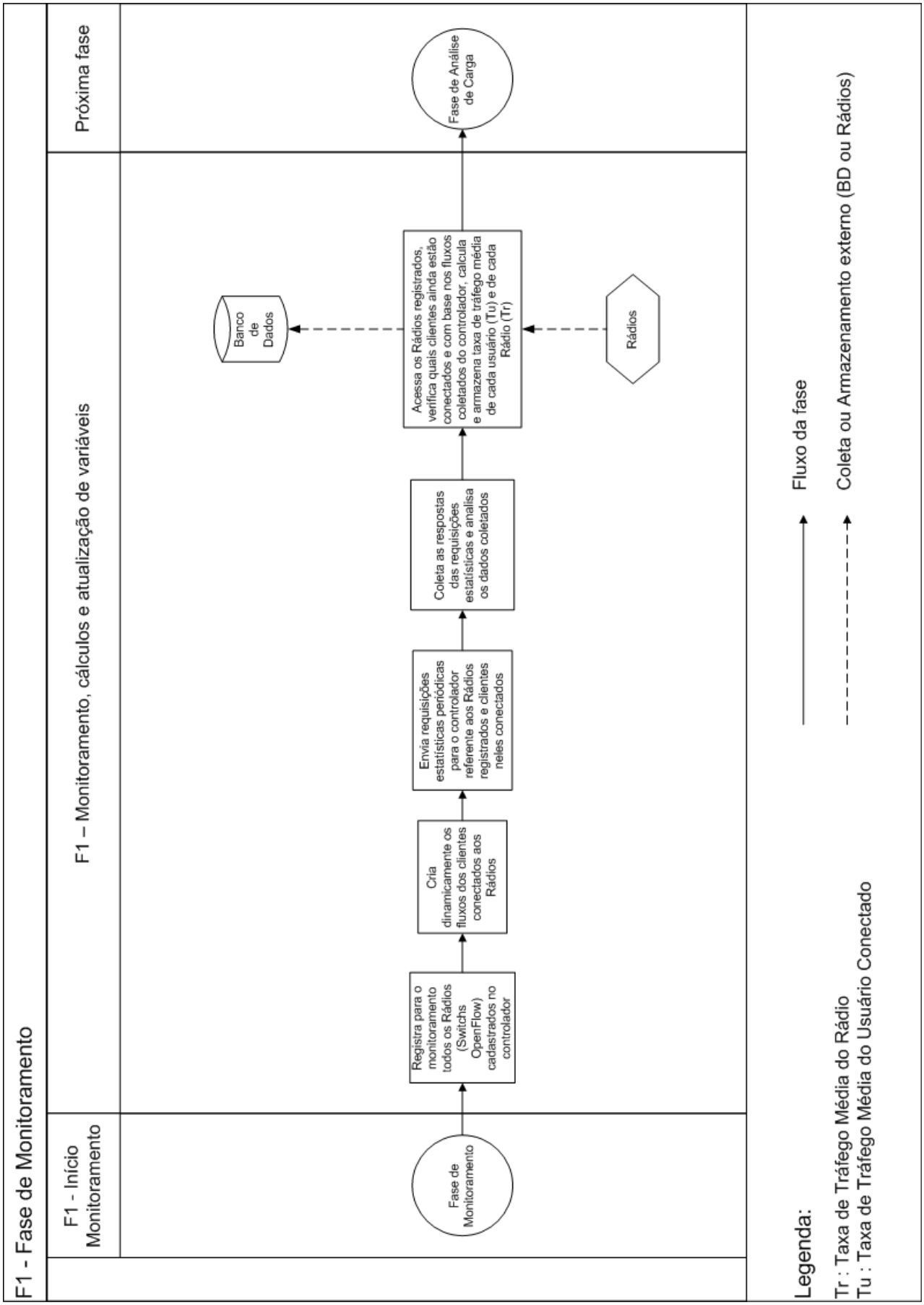


Figura 3.2: Fase de Monitoramento

3.2.2 Fase de Análise de Carga

A Análise de Carga é a fase onde é detectada a assimetria de carga nas Células de Balanceamento e está ilustrada na **Figura 3.3**. Esta fase está contida no módulo de Controle e para que seja executada é necessário que o administrador da rede realize algumas configurações de inicialização através da Interface WEB do SBC-Flow, tais como a definição das células monitoradas, rádios presentes nela, taxa de tráfego máxima suportada pelo rádio, endereço IP dos rádios, dentre outras informações. O modelo permite o cadastro de várias células e em cada célula é possível o cadastro de vários rádios, no entanto por questões de otimização do espectro e por se tratar de um ambiente confinado com redes em sobreposição de sinal, foram utilizados no máximo 3 (três) rádios na mesma frequência(2,4GHz) com tratamento estático de canais ortogonais(ch1,ch6,ch11).

Nesta fase o algoritmo percorre todas as células cadastradas, para cada célula são listados seus rádios e suas respectivas Taxas de Tráfego Média (T_r) a partir do banco de dados, em seguida é calculada a Taxa de Tráfego Média da Célula (TMC **- Fórmula 3.4**) que servirá como referência para as etapas seguintes. Após calcular o TMC é aplicada a formula de equidade(β **- Fórmula 3.5**) para verificar se há assimetria na célula, esta formula foi introduzida por (CHIU; JAIN, 1989) e anteriormente utilizada por (BALACHANDRAN; BAHL; VOELKER, 2002), (LUENGO, 2016) e (VELAYOS; ALEO; KARLSSON, 2004).

Caso haja assimetria na célula ela é marcada como "em Assimetria" em seguida, o T_r de cada Rádio é analisado com base no TMC e eles são categorizados como Sobrecarregados, Balanceados ou Subcarregados. As células em Assimetria são encaminhados para próxima fase, onde será analisado a carga que cada usuário está gerando a cada Rádio Sobrecarregado. Caso não haja células em Assimetria o algoritmo retorna à fase de Monitoramento aguardando novos dados a serem analisados.

Todas as informações produzidas nesta fase são armazenadas no banco de dados para serem utilizadas no módulo visual web e nas fases posteriores, o fluxograma desta fase pode ser observada na **Figura 3.3**.

FÓRMULAS:

$$TMC = \frac{\sum_{r=1}^{r=R} T_r}{R} \tag{3.4}$$

$$\beta = \frac{(\sum_{r=1}^{r=R} T_r)^2}{R \times (\sum_{r=1}^{r=R} T_r^2)} \tag{3.5}$$

LEGENDA

TMC = Taxa de Tráfego Média da Célula	β = Fórmula de Equidade
R = Rádios da Célula	

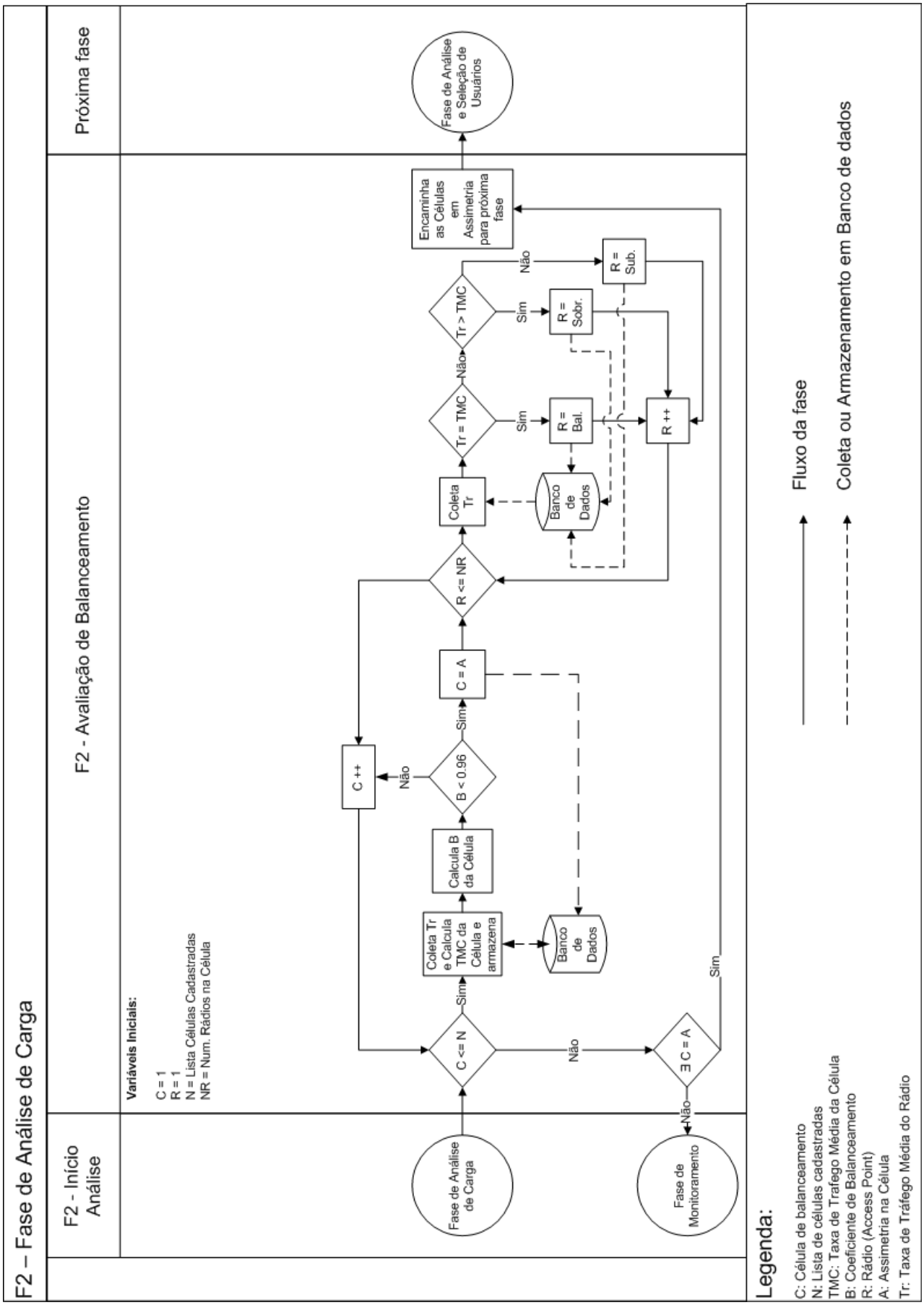


Figura 3.3: Fase de Análise de Carga

3.2.3 Fase de Análise e Seleção de Usuários

A Fase de Análise e Seleção de Usuários avalia a carga média que cada usuário está gerando ao Rádio sobrecarregado. É importante ressaltar que nem toda condição de Assimetria é passível de Balanceamento de Carga através deste Modelo, desta forma, inicialmente é avaliado se o rádio sobrecarregado tem os requisitos mínimos para prosseguir com o processo de migração, dentre os aspectos mínimos são analisados a quantidade mínima de usuários conectados e a taxa de tráfego média mínima com base na capacidade do máxima rádio, caso um destes aspectos não estejam de acordo com o mínimo esperado esta célula não prossegue para a Fase de Migração.

Com os requisitos mínimos atendidos o algoritmo segue, para cada rádio marcado como sobrecarregado é gerado a partir dele uma lista dos usuários conectados nele, SBC-Flow cria uma tabela (Tabela de Seleção de Usuários - TSU), ilustrada na **Tabela 3.1**, com os campos endereço MAC“, “taxa de trafego media(T_u)“ e “força do sinal“ de cada usuário conectado com base nos dados de seus fluxos provenientes do Módulo de Monitoramento. A tabela é ordenada de cima para baixo e de forma crescente pelo campo “força do sinal” onde os usuários com a pior percepção de força do sinal ficam no topo sendo prioritários para migração.

Endereço MAC	Taxa Média de Tráfego [Mbps]	Sinal [%]
Endereço MAC Usuário 1	Taxa Usuário 1	Sinal Usuário 1
Endereço MAC Usuário 2	Taxa Usuário 2	Sinal Usuário 2
:	:	:
Endereço MAC Usuário N	Taxa Usuário N	Sinal Usuário N

Tabela 3.1: Tabela de Seleção de Usuários (TSU)

A partir deste ponto o SBC-Flow percorre a tabela usuários sequencialmente executando o algoritmo de seleção, para cada usuário o algoritmo soma sua Taxa de Trafego médio(T_u) a uma variável acumuladora denotada aqui como Carga de Migração(CM). A CM é o montante de carga que será migrado do rádio sobrecarregado para um rádio subcarregado, mitigando a assimetria de entre os APs através da redução da carga inicial para um valor próximo ao da Taxa de Trafego média da célula(TMC). O algoritmo segue a regra descrita nas **Fórmulas 3.6 e 3.7**.

$$CM = \sum_{u=1}^{u=U} T_u \tag{3.6}$$

$$(T_{r,sobrecarregado} - CM) >= TMC - 0.5\% \tag{3.7}$$

É utilizada uma margem de 0.5% sobre o TMC aceito, o que apresenta melhores resultados na prática. Caso a condição não seja atendida o Trafego médio desse usuário é retirado do CM e o algoritmo segue para o próximo usuário da tabela, no caso da condição ter sido atendida, o usuário é marcado para migração, o processo se repete até o ultimo usuário da TSU. Ao final desta fase todos os dados são salvos no banco de dados e o algoritmo segue para a próxima fase.

Na **Figura 4.4** pode ser observado o fluxograma da fase de Análise e Seleção de Usuários.

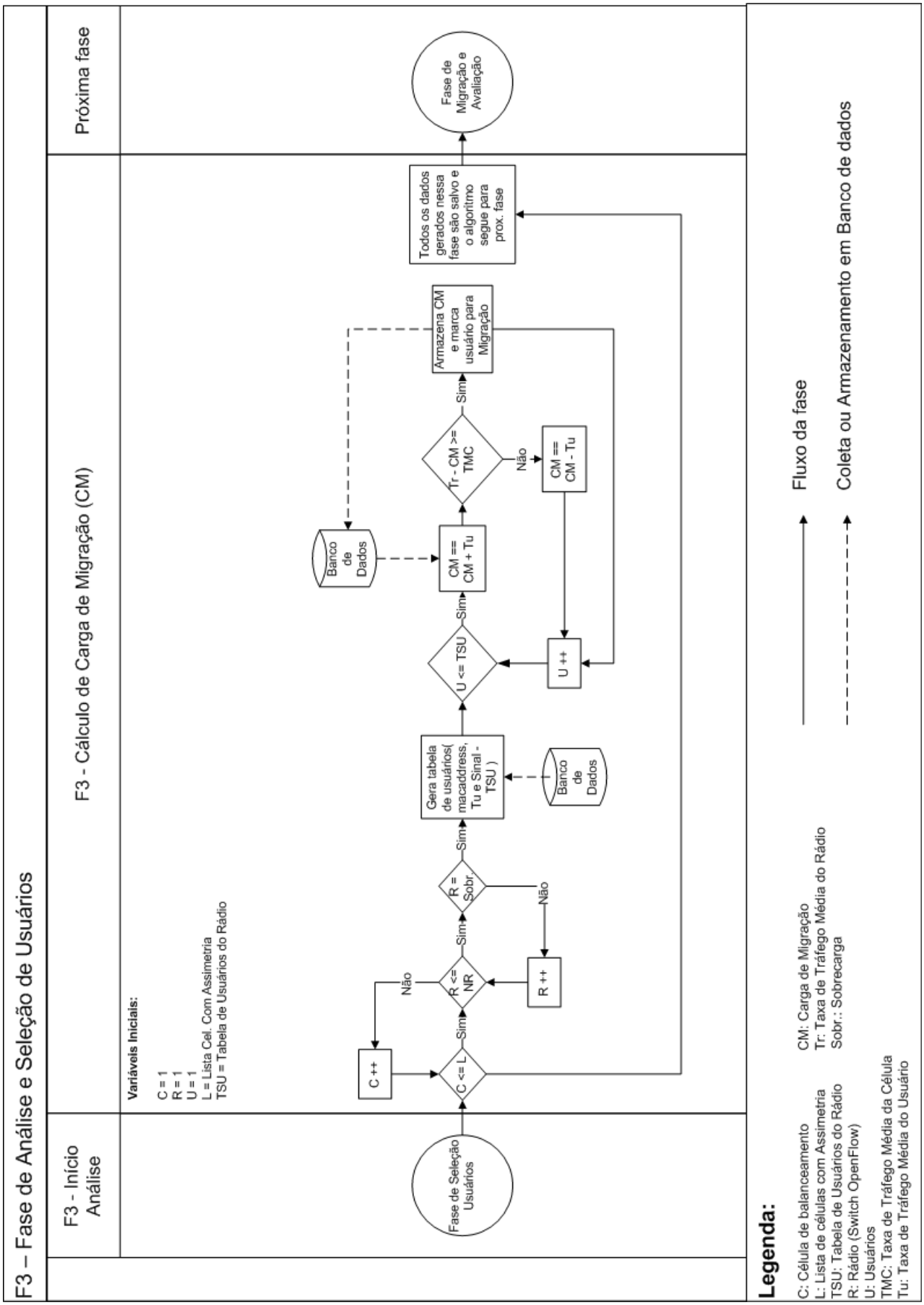


Figura 3.4: Fase de Análise e Seleção de Usuários

3.2.4 Fase de Migração e Avaliação

Esta é a fase final, onde os usuários selecionados são migrados, o balanceamento de carga das células é validado e o contador de tempo para a próxima fase de monitoramento é reiniciado. Com base nas células em assimetria, nos rádios sobrecarregados e nos usuários marcados para serem migrados, o SBC-Flow executa uma série de comandos via SSH diretamente nos rádios através da "*Unified Configuration Interface* (UCI)" do OpenWRT.

A estrutura de controle de migração é baseada em listas negras[1] (*blacklists* - *BL*), como este modelo não utiliza de modificações no lado do cliente estas *blacklists* controlam onde cada cliente pode se conectar durante as migrações, por se tratar de um modelo baseado na infraestrutura, as listas negras são configuradas dinamicamente dentro dos rádios e são atualizadas a cada ciclo de monitoramento.

O algoritmo de migração inicia seu processo percorrendo as células marcadas como assimétricas, logo em seguida percorre todos os rádios criando as *blacklists* de acordo com cada caso, o comando que efetivamente realiza a migração dos usuários só é executado após o algoritmo ter analisado e criado as listas negras em todos os rádios desta célula.

Para o Rádio marcado como sobrecarregado ele cria uma *blacklists* baseado nos usuários que estão sendo migrados e nos outros usuários dos outros rádios, isso permite que os usuários não selecionados para migração continuem no rádio e evita que ele torne a ficar sobrecarregado durante o processo de migração. Para o rádio subcarregado ele cria uma *blacklists* baseado nos usuários dos rádios sobrecarregados que não devem ser migrados, nos usuários dos outros rádios subcarregados e balanceados, isso permite que os usuários em processo de migração possam se conectar nele e evita migrações indesejadas que não foram programadas pelo SBC-Flow. Para o rádio balanceado ele cria uma *blacklists* baseado em todos os usuários dos outros rádios da célula, desta forma nenhum usuário será migrado para este rádio, evitando que fiquem sobrecarregados ou subcarregados durante todo o processo migração. Após o processo ser executado com sucesso o SBC-Flow aguarda um período pré-determinado e entra na fase de monitoramento novamente, segue abaixo as etapas da fase de Migração e Avaliação na **Figura 3.5**.

[1] são grupos específicos de usuários que não tem permissão para se conectar aos rádios.

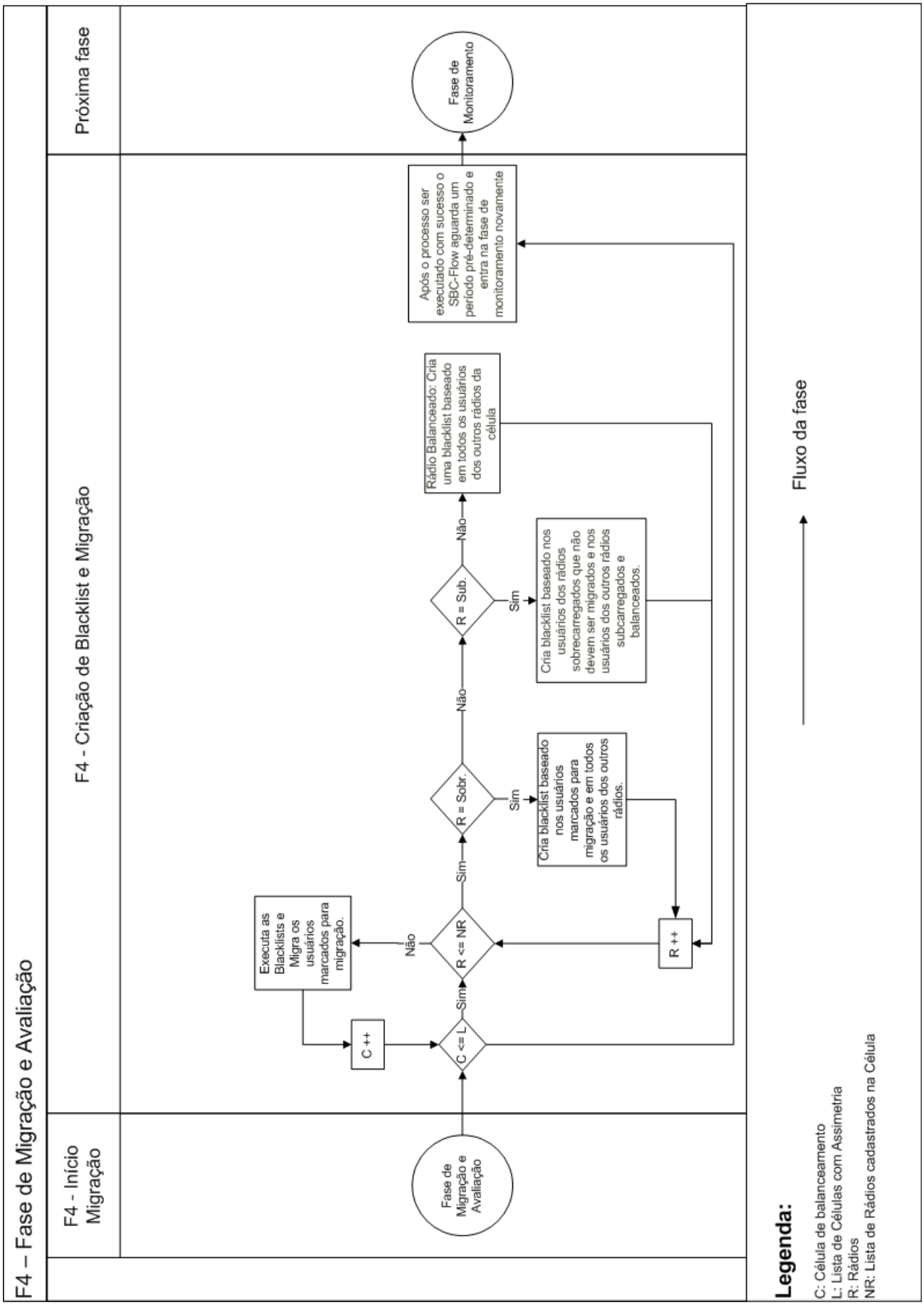


Figura 3.5: Fase de Migração e Avaliação

3.3 Sistema de Balanceamento de Carga - SBC-Flow

Nesta seção será apresentado o software desenvolvido baseado no Modelo SBC-Flow, ele foi desenvolvido em dois módulos, do de Monitoramento e o de Controle. Assim como no modelo o software coleta uma série informações da rede através do controlador SDN, detecta a assimetria na rede e de forma dinâmica realiza o balanceamento de carga. Segue abaixo os Módulos e suas funções dentro das fases do Modelo SBC-Flow.

3.3.1 Módulo de Monitoramento (SBC-Flow Monitor)

Este é o Módulo de Monitoramento, ele é uma aplicação Openflow e é executado junto ao controlador SDN. Como foi dito anteriormente o controlador utilizado para este projeto foi o Ryu, sendo assim a linguagem utilizada para desenvolver este módulo foi o Python. Este módulo é responsável por toda a coleta de informações e alguns cálculos do modelo, sendo a base de todas as outras fases das arquitetura.

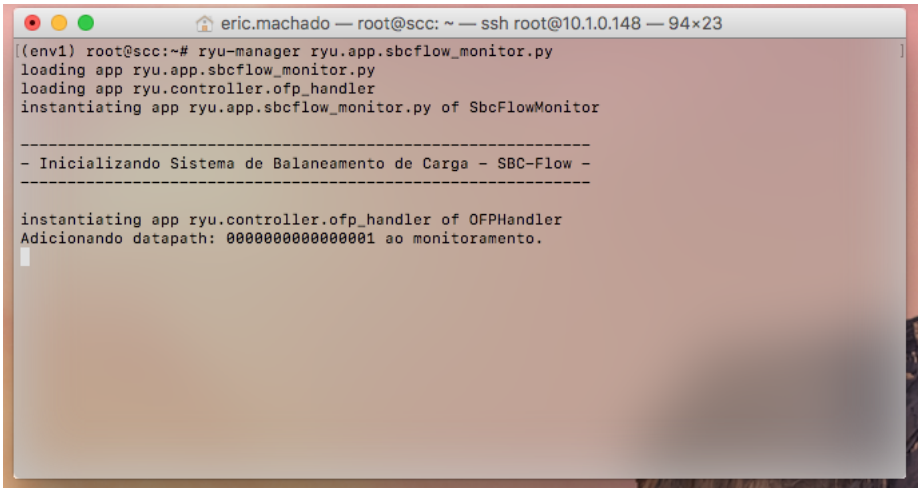


Figura 3.6: Inicialização SBC-Flow

Assim que o controlador SDN é iniciado a Aplicação de monitoramento(sbcflow_monitor.py) é inicializada junto, logo em seguida ele executa a rotina monitoramento dos switchs conectados ao controlador SDN e os registra para monitoramento contínuo e criação de fluxos dinâmica, conforme os usuários vão se conectando ao switch registrado suas respecttivas regras de fluxo são criadas automaticamente. Os Switchs são mantidos na lista de monitoramento enquanto estiverem ligados e conectados ao Controlador SDN, esta rotina pode ser observada na **Figura 3.6**.

Após registrar os switchs para monitoramento ele inicia a rotina cíclica de solicitação de estatísticas dos dispositivos registrados, o tempo desta rotina é definido pelo administrador da rede. Ao final de cada rotina ele comunica com a base de dados no Módulo de controle, e atualiza o horário da ultima coleta de dados, mantendo o módulo de controle atualizado sobre a necessidade de análise de novos dados.

Iniciado o processo de monitoramento, o sbcflow_monitor se conecta ao banco de dados do módulo de controle para coletar informações sobre a estrutura das células de monitoramento cadastradas nas configurações de inicialização do SBC-Flow (falaremos mais sobre as configurações de inicialização na seção do Módulo de Controle), para cada célula cadastrada, ele envia para o controlador as requisições de estatísticas de fluxo dos usuários conectados aos rádios em monitoramento.

As aplicações OpenFlow são orientadas a eventos, sendo assim após o envio das requisições de estatísticas de fluxo é necessária uma rotina para coletar e tratar o resultados da requisição. O SBC-Flow coleta, trata e armazena as amostras dos dados necessários para o cálculo de vazão dos rádios (switchs openflow) e dos usuários, ela se conecta aos rádios, lista os usuários conectados e coleta a percepção de sinal do rádio em relação ao usuário, ele separa 4 (quatro) amostras de vazão de cada usuário e envia para rotina de cálculo de vazão média dos usuários.

Na ultima etapa do monitoramento do SBC-Flow, a partir das amostras coletadas na etapa anterior, é calculada a vazão de cada amostra a partir do ***byte_count*** e do ***duration_sec*** e a partir das 4 (quatro) amostras ele calcula a vazão média de cada usuário, terminado o calculo, ele armazena a vazão média e a percepção de sinal de cada usuário no banco de dados do Módulo de Controle

Na **Figura 3.7** pode-se observar o log da aplicação na tela de console do controlador SDN com os resultados dos cálculos realizados nesta fase, como saída do log são apresentados o Rádio (Datapath - Switch Openflow) em análise, o endereço MAC, as amostras de vazão, a vazão média e o sinal de cada usuário conectado nele.

```
------------------
Datapath - Switch
------------------
0000000000000001
----------------- --------------- --------------- --------------- --------------- ------
MAC (Clientes)    trafego_medio   taxa_2          taxa_3          taxa_4          Sinal
----------------- --------------- --------------- --------------- --------------- ------
00:36:76:a0:bb:cc  0.91 Mbits/s    1.46 Mbits/s    0.73 Mbits/s    0.53 Mbits/s    -28
----------------- --------------- --------------- --------------- --------------- ------
00:36:76:a0:bc:01  0.00 Mbits/s    0.00 Mbits/s    0.00 Mbits/s    0.01 Mbits/s    -40
----------------- --------------- --------------- --------------- --------------- ------
00:36:76:a0:b9:35  1.72 Mbits/s    0.30 Mbits/s    1.68 Mbits/s    3.18 Mbits/s    -39
----------------- --------------- --------------- --------------- --------------- ------
68:94:23:4b:80:a6  0.15 Mbits/s    0.00 Mbits/s    0.00 Mbits/s    0.46 Mbits/s    -27
----------------- --------------- --------------- --------------- --------------- ------
68:94:23:4b:83:23  0.00 Mbits/s    0.00 Mbits/s    0.00 Mbits/s    0.00 Mbits/s    -31
----------------- --------------- --------------- --------------- --------------- ------
```

Figura 3.7: Log do SBC-Flow no console do controlador SDN.

3.3.2 Módulo de Controle (SBC-Flow WEB)

O Módulo de controle do Modelo SBC-Flow é uma aplicação desenvolvida em Python sobre o Framework Django (DJANGO, 2005), ela possui uma interface WEB onde é possível acessar e realizar as configurações de inicialização, monitoramento das fases e rotinas do

modelo e um banco de dados (BD) onde são armazenados todos os dados coletados e gerados pela aplicação. A interface WEB é dividida em dois ambientes, o Administrativo e o de Monitoramento. O ambiente Administrativo é criado dinamicamente pelo ***Django*** como base de interação entre as telas de acesso da Interface WEB e o banco de dados da aplicação, facilitando a criação de usuários de acesso a ferramenta, inserção de dados no BD e cadastro das informações de inicialização do SBC-Flow. Na **Figura 3.8** podemos observar a tela de login do ambiente administrativo.

Figura 3.8: Tela de acesso ao Ambiente Administrativo do SBC-Flow

Figura 3.9: Tela principal do Ambiente Administrativo do SBC-Flow

Na **Figura 3.9** está a tela principal do ambiente administrativo do SBC-Flow onde é possível criar grupos e usuários para ter acesso a este ambiente, definir as configurações de inicialização da aplicação como as Células de Monitoramento, as definições do controlador SDN e switchs (rádios) da estrutura a ser monitorada e balanceada, é possível ainda verificar os usuários que estão conectados nos rádios e verificar as ultimas alterações realizadas no log das ações recentes.

Para inicialização da ferramenta é necessário o cadastro das células de monitoramento, que neste modelo compreendem um grupo de rádios que se deseja monitorar e realizar balanceamento de carga entre eles, sendo assim as células representarão o ambiente confinado onde se deseja acomodar a carga dos usuários, como salas de aula, auditórios, anfiteatros, dentre outros. Após o cadastro das células o próximo passo é cadastrar os switchs (rádios) nas nelas. Na **Figura 3.10** podemos ver da tela de cadastro de switch.

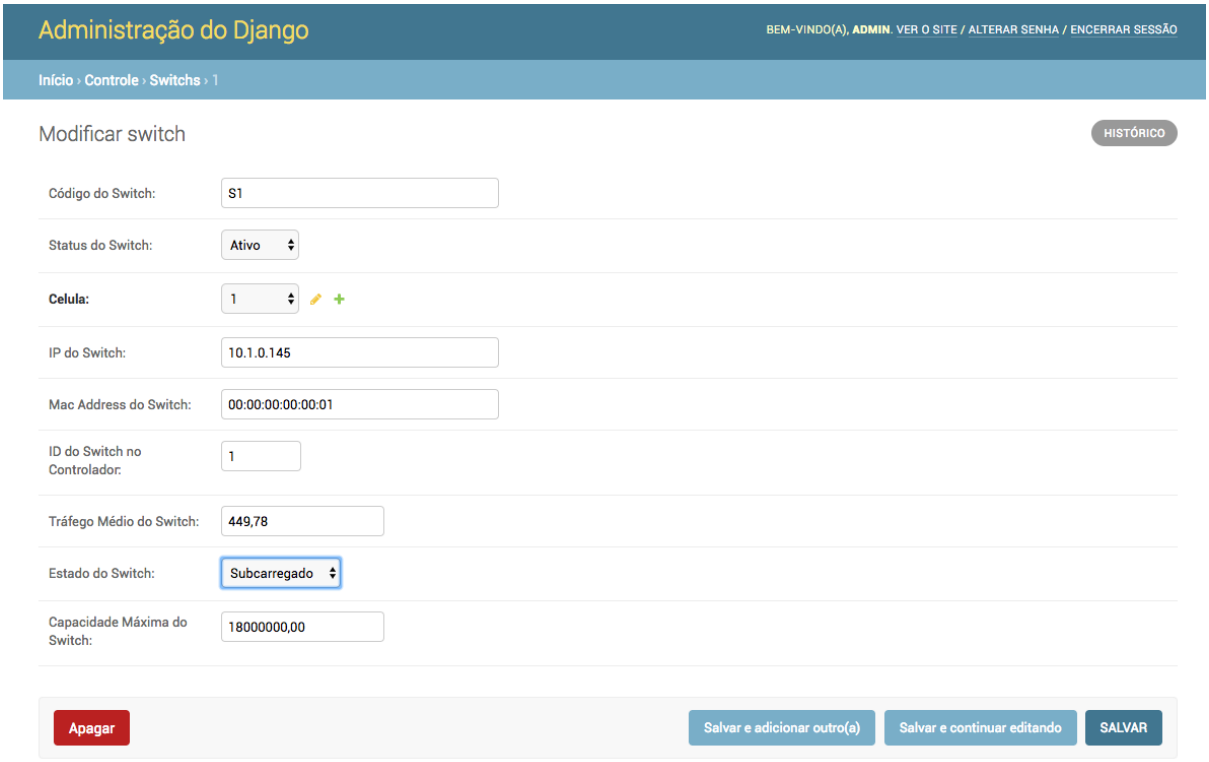


Figura 3.10: Tela de cadastro de Switchs

No cadastro dos switchs é informado pelo administrador o código do switch, o status dele (ativo/inativo), a célula a qual ele pertence, o endereço IP dele, o endereço MAC dele, o ID do switch no controlador (o ID de Datapath dele no controlador) e a capacidade máxima de vazão do switch. A capacidade máxima de vazão do switch é definida a partir de testes de desempenho realizados pelo administrador, este campo é editável pelo fato da arquitetura usar rádios com suporte ao OpenWRT, o que permitirá a utilização de uma grande quantidade de rádios comerciais (SOHO), dentre eles haverão rádios mais robustos que suportam uma capacidade maior de vazão máxima, este dado é utilizado no algoritmo que define se o rádios

tem os critérios mínimos para o balanceamento de carga.

Finalizadas as configurações de inicialização do SBC-Flow no ambiente administrativo, o controlador SDN e a aplicação OpenFlow sbcflow_monitor são ativados e o monitoramento começa, o status das células cadastrados pode ser observado através do ambiente de monitoramento da interface WEB do módulo de controle (**Figuras 3.11 a 3.14**, a imagem foi recortada para que seja possível visualizar todas as informações do ambiente de monitoramento).

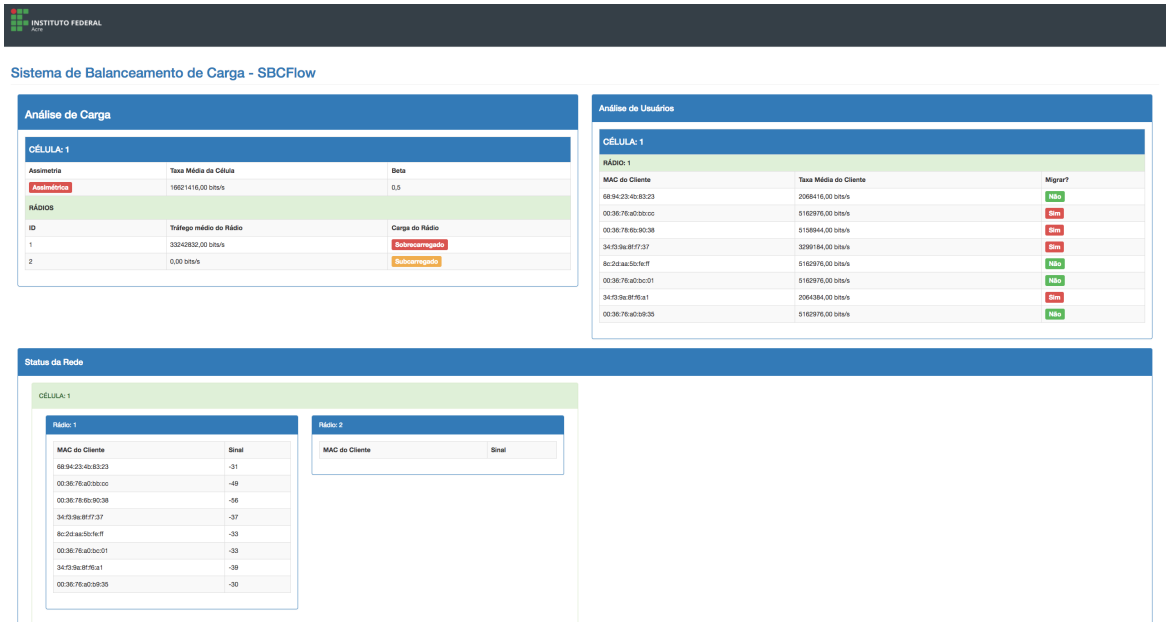


Figura 3.11: Interface WEB - Ambiente de Monitoramento.

A visualização do ambiente de monitoramento é formada dinamicamente de acordo com as condições de simetria das células. Há duas telas fixas neste ambiente, a de status da rede (**Figura 3.12**) e a de análise de carga (**Figura 3.13**) as quais, analisam a simetria das células de balanceamento e monitoram a localização dos clientes dentro dos rádios. Caso haja assimetria em alguma célula a tela de análise de usuários (**Figura 3.13**) aparece informando quais deles devem ser migrados.

A tela de status da rede (**Figura 3.12**), fomentada pelo módulo de monitoramento (sbcflow_monitor), monitora a cada ciclo a localização de cada usuário nos rádios. Essa informação é necessária para que as próximas fases do algoritmo de balanceamento sejam executadas, além de ficar visualmente disponível para o administrador da rede, facilitando a localização de usuários específicos para fins administrativos.

A primeira fase deste ambiente é a análise de carga (**Figura 3.13**), onde a partir da vazão dos fluxos dos usuários, provenientes da fase de monitoramento da aplicação openflow, é calculada a carga de cada rádio. De posse da vazão de cada rádio três ações são realizadas nesta etapa, primeiramente é calculada a vazão média da célula (taxa média da célula), em seguida é calculado o coeficiente de equidade (β), que é responsável por aferir a simetria dos rádios na célula. Detectada a assimetria em alguma célula, os rádios dela passam a ser categorizados em

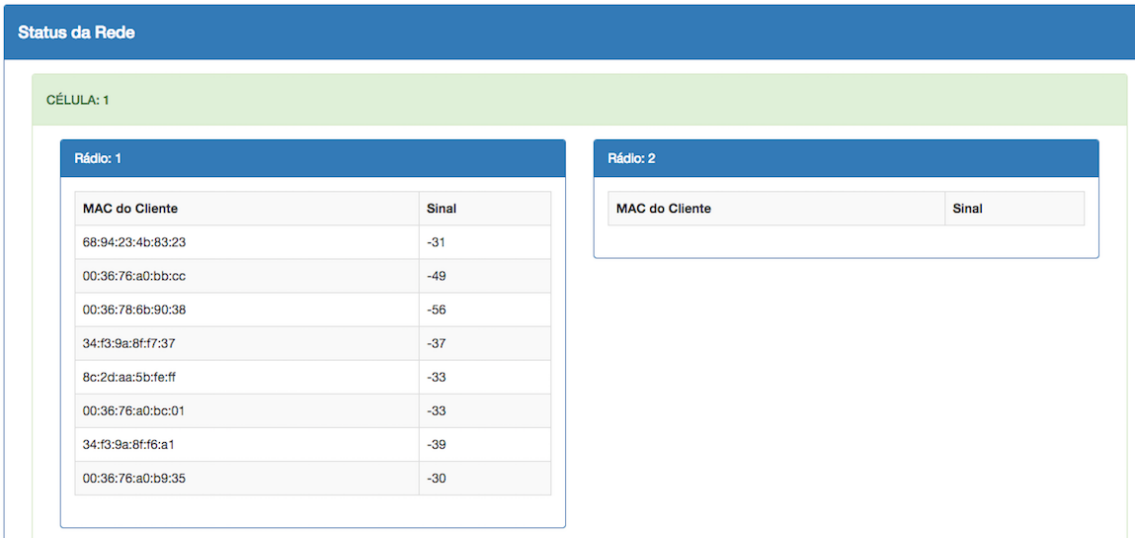


Figura 3.12: Interface WEB - Status da Rede.

subcarregado (U), balanceado (B) e sobrecarregado (O). Todo processo ocorre conforme foi explicado na **seção 3.2.2** (fase de análise de carga).

Figura 3.13: Interface WEB - Análise de Carga.

Como foi dito anteriormente, em um cenário onde seja detectada a assimetria em uma célula a tela de análise de usuários (**Figura 3.14**) aparece indicando os usuários que serão migrados para balancear a carga, para que isto seja possível cada rádio sobrecarregado é analisado e são definidos quais usuário devem ser migrados, como foi explicado anteriormente na **seção 3.2.3** (fase de análise e seleção de usuários).

Nesta fase o algoritmo valida se há as condições mínimas necessárias para executar as migrações, como os usuários são ordenados pela potência de sinal e selecionados para migração. Como esta solução balanceia a carga através da realocação da carga, são compreendidos como condições mínimas para migrar usuários, o rádio possuir no mínimo 3 (três) usuários e a vazão do rádio ser maior que 50% (cinquenta por cento) da capacidade máxima cadastrada dele (configuração realizado na inicialização da ferramenta).

Após o monitoramento, a análise de carga e a análise e seleção de usuários, a última

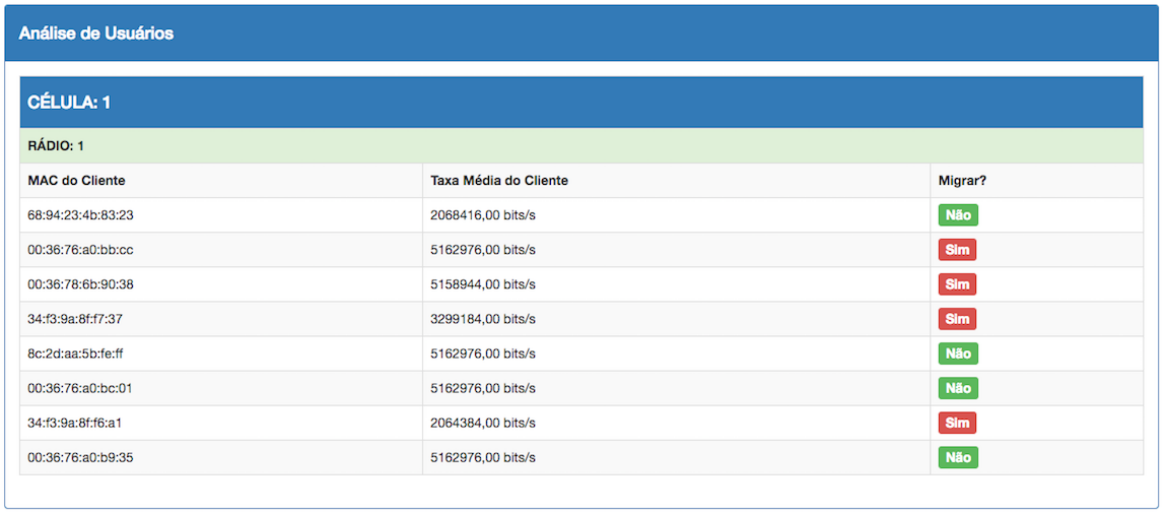


Figura 3.14: Interface WEB - Análise de Usuários.

fase é iniciada. Ao chegar na fase de migração todos os pré-requisitos para que a célula seja balanceada já foram executados, restando somente a efetiva realocação dos usuários. Para que a realocação seja efetiva e os usuários saiam de rádios sobrecarregados e migrem para rádios subcarregados, uma série de comandos são executados nos rádios criando listas negras em todos os rádios. As listas negras são criadas com base nas análises realizadas nas fases anteriores, induzindo todos os rádios a ficarem com a vazão média próximo da média da célula, além de fazer com que o tráfego nesse segmento da rede seja equilibrado, melhorando a vazão média (*throughput*) e a perda de pacote dos usuários.

Logo após a execução bem sucedida de todas as fases da solução, as células antes em assimetria entram em um estado de equilíbrio e a interface WEB é atualizada, mostrando o status atual da rede e a nova distribuição dos usuários nos rádios, como pode ser observado na **Figura 3.15**.

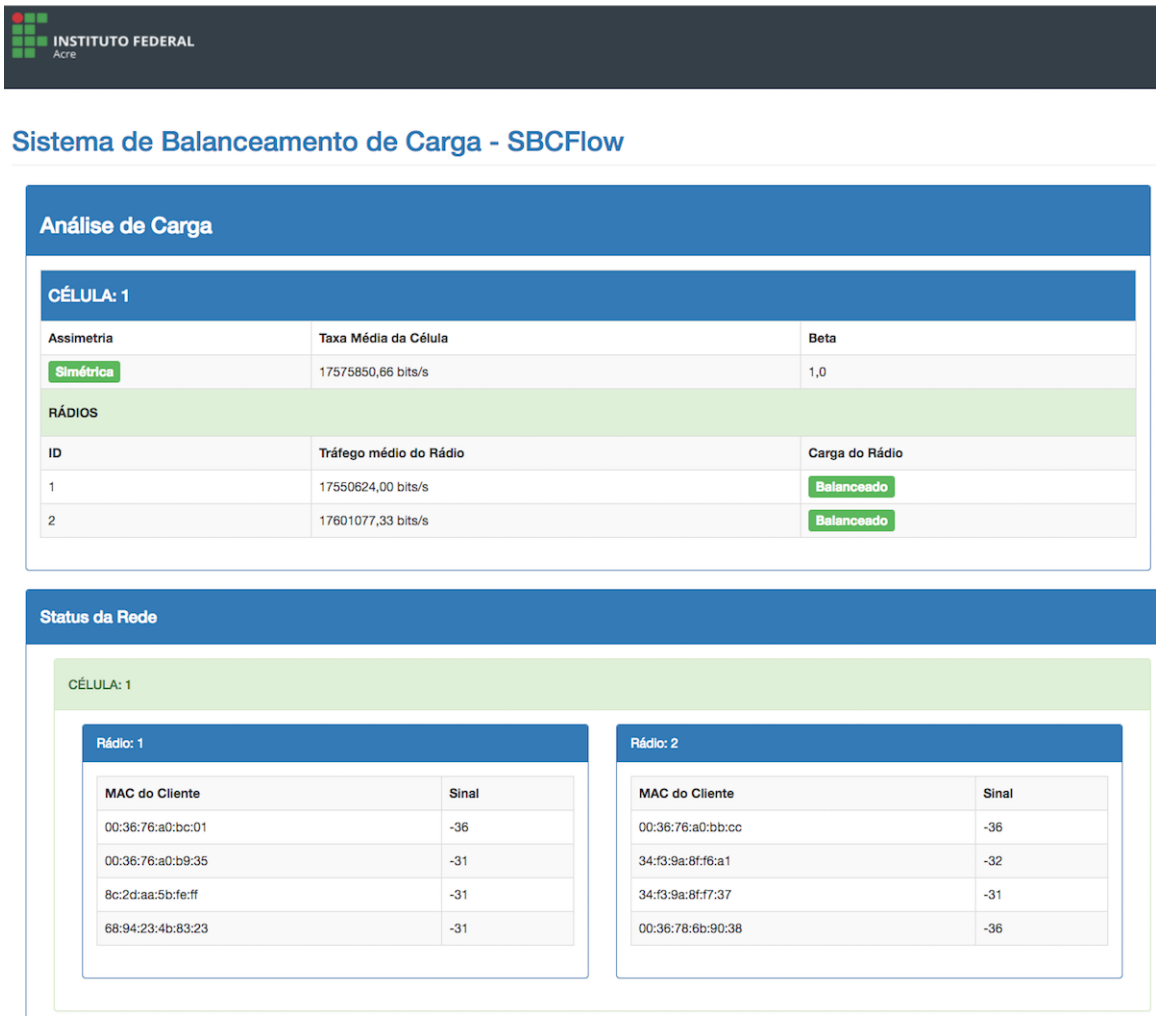


Figura 3.15: Interface WEB - Cenário balanceado.

4

AVALIAÇÃO DOS RESULTADOS

Neste capítulo é apresentado o ambiente onde a solução foi implementada e os testes foram realizados. Cenários foram criados para testar a solução e os resultados foram apresentados ao fim do capítulo.

4.1 Ambiente de Testes

A solução e o ambiente de testes dela foram implementados no Instituto Federal de Educação, Ciência e Tecnologia do Acre (IFAC). Para implementação da solução foram utilizados recursos existentes no IFAC (servidores físicos, infraestrutura existente de rede, etc.), ferramentas e softwares livres (sistema operacional linux Mint, OpenWRT, Open vSwitch, Ryu, OpenFlow, iperf, etc.) e adquiridos equipamentos de baixo custo (Rádios SOHO, adaptadores de rede sem fio, etc.).

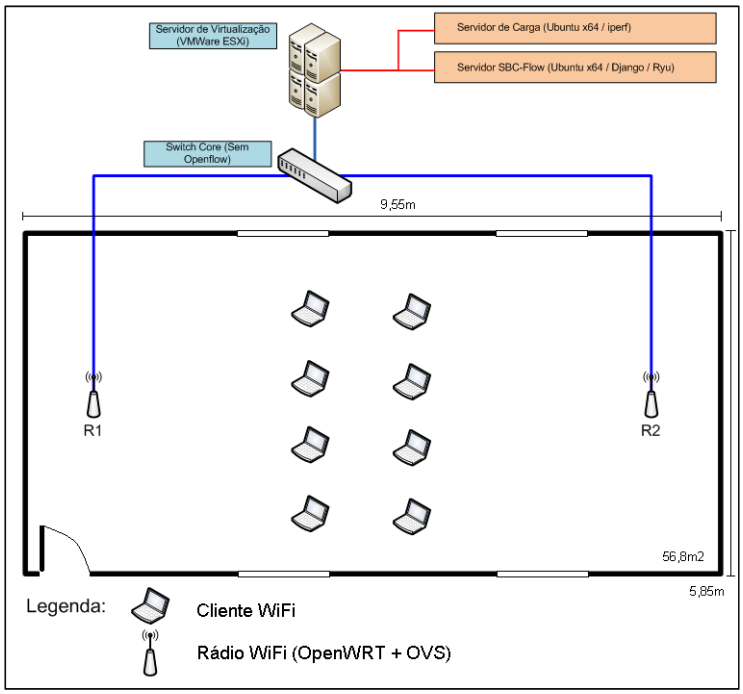


Figura 4.1: Ambiente de teste

A **Figura 4.1** mostra o ambiente de teste e todos os elementos necessários para implantar e testar a solução proposta. Foi utilizada uma sala de aula padrão do IFAC, medindo $56,8m^2$, os APs e os notebooks foram instalados e acomodados na sala em posições fixas, o switch legado foi acomodado em um rack no corredor e o servidor dentro do datacenter do campus.

A posição dos equipamentos utilizados é fixa e inalterada para todos os cenários de teste propostos, em virtude do perfil dos usuários nestes ambientes (estacionários assim que acomodados em suas cadeiras), e todos os APs cobrem toda área física do ambiente de testes. As variáveis dos cenários são a quantidade de carga gerada para cada cliente e seu ponto de comunicação inicial (o AP onde o cliente está ancorado inicialmente).

4.1.1 Detalhes técnicos

Para montar o ambiente foram necessários vários equipamentos, softwares e ferramentas, na **Tabela 4.1** é apresentado tudo que foi utilizado para implementar a solução e realizar os testes.

ITENS	TIPO	QUANTIDADE	LICENCIAMENTO	SITUAÇÃO	CUSTO (R$)
Servidor Rack 2U Dell R710	Equipamento	01	X	Em uso institucional	X
Rádio roteador TP-Link TL-WR1043ND	Equipamento	02	X	Adquirido novo	300,00
Adaptador Wifi USB Ralink	Equipamento	04	X	Adquirido novo	15,00
Notebook Dell Latitude E5430	Equipamento	04	X	Em uso institucional	X
Switch Dell Power Connect 6330P 24P	Equipamento	01	X	Em uso institucional	X
Linux Mint	Sistema Operacional (Software)	X	Livre	X	X
OpenWRT	Linux para embarcados (Software)	X	Livre	X	X
Open vSwitch	Switch Virtual SDN/OpenFlow (Software)	X	Livre	X	X
Ryu	Controlador SDN/OpenFlow (Software)	X	Livre	X	X
Django (Python)	Framework/linguagem de programação (Software)	X	Livre	X	X
VMWare vSphere ESXi 6.0 U2	*Hypervisor bare-metal* (Software)	X	Versão gratuita	X	X
Oracle VirtualBox	*Hypervisor* (Software)	X	Versão gratuita	X	X
PostegreSQL	Gerenciador de Banco de dados (Software)	X	Livre	X	X

Tabela 4.1: Tabela de itens da solução

Inicialmente foi preparado o ambiente das aplicações da solução SBC-Flow, no servidor Dell foi instalado o VMWare vSphere ESXi (VMWARE, 2017) como ambiente de virtualização. Nele foram levantados duas instâncias do sistema operacional (SO) Linux Mint (MINT, 2006), a primeira foi convertida em um servidor de carga com a ferramenta IPerf (IPERF, 2017) e a segunda foi configurada para ser o servidor SBC-Flow. No servidor SBC-Flow foi instalado e configurado o controlador SDN Ryu (COMMUNITY, 2014), o *framework* do Django (DJANGO, 2005) e o sistema de gerenciamento de banco de dados (SGBD) PostgreSQL (POSTGRESQL, 2017) para implementação do módulo de monitoramento e de controle (WEB).

Com as aplicações prontas e configuradas de acordo com os dados necessários de inicialização (configurações de células, rádios, coeficiente β, etc.), o próximo passo foi configurar

os rádios para suportarem o protocolo OpenFlow (MCKEOWN et al., 2008), para isto, foi necessário antes instalar e configurar o OpenWRT (OPENWRT, 2004) neles. Nas soluções apresentadas no capítulo trabalhos relacionados vários autores utilizaram o firmware Pantou (YIAKOUMIS; SCHULZ-ZANDER; ZHU, 2012), o qual é uma versão de OpenWRT modificado com um módulo de suporte ao OpenFlow 1.0 no espaço do usuário, porém, ela limita a vazão do rádio e não suporta versões superiores do OpenFlow. Através de uma metodologia para ganho de desempenho apresentada por (JÚNIOR, 2016) foi possível a instalação do Open vSwitch (OVS, 2016) no espaço do *kernel*, permitindo o AP suportar versões superiores do OpenFlow (até a 1.5) além de obter maior desempenho (suportar maior vazão, maior número de usuários conectados, etc.). O protocolo OpenFlow utilizado nessa solução foi a versão 1.3, por apresentar as especificações necessárias para as análises.

De acordo com (JÚNIOR, 2016), após uma série de testes os resultados alcançados demonstram que a utilização do software switch openflow executado no espaço do usuário não entrega à rede um desempenho aceitável, diante desse contexto a opção do software switch em execução no espaço do kernel, mostrou-se a melhor alternativa para o desenvolvimento de uma solução aplicável em ambiente real. Na **Tabela 4.2** foi descrito os testes e na **Figura 4.2** apresentado o resultado.

Vazão OVS x Switch CPQD						
Switch OpenFlow 1.3	Espaço de Execução	Tipo de Teste	Fluxo Sintético Enviado	Vazão Média alcançada	Protocolo de Transporte	Intervalo de Confiança
OpenVSwitch v2.4	Kernel	Vazão	100Mbps	85,087Mbps	UDP	95%
Switch CPQD	Usuário	Vazão	100Mbps	29,830Mbps	UDP	

Tabela 4.2: Resultado sintético do teste de Vazão para 95% de IC. (JÚNIOR, 2016)

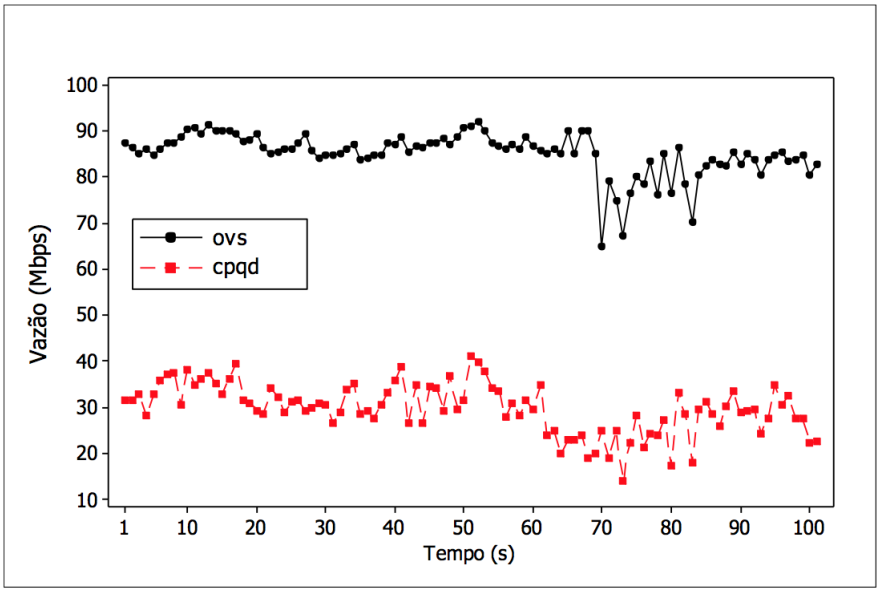


Figura 4.2: Análise de performance entre OpenFlow no kernel e espaço do usuário. (JÚNIOR, 2016)

Com o AP modificado e pronto para comunicar com o controlador, foram criados uma série de *scripts* para facilitar a interação entre os APs, o módulo de monitoramento (dentro do controlador SDN) e o módulo de controle (módulo WEB), criando um canal de comunicação seguro (conexão via SSH), otimizando a consulta a dados nos APs e permitindo a execução de comandos remotos oriundos dos módulos do SBC-Flow.

Após a conclusão de todos os passos anteriores com sucesso, a solução está pronta para ser testada. Para a montagem dos cenários de teste foram necessários 04 (quatro) notebooks, 04 (quatro) adaptadores sem fio usb. Em cada notebook foi instalado um adaptador sem fio usb e através do VirtualBox (ORACLE, 2017) foram virtualizadas 02 (dois) SO Linux Mint em modo *bridge* com as placas de rede sem fio, desta forma cada máquina física representa 02 (dois) clientes sem fio, essa técnica foi utilizada em (LUENGO, 2016) e reproduzida neste ambiente. Após a configuração dos APs todos os quesitos necessários para realização dos testes foram concluídos.

4.1.2 Dificuldades encontradas

Durante a análise e implementação da solução algumas dificuldades foram encontradas, das principais pode-se destacar algumas, como a dificuldade de encontrar informações de como lidar com problemas apresentados pelas ferramentas abertas, normalmente as soluções, quando encontradas, só estão em fóruns de discussão sobre o assunto, além da necessidade de criação de *scripts* para contornar pequenos *bugs* (erros) das ferramentas.

Durante a etapa de testes houve grande dificuldade em reunir a quantidade de equipamentos para simular o ambiente real, em torno de 40 (quarenta) dispositivos móveis, sendo assim foram necessários técnicas como virtualização de clientes e limitação de vazão nos APs para realização dos testes.

Por último, os adaptadores sem fio usb apresentaram um desempenho inferior ao apresentado pelas placas de rede sem fio de marcas reconhecidas no mercado (Dell, HP, Apple, etc.), pelo fato do adaptador não possuir um processador próprio de seus *drivers* precisarem ser emulados pelo processador do computador, e devido a prioridade de outros processos em execução, eventualmente o processador mata o processo do *driver* e o reinicia logo em seguida prejudicando a eficiência do adaptador.

4.1.3 Cenários

Todos os cenários foram definidos em cima do cenário base presente na **Figura 4.1**, como critério foram apresentadas situações assimetria e sobrecarga em ambientes internos, os APs presentes no cenário estão na mesma frequência (2.4Ghz) com tratamento ortogonal de canais (ch1, ch6 e ch11) e com o mesmo SSID (SBC-FLOW). Como os testes foram realizados com x08 (oito) *hosts* (clientes) foi necessário limitar a capacidade de cada rádio a 18Mbits (dezoito) de vazão máxima utilizando o *Traffic Control* (TC).

Do lado do servidor o coeficiente de equidade (β) do SBC-Flow foi configurado para detectar assimetria em qualquer análise onde o resultado seja menor ou igual a **0.96**, este coeficiente varia de 0 à 1 quanto mais próximo de 0 mais desbalanceado está o cenário, este critério é definido pelo administrador da rede de acordo com o nível de balanceamento que é desejado em sua rede. Para os testes, rotina de monitoramento foi configurada para ser executada x01 vez a cada 120 segundos. Cada teste dura 300 segundos e foram repetidos 60 vezes por cenário, 30 vezes sem e 30 vezes com a proposta do SBC-Flow. É interessante ressaltar que o SBC-Flow é uma proposta reativa, ou seja, ela precisa detectar a sobrecarga e em seguida mitiga-la, desta forma, durante os testes com a proposta foram aguardados 30 segundos antes da primeira rotina de monitoramento para que o controlador colete estatísticas sobre o estado atual da rede.

Os clientes foram divididos em dois grupos, os de vazão leve (2.0Mbits) e os de vazão moderada (5.0Mbits), os valores foram baseados no consumo médio de um usuário: para vazão **leve** realizando várias atividades (escutar áudio sob demanda, navegar em redes sociais, etc.); paras vazão **moderada** assistindo um vídeo em *full HD* sob demanda. Este tráfego é injetado sinteticamente pelo servidor de carga (iPerf) no sentido *Downlink* e do tipo UDP para cada cliente. Os cenários estão dispostos de acordo com a **Tabela 4.3**.

Cenários			
Vazão	**80% Moderado 20% Leve**	**20% Moderado 80% Leve**	**50% Moderado 50% Leve**
Ancoragem Inicial Hosts no R1: x08 Hosts no R2: x00	Cenário 01	Cenário 02	Cenário 03

Tabela 4.3: Cenários de teste

4.2 Métricas

Para que a solução seja eficiente ela deve garantir que as aplicações dos usuários possam ser executadas com qualidade, a melhor forma para avaliar sua eficiência é através de métricas de QoS. Para esta solução foram utilizadas as seguintes métricas de QoS:

- **Vazão (*Throughput*)**: é a métrica de QoS resultante da relação entre a taxa de envio sobre o tempo, quanto maior a vazão melhor é a conexão;
- **Perda de Pacote**: é a métrica de QoS resultante da quantidade de pacotes perdidos durante uma transmissão, quanto menor a perda de pacote melhor é a conexão.

4.3 Resultados Obtidos

Nesta seção serão apresentados os resultados dos testes sobre os cenários criados, todos os cenários foram testados 60 vezes, 30 vezes sem a proposta e 30 vezes com a proposta, para avaliar se houve ganho com a utilização da proposta sobre condições de assimetria e sobrecarga na rede. De cada cenário foram destacados dois hosts para serem analisados, um de vazão leve e outro de vazão moderada, e foram gerados gráficos de média das métricas com intervalo de confiança (IC) de 95%.

4.3.1 Cenário 1

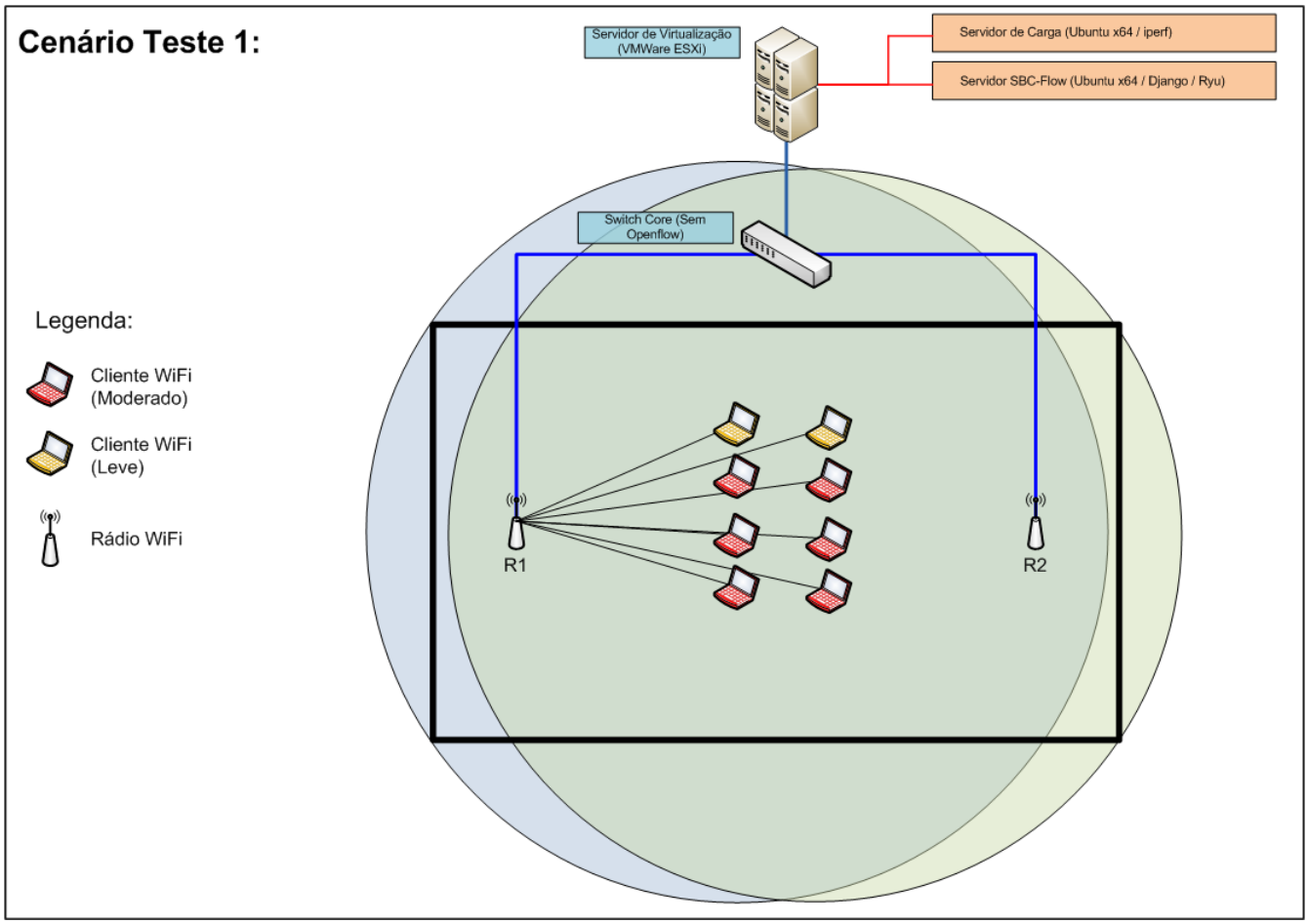


Figura 4.3: Cenário 1

Na **Figura 4.3** podemos observar o cenário 1, todos os clientes (*hosts*) começam conectados no AP01(R1) e dos 08 clientes presentes 06 são moderados (vazão de 5.0Mbits/s) e 02 são leves (vazão de 2.0Mbits/s). Este cenário pode ser considerado o de maior assimetria e sobrecarga do ambiente de testes.

Como pode ser observado na **Figura 4.4** a vazão geral deste cenário sem a proposta era de 16,76Mbits/s e a perda de pacote era de 44%, com a utilização da proposta SBC-Flow a vazão

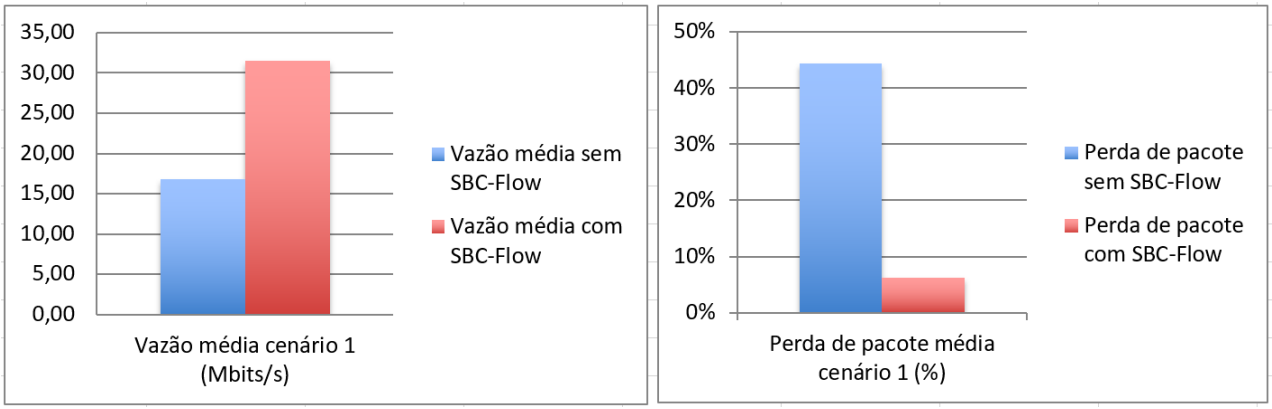
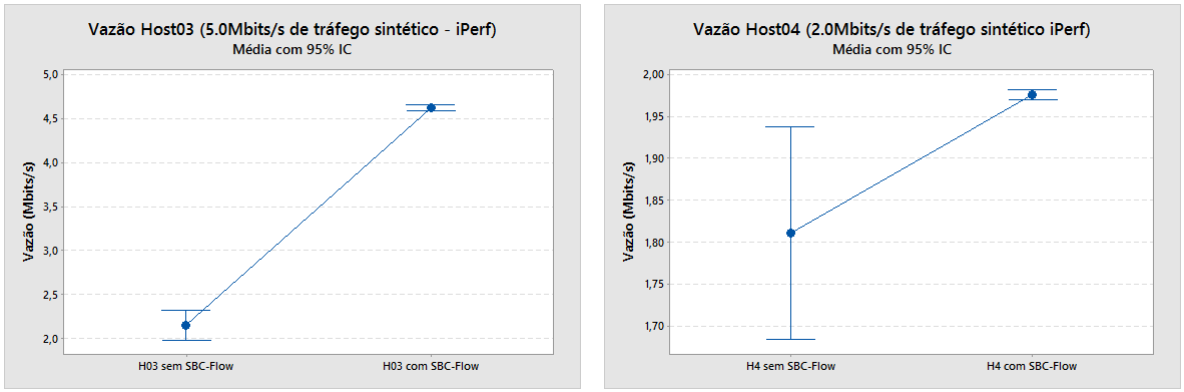


Figura 4.4: Média da vazão e perda de pacotes do cenário 1 com e sem SBC-Flow.

geral foi para 31,55Mbits/s representando um ganho percentual de 46,87% e a perda de pacote caiu para 6,3%, uma redução de 37,7% na perda de pacotes, apresentando um ganho percentual de 85,68% enquanto o SBC-Flow estava ativo. Nas **Figuras 4.5 e 4.6** foram realizados uma análise de média com intervalo de confiança de 95% sobre 02 *hosts* do cenário, um moderado (vazão de 5.0Mbits/s) e um leve (vazão de 2.0Mbits/s).



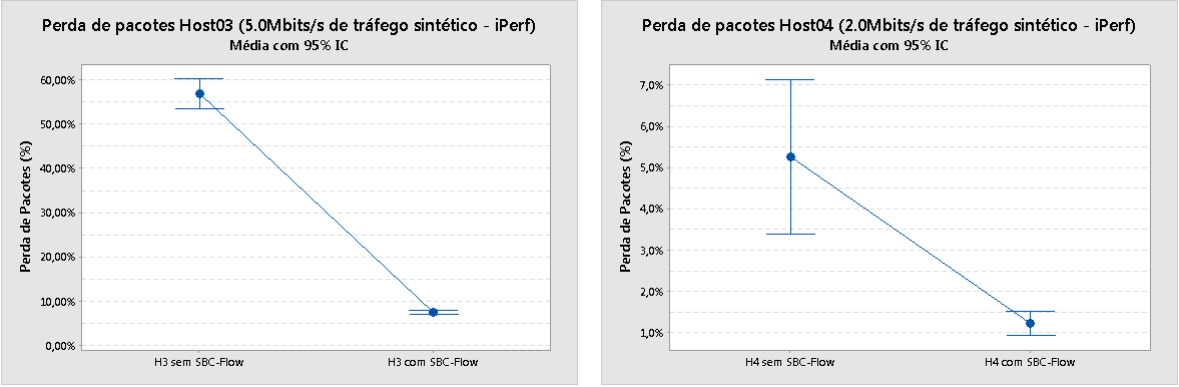
(a) Host 03 moderado (5.0Mbits/s) (b) Host 04 leve (2.0Mbits/s)

Figura 4.5: Média da vazão com 95% IC do Cenário 1

Quando analisamos individualmente estes *hosts* observamos que sem a proposta eles possuíam uma vazão média de 2,15Mbits/s (*host* 03) e 1,81Mbits/s (*host* 04) além de 57% e 9% de perda de pacotes respectivamente. Com a utilização da proposta SBC-Flow a vazão média foi para 4,60Mbits/s (*host* 03) e 1,95Mbits/s (*host* 04), já a perda de pacotes caiu para 7,6% e 1,2% respectivamente, obtendo-se o ganho percentual de 60,86% na vazão média e 86,66% na perda de pacotes do *host* 03 e 7,18% na vazão média e 86,66% na perda de pacotes do *host* 04.

4.3.2 Cenário 2

Na **Figura 4.7** podemos observar o cenário 2, todos os clientes (*hosts*) começam conectados no AP01 (R1) e dos 08 clientes presentes 02 são moderados (vazão de 5.0Mbits/s) e 06 são



(a) Host 03 moderado (5.0Mbits/s) (b) Host 04 leve (2.0Mbits/s)

Figura 4.6: Média da perda de pacotes com 95% IC do Cenário 1

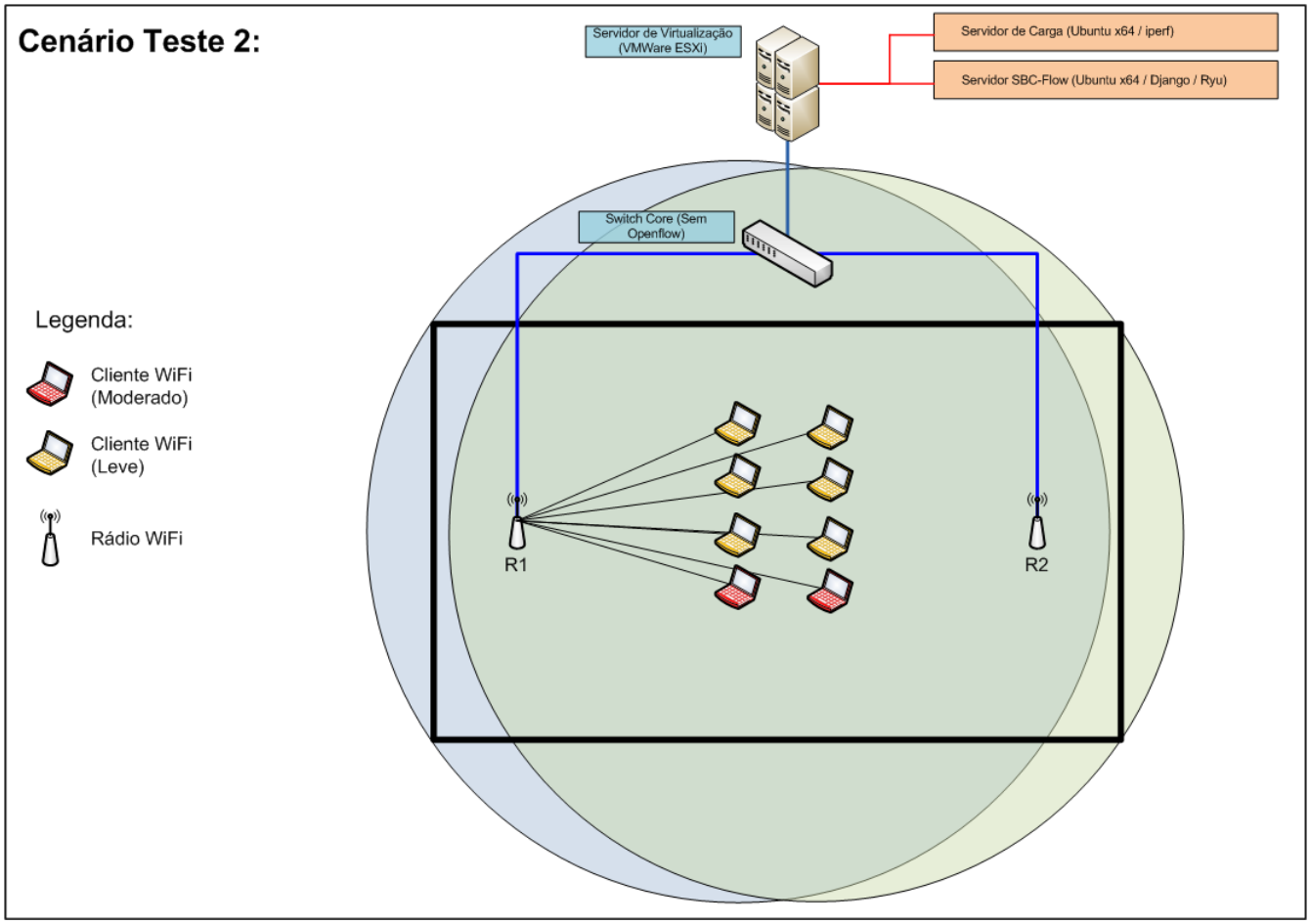


Figura 4.7: Cenário 2

leves (vazão de 2.0Mbits/s). Neste cenário existem vários hosts leves e o nível de sobrecarga é inferior ao do cenário 1.

Como pode ser observado na **Figura 4.8** a vazão geral deste cenário sem a proposta era de 17,57Mbits/s e a perda de pacote era de 11,2%, com a utilização da proposta SBC-Flow a vazão geral foi para 21,22Mbits/s representando um ganho percentual de 17,2% e a perda de pacote

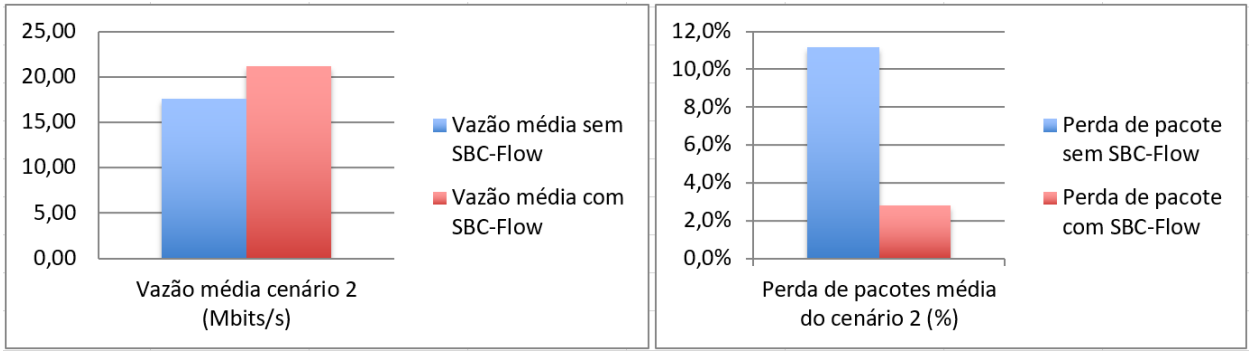
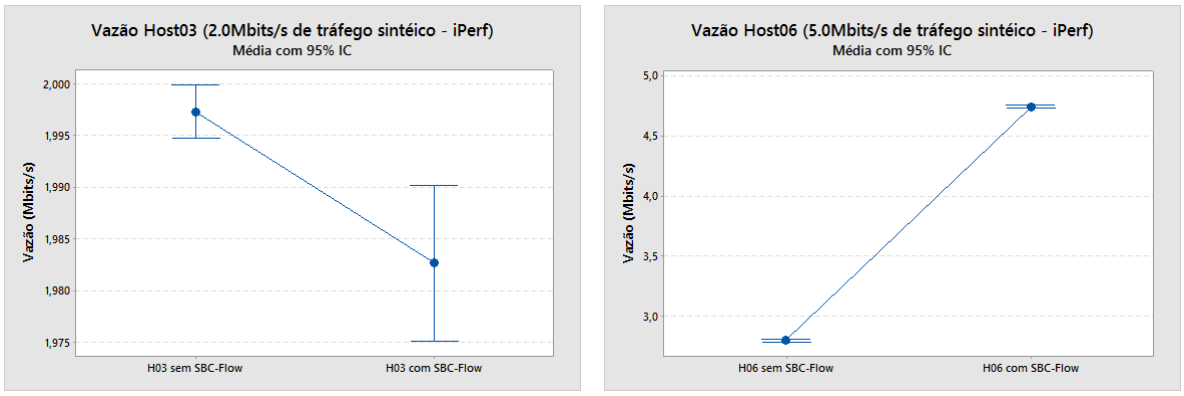


Figura 4.8: Média da vazão e perda de pacotes do cenário 2 com e sem SBC-Flow.

caiu para 2,8%, uma redução de 8,4% na perda de pacotes, apresentando um ganho percentual de 75% enquanto o SBC-Flow estava ativo. Nas **Figuras 4.9 e 4.10** foram realizados uma análise de média com intervalo de confiança de 95% sobre 02 *hosts* do cenário, um moderado (vazão de 5.0Mbits/s) e um leve (vazão de 2.0Mbits/s).

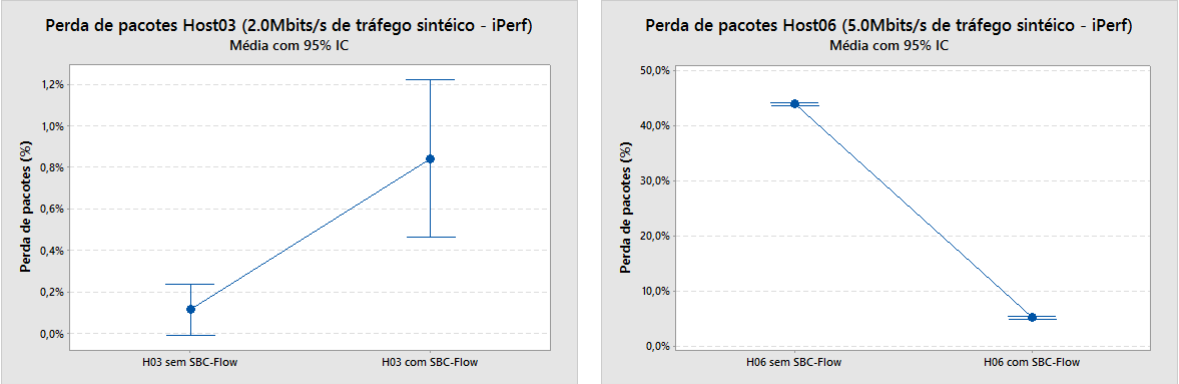


(a) Host 03 leve (2.0Mbits/s)

(b) Host 06 moderado (5.0Mbits/s)

Figura 4.9: Média da vazão com 95% IC do Cenário 2

Este cenário é caracterizado pela grande quantidade de *hosts* leves (2.0Mbits/s), a carga a ser balanceada é menor e estes *hosts* leves não foram tão afetados pela sobrecarga, no entanto, os *hosts* moderados foram muito afetados e apresentaram uma melhoria grande enquanto o SBC-Flow estava ativo. Sem a proposta o *host* 06 apresentava uma vazão média de 2,80Mbits/s e 44% de perda de pacotes, com o SBC-Flow ativo a vazão subiu para 4,74Mbits/s e a perda de pacotes caiu para 5,2%, representando um ganho percentual de 41,35% na vazão média e 81,18% sobre a perda de pacotes.



(a) Host 03 leve (2.0Mbits/s) (b) Host 06 moderado (5.0Mbits/s)

Figura 4.10: Média da perda de pacotes com 95% IC do Cenário 2

4.3.3 Cenário 3

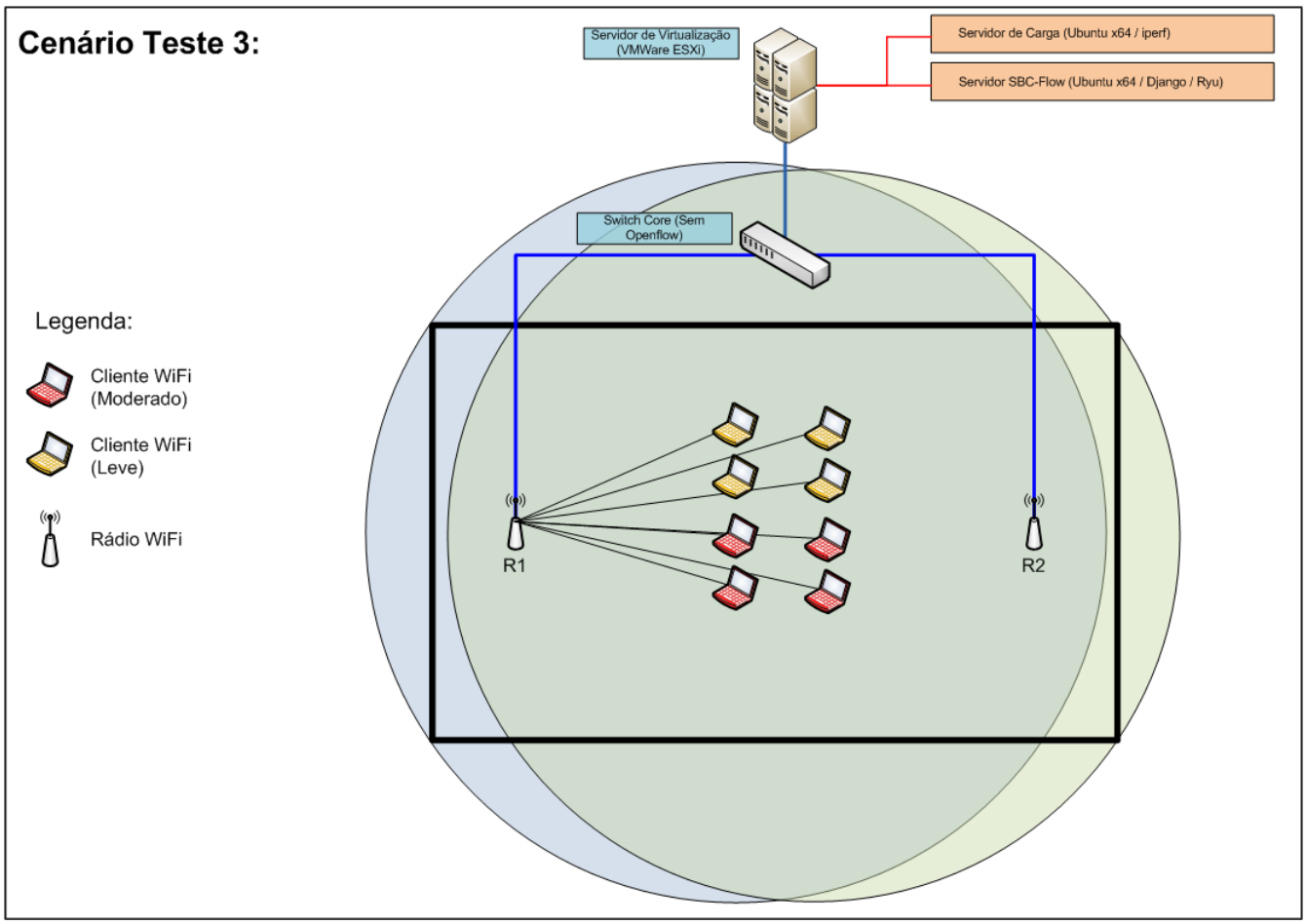


Figura 4.11: Cenário 3

Na **Figura 4.11** podemos observar o cenário 3, todos os clientes (*hosts*) começam conectados no AP01 (R1) e dos 08 clientes presentes 04 são moderados (vazão de 5.0Mbits/s) e 04 são leves (vazão de 2.0Mbits/s). Neste cenário todos os clientes ainda estão inicialmente

conectados no mesmo AP.

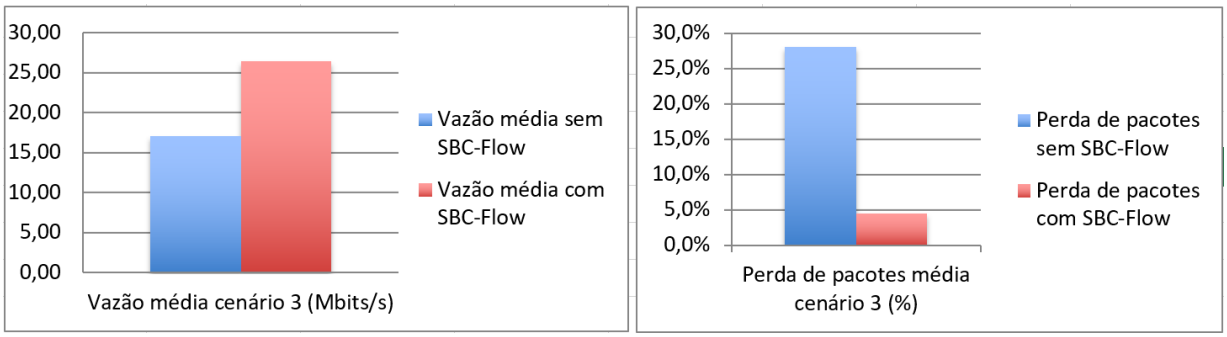
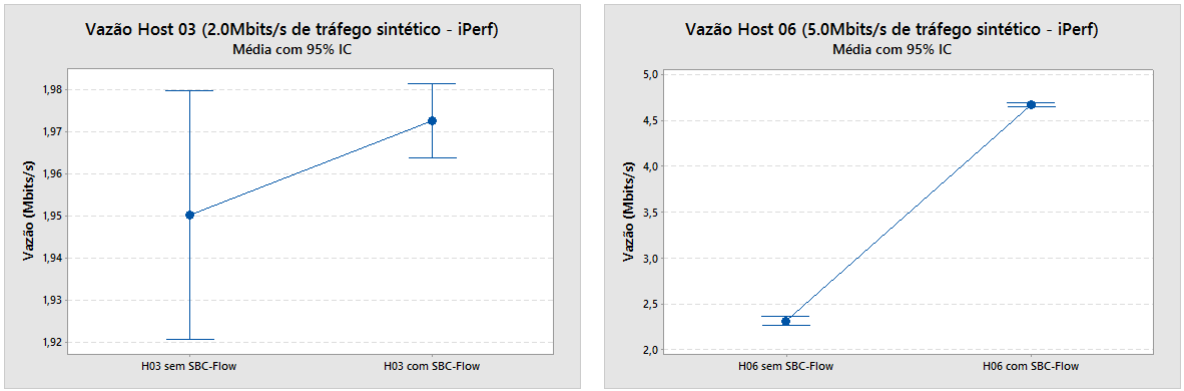


Figura 4.12: Média da vazão e perda de pacotes do cenário 3 com e sem SBC-Flow.

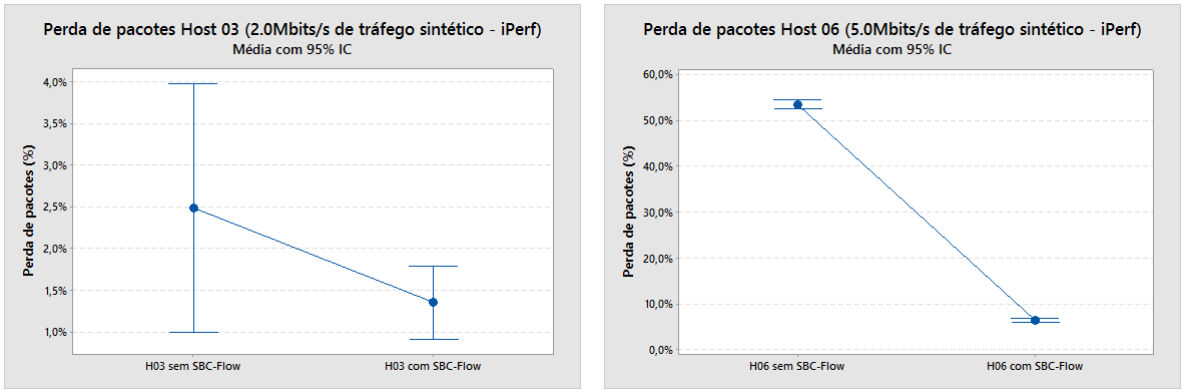
Como pode ser observado na **Figura 4.12** a vazão geral deste cenário sem a proposta era de 17,09Mbits/s e a perda de pacote era de 28%, com a utilização da proposta SBC-Flow a vazão geral foi para 26,42Mbits/s representando um ganho percentual de 35,31% e a perda de pacote caiu para 4,5%, uma redução de 23,5% na perda de pacotes, apresentando um ganho percentual de 83,92% enquanto o SBC-Flow estava ativo. Nas **Figuras 4.13 e 4.14** foram realizados uma análise de média com intervalo de confiança de 95% sobre 02 *hosts* do cenário, um moderado (vazão de 5.0Mbits/s) e um leve (vazão de 2.0Mbits/s).



(a) Host 03 leve (2.0Mbits/s) (b) Host 06 moderado (5.0Mbits/s)

Figura 4.13: Média da vazão com 95% IC do Cenário 3

Percebe-se neste cenário que os *hosts* leves oscilam muito, possuindo momentos de boa e péssima qualidade de conexão, no entanto, os *hosts* moderados foram muito afetados e apresentaram uma melhoria grande enquanto o SBC-Flow estava ativo. Sem a proposta o QoS do *host* 03 (leve) varia bastante, a vazão oscila entre de 1,98 a 1,92Mbits/s obtendo uma média de 1,95Mbits/s e a perda de pacotes oscila entre 1% a 4% obtendo uma média de 2,5%. Quando o SBC-Flow é ativado a oscilação diminui e o *host* 03 obtém uma vazão média de 1,97Mbits/s e a perda de pacotes cai para 1,4%. Quando o *host* 06 (moderado) é analisado observa-se que sem a proposta sua vazão média é de 2,32Mbits/s com 53,6% de perda de pacotes, quando o SBC-Flow é ativado a vazão média sobe para 4,68Mbits/s alcançando um ganho percentual de 50.48%, sua perda de pacotes caiu para 6,5% indicando um ganho percentual de 87,87%.



(a) Host 03 leve (2.0Mbits/s)

(b) Host 06 moderado (5.0Mbits/s)

Figura 4.14: Média da perda de pacotes com 95% IC do Cenário 3

5

CONCLUSÃO E TRABALHOS FUTUROS

5.1 Considerações Finais

Este trabalho apresentou o SBC-Flow, um arcabouço para o balanceamento de carga em redes sem fio com uso do paradigma de SDN. Foram apresentados conceitos sobre as áreas de redes sem fio e SDN e realizado um estudo sobre os diversos trabalhos apresentados nas áreas de balanceamento de carga em redes sem fio e SDN em redes sem fio, além de alguns em balanceamento de carga em redes sem fio definidas por software. No contexto do SBC-Flow, foi apresentada a sua arquitetura e descrito todas as suas fases (monitoramento, análise de carga, análise e seleção de usuários e migração) e módulos (módulo de monitoramento e módulo de controle). Como prova de conceito a aplicação SBC-Flow foi desenvolvida, cenários de assimetria e congestionamento foram criados e testes foram realizados. Como pôde ser observado na avaliação da proposta, a solução SBC-Flow obteve êxito em seu propósito, atingindo todos os objetivos traçados inicialmente, além de alcançar ótimos resultados, o que foi comprovado através da análise de métricas de QoS sobre os cenários criados. No cenário 1 houve um ganho geral de **46,87%** sobre a vazão e **85,68%** sobre a perda de pacotes, no cenário 2 houve um ganho geral de **17,2%** sobre a vazão e **75%** sobre a perda de pacotes e no cenário 3 houve um ganho geral de **35,31%** sobre a vazão e **83,92%** sobre a perda de pacotes.

5.2 Trabalhos Futuros

O SBC-Flow realiza o balanceamento de carga em redes sem fio definidas por software, através da realocação dos usuários que estão gerando a carga excedente. No entanto, apenas os aspectos de sinal e vazão do usuário são analisados. Como trabalho futuro seria interessante criar um critério de peso onde o tráfego dos usuários fosse classificado, funcionando como mais um critério na decisão da migração, este critério evitaria que usuários considerados estáveis (usuários que tem sua carga acomodada adequadamente, mesmo estando conectado a um rádio sobrecarregado) fossem migrados. Outro aspecto interessante para um trabalho futuro seria um tratamento adequado no *handover* para evitar a quebra, como o apresentado no (JÚNIOR, 2016),

isso reduziria drasticamente a perda de pacote durante o processo de migração e suavizaria a transição dos usuários enquanto migram de um rádio para o outro. Por último, um aspecto de melhoria a ser implementado, seria a expansão da interface WEB para a configuração das informações estáticas dos rádios através de um ponto centralizado, como SSID, canal de comunicação, potência da antena do rádio, dentre outras informações.

Referências

BALACHANDRAN, A.; BAHL, P.; VOELKER, G. M. Hot-spot congestion relief in public-area wireless networks. In: MOBILE COMPUTING SYSTEMS AND APPLICATIONS, 2002. PROCEEDINGS FOURTH IEEE WORKSHOP ON. **Anais...** [S.l.: s.n.], 2002. p.70–80.

BEJERANO, Y.; HAN, S.-J.; LI, L. E. Fairness and load balancing in wireless LANs using association control. In: MOBILE COMPUTING AND NETWORKING, 10. **Proceedings...** [S.l.: s.n.], 2004. p.315–329.

BERNASCHI, M. et al. OpenCAPWAP: an open source capwap implementation for the management and configuration of wifi hot-spots. **Computer Networks**, [S.l.], v.53, n.2, p.217–230, 2009.

BRICKLEY, O.; REA, S.; PESCH, D. Load balancing for QoS enhancement in IEEE802. 11e WLANs using cell breathing techniques. In: IFIP INTERNATIONAL CONFERENCE ON MOBILE AND WIRELESS COMMUNICATIONS NETWORKS, MAROC, 7. **Anais...** [S.l.: s.n.], 2005.

CHAUDET, C.; HADDAD, Y. Wireless software defined networks: challenges and opportunities. In: MICROWAVES, COMMUNICATIONS, ANTENNAS AND ELECTRONICS SYSTEMS (COMCAS), 2013 IEEE INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2013. p.1–5.

CHIU, D.-M.; JAIN, R. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. **Computer Networks and ISDN systems**, [S.l.], v.17, n.1, p.1–14, 1989.

CISCO, S. **Wireless LAN Controller**. [Online; accessed 02-October-2015], `http://www.cisco.com/c/en/us/products/wireless/wireless-lan-controller/index.html`.

COMMUNITY, R. S. F. **WHAT'S RYU?** [Online; accessed 28-October-2016], `https://osrg.github.io/ryu/index.html`.

DELL, N. **Scalable mobility controllers and access points**. [Online; accessed 02-October-2015], `http://www.dell.com/us/business/p/powerconnect-w-controllers/pd`.

DELY, P. et al. Cloudmac—an Openflow based architecture for 802.11 MAC layer processing in the cloud. In: IEEE GLOBECOM WORKSHOPS, 2012. **Anais...** [S.l.: s.n.], 2012. p.186–191.

DJANGO, S. F. **The web framework for perfectionists with deadlines.** [Online; accessed 08-march-2017], `https://www.djangoproject.com/`.

GUEDES, D. et al. Redes Definidas por Software: uma abordagem sistêmica para o desenvolvimento de pesquisas em redes de computadores. **Minicursos do Simpósio Brasileiro de Redes de Computadores-SBRC 2012**, [S.l.], v.30, n.4, p.160–210, 2012.

HP, H. P. **Switches/controladores de WLAN**. [Online; accessed 02-October-2015], http://www8.hp.com/br/pt/products/networking-wireless/product-detail.html?oid=3963981#!tab=features.

IEEE. IEEE Standard for Information technology–Telecommunications and information exchange between systems Local and metropolitan area networks–Specific requirements Part 11: wireless lan medium access control (mac) and physical layer (phy) specifications - redline. **IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007) - Redline**, [S.l.], p.1–5229, March 2012.

IEEE. IEEE Standard for Information technology– Telecommunications and information exchange between systems Local and metropolitan area networks– Specific requirements–Part 11: wireless lan medium access control (mac) and physical layer (phy) specifications–amendment 4: enhancements for very high throughput for operation in bands below 6 ghz. **IEEE Std 802.11ac-2013 (Amendment to IEEE Std 802.11-2012, as amended by IEEE Std 802.11ae-2012, IEEE Std 802.11aa-2012, and IEEE Std 802.11ad-2012)**, [S.l.], p.1–425, Dec 2013.

IPERF, E. **iPerf - The ultimate speed test tool for TCP, UDP and SCTP.** [Online; accessed 10-may-2017], https://iperf.fr/.

JÚNIOR, E. C. d. A. Wi-Flow: uma arquitetura baseada em sdn para o gerenciamento e mobilidade em redes wi-fi com suporte à autenticação 802.1 x. , [S.l.], 2016.

KHONDOKER, R. et al. Feature-based comparison and selection of Software Defined Networking (SDN) controllers. In: COMPUTER APPLICATIONS AND INFORMATION SYSTEMS (WCCAIS), 2014 WORLD CONGRESS ON. **Anais...** [S.l.: s.n.], 2014. p.1–7.

LINS, T. **Redes Definidas Por Software (Software Defined Networks) SDN**. [Online; accessed 28-October-2016], http://www.decom.ufop.br/imobilis/redes-definidas-por-software-software-defined-networks-sdn/.

LUENGO, E. **An openflow-based wireless user management system**. 2016. Tese (Doutorado em Ciência da Computação) — University of Ontario Institute of Technology.

MCKEOWN, N. et al. OpenFlow: enabling innovation in campus networks. **ACM SIGCOMM Computer Communication Review**, [S.l.], v.38, n.2, p.69–74, 2008.

MINT, L. **From freedom came elegance.** [Online; accessed 10-may-2017], https://www.linuxmint.com/.

MOURA, H. et al. Ethanol: software defined networking for 802.11 wireless networks. In: IFIP/IEEE INTERNATIONAL SYMPOSIUM ON INTEGRATED NETWORK MANAGEMENT (IM), 2015. **Anais...** [S.l.: s.n.], 2015. p.388–396.

NICHOLSON, A. J. et al. Improved access point selection. In: MOBILE SYSTEMS, APPLICATIONS AND SERVICES, 4. **Proceedings...** [S.l.: s.n.], 2006. p.233–245.

NUNES, B. et al. A Survey of Software-Defined Networking: past, present, and future of programmable networks. **Communications Surveys Tutorials, IEEE**, [S.l.], v.16, n.3, p.1617–1634, Third 2014.

ONF, O. N. F. **Software-Defined Networking (SDN) Definition**. [Online; accessed 14-february-2017], https://www.opennetworking.org/sdn-resources/sdn-definition.

OPENWRT, P. **OpenWRT Wireless Freedom**. [Online; accessed 28-October-2016], https://openwrt.org/.

ORACLE, V. **VirtualBox - Welcome to VirtualBox.org!** [Online; accessed 11-may-2017], https://www.virtualbox.org/.

OVS, A. L. F. C. P. **Open vSwitch**. [Online; accessed 14-february-2017], http://openvswitch.org/.

PFAFF, B.; DAVIE, B. **The open vSwitch database management protocol**. [S.l.: s.n.], 2013.

POSTGRESQL. **PostgreSQL - The world's most advanced open source database.** [Online; accessed 10-may-2017], https://www.postgresql.org/.

SCHULZ-ZANDER, J. et al. Programmatic orchestration of wifi networks. In: USENIX ANNUAL TECHNICAL CONFERENCE (USENIX ATC 14), 2014. **Anais...** [S.l.: s.n.], 2014. p.347–358.

SHIMPI, A. L. **5th Generation WIFI**: 802.11ac, gigabit wifi primer. [Online; accessed 12-October-2015], http://www.anandtech.com/show/5292/80211ac-gigabit-wifi-primer.

STANLEY, D.; CALHOUN, P.; MONTEMURRO, M. **Control and provisioning of wireless access points (CAPWAP) protocol specification**. [S.l.: s.n.], 2009.

SURESH, L. et al. Towards programmable enterprise WLANS with Odin. In: HOT TOPICS IN SOFTWARE DEFINED NETWORKS. **Proceedings...** [S.l.: s.n.], 2012. p.115–120.

TELECO. **LAN / MAN Wireless I**: padrões 802.11 a, b e g. [Online; accessed 10-July-2016], http://www.teleco.com.br/tutoriais/tutorialrwlanman1/pagina_3.asp.

ULC, S. **Sandvine global Internet phenomena Report-1h2014**. [S.l.]: Technical report, 2014.

VELAYOS, H.; ALEO, V.; KARLSSON, G. Load balancing in overlapping wireless LAN cells. In: COMMUNICATIONS, 2004 IEEE INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2004. v.7, p.3833–3836.

VMWARE, I. **vSphere Hypervisor**. [Online; accessed 10-may-2017], http://www.vmware.com/br/products/vsphere-hypervisor.html.

YAMAHATA, I. Ryu: sdn framework and python experience. In: **Anais...** PYCON APAC: September 14, 2013. p.1–43.

YEN, L.-H.; YEH, T.-T. SNMP-based approach to load distribution in IEEE 802.11 networks. In: IEEE 63RD VEHICULAR TECHNOLOGY CONFERENCE, 2006. **Anais...** [S.l.: s.n.], 2006. v.3, p.1196–1200.

YEN, L.-H.; YEH, T.-T.; CHI, K.-H. Load balancing in IEEE 802.11 networks. **IEEE Internet Computing**, [S.l.], v.13, n.1, p.56–64, 2009.

YIAKOUMIS, Y.; SCHULZ-ZANDER, J.; ZHU, J. **Pantou**: openflow 1.0 for openwrt@ online. 2012.

Apêndice

A

Código fonte do módulo de monitoramento

Módulo de monitoramento - Aplicação openflow SBC-Flow

```
# _*_ coding:utf-8 _*_
from __future__ import division
from operator import attrgetter
import paramiko
import psycopg2
import ast

from ryu.app import simple_switch_13
from ryu.controller import ofp_event
from ryu.controller.handler import MAIN_DISPATCHER, DEAD_DISPATCHER
from ryu.controller.handler import set_ev_cls
from ryu.lib import hub
from time import sleep
from decimal import *
import json
from datetime import datetime

#Cria os Fluxos referentes ao switchs e usuarios dinamicamente conforme vao se conectando
class SbcFlowMonitor(simple_switch_13.SimpleSwitch13):
    #Declara as variaveis globais e inicializa a rotina de monitoramento atraves do hub.spawn
    def __init__(self, *args, **kwargs):
        super(SbcFlowMonitor, self).__init__(*args, **kwargs)
        self.datapaths = {}
        self.s = 0
        self.monitor_thread = hub.spawn(self._monitor)
        #self.trafego_porta = {}
        self.fluxo = {}
        #self.amostra_porta = 1
        self.amostra_fluxo = 1
        self.ipswitch = ''
        self.logger.info('                                                              ')
        self.logger.info('--------------------------------------------------------------')
        self.logger.info('- Inicializando Sistema de Balaneamento de Carga - SBC-Flow -')
        self.logger.info('--------------------------------------------------------------')
        self.logger.info('                                                              ')

    #Monitora a entrada e saida dos switchs no controlador, adicionando e removendo as instancias de monitoramento
    @set_ev_cls(ofp_event.EventOFPStateChange,[MAIN_DISPATCHER, DEAD_DISPATCHER])
    def _state_change_handler(self, ev):
        datapath = ev.datapath
        if ev.state == MAIN_DISPATCHER:
            if not datapath.id in self.datapaths:
                self.logger.info('Adicionando datapath: %016x ao monitoramento.', datapath.id)
                self.datapaths[datapath.id] = datapath
            elif ev.state == DEAD_DISPATCHER:
                if datapath.id in self.datapaths:
                    self.logger.info('Removendo datapath: %016x do monitoramento.', datapath.id)
                    del self.datapaths[datapath.id]
```

Continuação...

```
    #Rotina ciclica de solicitacao de estatistica de switchs instanciados
    def _monitor(self):
        while True:
            for dp in self.datapaths.values():
                for d in range(1,5):
                    self.logger.info('DpID: %s / Ciclo de Fluxo: %d', dp.id, d)
                    self._request_flow(dp) #Invoca rotina de requisicao de dados dos fluxos dos switchs
                    sleep(1)
                conn = psycopg2.connect("dbname='teste' user='postgres' host='localhost' password='postgressbc'")
                dthr_atual = datetime.now()
                cur = conn.cursor()
                cur.execute("""UPDATE CONTROLADOR SET data_atualizacao = (TIMESTAMP '%s');""" % str(dthr_atual))
                conn.commit()
                conn.close()
            hub.sleep(600)

    #Monta requisicao de fluxo e envia ao controlador
    def _request_flow(self, datapath):
        self.logger.info('Solicitacao de Fluxo do Switch: %016x', datapath.id)
        try:
            conn = psycopg2.connect("dbname='teste' user='postgres' host='localhost' password='postgresscc'")
        except:
            self.logger.info('Conexao com BD nao realizada')

        cur = conn.cursor()
        try:
            cur.execute("""SELECT IPSWITCH FROM SWITCH WHERE IDCELULA=%s""", str(datapath.id))
            rows = cur.fetchall()
            for row in rows:
                self.ipswitch = str(row)[2:-3]
        except:
            self.logger.info('Nao encontrou Switch!')

        conn.close()

        parser = datapath.ofproto_parser
        req = parser.OFPFlowStatsRequest(datapath)
        datapath.send_msg(req)


    '''Monitora as respostas de requisicao de fluxo do controlador e coleta os dados necessarios para calculo de
       ↪    trafego medio de cada fluxo'''
    @set_ev_cls(ofp_event.EventOFPFlowStatsReply, MAIN_DISPATCHER)
    def _flow_stats_reply_handler(self, ev):
        body = ev.msg.body

        for stat in sorted([flow for flow in body if flow.priority == 1], key=lambda flow: (flow.match['in_port'],
           ↪    flow.match['eth_dst'])):
            if stat.instructions[0].actions[0].port == 1:
                if not self.fluxo.get(stat.match['eth_dst']):
                    self.fluxo[stat.match['eth_dst']] = {'txi': stat.byte_count, 'dsi': stat.duration_sec}
                else:
                    self.fluxo[stat.match['eth_dst']].update({'tx' + str(self.amostra_fluxo): stat.byte_count, 'ds' +
                       ↪    str(self.amostra_fluxo): stat.duration_sec})
        self.amostra_fluxo += 1

        if self.amostra_fluxo == 5:
            '''Realiza conexao com o Radio(switch) cliente e requisita os clientes conectados a ele naquele
               ↪    momento'''
            cliente = paramiko.SSHClient()
            cliente.set_missing_host_key_policy(paramiko.AutoAddPolicy())
            cliente.connect(str(self.ipswitch), username='root', password='root')
            stdin, stdout, stderr = cliente.exec_command('./clientes.sh')
            data = stdout.read().splitlines()
            cliente.close()
            lista = []
            for line in data:
                lista.append(ast.literal_eval(line))
            #Criacao do dicionario que sera avaliado para calculo do fluxo do radio
            fluxo_derivado = {}
            #Verifica se todos os fluxos recebidos pelo controlador sao referentes aos clientes conectados ao radio
               ↪    no momento
            for f in self.fluxo:
                for l in lista:
                    if l['mac'] == f:
                        fluxo_derivado[f] = self.fluxo[f]
                        fluxo_derivado[f].update({'sinal': l['sinal']})
            #self.logger.info('fluxo derivado: %s', fluxo_derivado)
            self._imprime_fluxo(ev.msg.datapath.id, fluxo_derivado) #Envia os dados para a funcao que calcula e
               ↪    mostra os resultados
            self.amostra_fluxo = 1
```

Continuação...

```
    #Calcula e disponibiliza os resultados do trafego medio de cada fluxo para a proxima fase de Balanceamento
    def _imprime_fluxo(self, datapath, fluxo):
        trafego_fluxo = {}

        try:
            conn = psycopg2.connect("dbname='teste' user='postgres' host='localhost' password='postgresscc'")
            cur = conn.cursor()
        except:
            self.logger.info('Conexao com BD nao realizada')

        try:
            cur.execute("""DELETE FROM USUARIO WHERE SWITCH_ID=%s;""", str(datapath))
            conn.commit()
        except:
            self.logger.info('Nao possui registros para apagar!')

        self.logger.info('-----------------')
        self.logger.info('Datapath - Switch')
        self.logger.info('-----------------')
        self.logger.info('%016x', datapath)
        self.logger.info('----------------- -------------- -------------- -------------- -------------- ------')
        self.logger.info('MAC (Clientes)    trafego_medio  taxa_2         taxa_3         taxa_4         Sinal')
        self.logger.info('----------------- -------------- -------------- -------------- -------------- ------')

        for s in fluxo:
            trx2 = self._calcula_trafego(fluxo, s, 'txi', 'dsi', 'tx2', 'ds2')
            trx3 = self._calcula_trafego(fluxo, s, 'txi', 'dsi', 'tx3', 'ds3')
            trx4 = self._calcula_trafego(fluxo, s, 'txi', 'dsi', 'tx4', 'ds4')
            tm = (trx2 + trx3 + trx4) / 3
            self.logger.info('%017s  %3.2f Mbits/s   %3.2f Mbits/s   %3.2f Mbits/s   %3.2f Mbits/s   %s', s,
                ↪   self._megabit(tm), self._megabit(trx2), self._megabit(trx3), self._megabit(trx4),
                ↪   str(fluxo[s]['sinal']))
            self.logger.info('----------------- -------------- -------------- -------------- -------------- ------')
            trafego_fluxo[s] = {'tm': tm, 'sinal': fluxo[s]['sinal']}
            '''Grava no banco de dados os dados referentes aos usuarios do radio em questao'''
            try:
                cur.execute("""INSERT INTO USUARIO (macadress, switch_id, tu, sinal, migracao) VALUES ('%s', %d, %s,
                    ↪   %d, false);""" % (str(s), datapath, Decimal(tm).quantize(Decimal(10) ** -2),
                    ↪   int(fluxo[s]['sinal'])) )
            except:
                self.logger.info('Nao encontrou Switch! %s - %d - %s - %d' % (str(s), datapath,
                    ↪   str(Decimal(tm).quantize(Decimal(10) ** -2)), int(fluxo[s]['sinal'])))
        conn.commit()
        conn.close()
        f = open('/home/logscc/tu_' + str(datapath) + '.json', 'w')
        f.write(json.dumps(trafego_fluxo))
        f.close()
        self.fluxo = {}

    #Funcao que calcula o trafego medio de fluxos
    def _calcula_trafego(self, trafego, s, txi, dsi, txf, dsf):
        try:
            trx = (trafego[s][txf] - trafego[s][txi]) * 8 / (trafego[s][dsf] - trafego[s][dsi])
        except Exception:
            trx = 0
        return trx

    #Converte para Megabit por segundo
    def _megabit (self, tm):
        try:
            tr = round((tm / 1048576), 2)
        except Exception:
            tr = 0
        return tr
```

B

Scripts para automatizar as funções nos rádios

Coleta de clientes

```
for interface in `iw dev | grep Interface | cut -f 2 -s -d" "`
do
  maclist=`iw dev $interface station dump | grep Station | cut -f 2 -s -d" "`
  for mac in $maclist
  do
    ip="UNKN"
    host=""
    ip=`cat /tmp/dhcp.leases | cut -f 2,3,4 -s -d" " | grep $mac | cut -f 2 -s -d" "`
    signal=`iw dev $interface station get $mac | grep avg | cut -c14-16`
    host=`cat /tmp/dhcp.leases | cut -f 2,3,4 -s -d" " | grep $mac | cut -f 3 -s -d" "`
    echo -e "{'mac': '$mac', 'sinal': '$signal'}"
  done
done
```

Controle de vazão do Rádio

```
tc qdisc add dev wlan0 root handle 1: htb
tc class add dev wlan0 parent 1: classid 1:1 htb rate 18000kbit
tc class add dev wlan0 parent 1:1 classid 1:10 htb rate 18000kbit
tc class add dev wlan0 parent 1:1 classid 1:20 htb rate 18000kbit
tc qdisc add dev wlan0 parent 1:10 handle 10: sfq perturb 10
tc qdisc add dev wlan0 parent 1:20 handle 20: sfq perturb 10
tc filter add dev wlan0 protocol ip priority 1 u32 match ip protocol 1 0xff flowid 1:10
tc filter add dev wlan0 protocol ip priority 1 u32 match ip protocol 17 0xff flowid 1:10
tc filter add dev wlan0 protocol ip priority 1 u32 match ip protocol 1 0xff flowid 1:20
tc filter add dev wlan0 protocol ip priority 1 u32 match ip protocol 17 0xff flowid 1:20
tc qdisc show dev wlan0
tc class show dev wlan0
tc filter show dev wlan0
```

Configuração do Open vSwitch no rádio

```
ovs-vsctl del-port r1 wlan0
ovs-vsctl del-port r1 eth1.3
ovs-vsctl add-port r1 wlan0 -- set Interface wlan0 ofport_request=1
ovs-vsctl add-port r1 eth1.3 -- set Interface eth1.3 ofport_request=2
ovs-vsctl show
```

C

Roteiro de configuração dos rádios

Versão do Rádio: TP-LINK 1043ND V2
Configuração Rádio R1(r1)

- Instalar versão 15.05 OpenWRT
- Instalar OpenVSwitch (opkg install openvswitch)
- Instalar Hostapd CLI (opkg hostapd-utils)
- Instalar o Traffic Control e HTB (opkg install kmod-sched tc)
- Configurar o Rádio (vim /etc/config/networks)

Configurar o Rádio (vim /etc/config/networks)

```
config interface 'loopback'
        option ifname 'lo'
        option proto 'static'
        option ipaddr '127.0.0.1'
        option netmask '255.0.0.0'

config globals 'globals'
        option ula_prefix 'fd03:66be:e87d::/48'

config interface 'lan'
        option force_link '1'
        option type 'bridge'
        option proto 'static'
        option netmask '255.255.255.0'
        option ip6assign '60'
        option _orig_ifname 'eth1 wlan0'
        option _orig_bridge 'true'
        option ifname 'eth1'
        option ipaddr '10.10.10.1'

config interface 'lan1'
        option ifname 'eth1.3'
        option proto 'static'

config interface 'wlan0'
        option ifname 'wlan0'
        option proto 'static'

config interface 'wan'
        option ifname 'eth0'
        option proto 'dhcp'

config interface 'wan6'
        option ifname 'eth0'
        option proto 'dhcpv6'

config switch
        option name 'switch0'
        option reset '1'
        option enable_vlan '1'
        option enable_learning '0'

config switch_vlan
        option device 'switch0'
        option vlan '1'
        option ports '0 2 3 4'

config switch_vlan
        option device 'switch0'
        option vlan '2'
        option ports '5 6'

config switch_vlan
        option device 'switch0'
        option vlan '3'
        option ports '1 0t'
```

- Configurar o Rádio (vim /etc/config/firewall)

Configurar o Rádio (vim /etc/config/firewall)

```
config defaults
        option syn_flood        1
        option input            ACCEPT
        option output           ACCEPT
        option forward          ACCEPT
```

- Configurar o OVS no rádio

Configurar o OVS no rádio

```
ovs-vsctl add-br r1

ovs-vsctl set bridge r1 other-config:hwaddr=00:00:00:00:00:01

ovs-vsctl set bridge r1 protocols=OpenFlow10,OpenFlow13

ovs-vsctl set-controller r1 tcp:10.10.10.200:6633

ovs-vsctl set controller r1 connection-mode=out-of-band

ovs-vsctl set bridge r1 fail_mode=secure

ovs-vsctl add-port r1 wlan0 -- set Interface wlan0 ofport_request=1

ovs-vsctl add-port r1 eth1.3 -- set Interface eth1.3 ofport_request=2
```

- Configurar o wifi no rádio, setar wlan0 para interface wifi

Configuração Rádio R2(r2)

- Instalar versão 15.05 OpenWRT
- Instalar OpenVSwitch (opkg install openvswitch)
- Instalar Hostapd CLI (opkg hostapd-utils)
- Instalar o Traffic Control e HTB (opkg install kmod-sched tc)
- Configurar o Rádio (vim /etc/config/networks)

Configurar o Rádio (vim /etc/config/networks)

```
config interface 'loopback'
        option ifname 'lo'
        option proto 'static'
        option ipaddr '127.0.0.1'
        option netmask '255.0.0.0'

config globals 'globals'
        option ula_prefix 'fd03:66be:e87d::/48'

config interface 'lan'
        option force_link '1'
        option type 'bridge'
        option proto 'static'
        option netmask '255.255.255.0'
        option ip6assign '60'
        option _orig_ifname 'eth1 wlan0'
        option _orig_bridge 'true'
        option ifname 'eth1'
        option ipaddr '10.10.10.2'

config interface 'lan1'
        option ifname 'eth1.3'
        option proto 'static'

config interface 'wlan0'
        option ifname 'wlan0'
        option proto 'static'

config interface 'wan'
        option ifname 'eth0'
        option proto 'dhcp'

config interface 'wan6'
        option ifname 'eth0'
        option proto 'dhcpv6'

config switch
        option name 'switch0'
        option reset '1'
        option enable_vlan '1'
        option enable_learning '0'

config switch_vlan
        option device 'switch0'
        option vlan '1'
        option ports '0 2 3 4'

config switch_vlan
        option device 'switch0'
        option vlan '2'
        option ports '5 6'

config switch_vlan
        option device 'switch0'
        option vlan '3'
        option ports '1 0t'
```

- Configurar o Rádio (vim /etc/config/firewall)

Configurar o Rádio (vim /etc/config/firewall)

```
config defaults
        option syn_flood        1
        option input            ACCEPT
        option output           ACCEPT
        option forward          ACCEPT
```

- Configurar o OVS no rádio

Configurar o OVS no rádio

```
ovs-vsctl add-br r2

ovs-vsctl set bridge r2 other-config:hwaddr=00:00:00:00:00:02

ovs-vsctl set bridge r2 protocols=OpenFlow10,OpenFlow13

ovs-vsctl set-controller r2 tcp:10.10.10.200:6633

ovs-vsctl set controller r2 connection-mode=out-of-band

ovs-vsctl set bridge r2 fail_mode=secure

ovs-vsctl add-port r2 wlan0 -- set Interface wlan0 ofport_request=1

ovs-vsctl add-port r2 eth1.3 -- set Interface eth1.3 ofport_request=2
```

- Configurar o wifi no rádio, setar wlan0 para interface wifi

D

Código fonte do módulo de controle

Módulo de controle - Aplicação python SBC-Flow

```
#_*_coding:utf-8_*_
from django.template.response import TemplateResponse
from time import sleep
from funcoes import *
from django.http import JsonResponse, HttpResponse
import json
import datetime
from models import *
def index(request):
    titulo = 'Sistema de Balanceamento de Carga - SBCFlow'
    return TemplateResponse(request, 'index.html', locals())
def monitoramento(request):
    controlador = Controlador.objects.all()[0]
    #Pega do BD todas as celulas cadastradas
    celula = Celula.objects.filter(status='A')
    celulas = []
    radios = []
    clientes = []
    carga_radio()
    if controlador.data_atualizacao != controlador.data_monitoramento:
        controlador.data_monitoramento = controlador.data_atualizacao
        controlador.save()
        analise_carga()
        analise_usuario()
        #Para cada celula pega do BD cada switch cadastrado para aquela Celula
        for c in celula:
            switches = Switch.objects.filter(celula=c)
            soma = 0
            denominador = 0
            for s in switches:
                soma += s.tr
                denominador += s.tr ** 2
                radios.append(
                    {
                        'celulaid': c.id,
                        'switch': s.idcelula,
                        'tms': s.tr,
                        'carga': s.carga,
                    }
                )
                usuarios = Usuario.objects.filter(switch=s)
                for u in usuarios:
                    clientes.append(
                        {
                            'celulaid': c.id,
                            'radioid': s.idcelula,
                            'mac': u.macadress,
                            'tm': u.tu,
                            'sinal': u.sinal,
                            'migrar': u.migracao
                        }
                    )
            beta = (soma ** 2) / (len(switches) * denominador)
            celulas.append(
                {
                    'celulaid': c.id,
                    'tmc': c.tmc,
                    'assimetria': c.assimetria,
                    'beta': round(beta, 2)
                }
            )
```

Continuação...

```
        #Para cada celula pega do BD cada switch cadastrado para aquela Celula
        for c in celula:
            switches = Switch.objects.filter(celula=c)
            soma = 0
            denominador = 0
            for s in switches:
                soma += s.tr
                denominador += s.tr ** 2
                radios.append(
                    {
                        'celulaid': c.id,
                        'switch': s.idcelula,
                        'tms': s.tr,
                        'carga': s.carga,
                    }
                )
                usuarios = Usuario.objects.filter(switch=s)
                for u in usuarios:
                    clientes.append(
                        {
                            'celulaid': c.id,
                            'radioid': s.idcelula,
                            'mac': u.macadress,
                            'tm': u.tu,
                            'sinal': u.sinal,
                            'migrar': u.migracao
                        }
                    )
            beta = (soma ** 2) / (len(switches) * denominador)
            celulas.append(
                {
                    'celulaid': c.id,
                    'tmc': c.tmc,
                    'assimetria': c.assimetria,
                    'beta': round(beta, 2)
                }
            )

    return TemplateResponse(request, 'monitoramento.html', locals())

def carga_radio():
    #Pega do BD todas as celulas cadastradas
    celula = Celula.objects.filter(status='A')
    #Para cada celula pega do BD cada switch cadastrado para aquela Celula
    for c in celula:
        switches = Switch.objects.filter(celula=c)
        for s in switches:
            soma = 0
            usuarios = Usuario.objects.filter(switch=s)
            for u in usuarios:
                soma += u.tu
            s.tr = soma
            s.save()


def analise_carga():
    #Pega do BD todas as celulas cadastradas
    celula = Celula.objects.filter(status='A')
    #Para cada celula pega do BD cada switch cadastrado para aquela Celula
    for c in celula:
        switches = Switch.objects.filter(celula=c)
        soma = 0
        denominador = 0
        for s in switches:
            soma += s.tr
            denominador += s.tr ** 2
        try:
            tmc = soma / len(switches)
        except Exception:
            tmc = 0
        c.tmc = tmc
        try:
            beta = (soma ** 2) / (len(switches) * denominador)
        except Exception:
            beta = 0
        if round(beta, 2) < 0.98:
            c.assimetria = True
            for s in switches:
                if s.tr > tmc:
                    s.carga = 'O'
                if s.tr < tmc:
                    s.carga = 'U'
                if s.tr == tmc:
                    s.carga = 'B'
                s.save()
```

Continuação...

```
        else:
            c.assimetria = False
            for s in switches:
                s.carga = 'B'
                s.save()
        c.save()

def analise_usuario():
    #pesquisar somente nas células assimétricas
    celula = Celula.objects.filter(assimetria=True)
    #percorrer todas as celulas assimétricas
    for c in celula:
        #Seleciona apenas os switches que estão sobrecarregados de uma celula assimetrica
        switches = Switch.objects.filter(carga='O', celula=c)
        users = []
        t_mc = float(c.tmc)
        totaltmc = t_mc - (t_mc * 0.005)
        #percorre os switches sobrecarregados
        for s in switches:
            #Seleciona os usuários do switch sobrecarregado (do BD)
            usuarios = Usuario.objects.filter(switch=s).order_by('sinal')
            cm = 0
            cap = s.capacidade / 2
            #Valida condições mínimas para as próximas fases do algoritmo (Usuários mínimos por switch [3])
            if usuarios.count() >= 3 and s.tr > cap:
                for u in usuarios:
                    if u.tu > 0:
                        cm += u.tu
                        if s.tr - cm >= totaltmc:
                            u.migracao = True
                            u.save()
                        else:
                            cm = cm - u.tu

'''PROCEDIMENTO MIGRACAOO OTIMIZADO (SEM CONSULTA AOS RADIOS)'''
def migracao():
    celula = Celula.objects.filter(assimetria=True)

    for c in celula:
        usuarios = []
        switches = Switch.objects.filter(celula=c)
        for s in switches:
            usuario = Usuario.objects.filter(switch=s)
            for u in usuario:
                usuarios.append(
                    {
                        'radio': s.id,
                        'mac': u.macadress,
                        'migracao': u.migracao,
                    }
                )
        over = Switch.objects.filter(celula=c, carga='O')
        for o in over:
            comandoOver = ''
            for u in usuarios:
                if (u['radio'] != o.id) or (u['radio'] == o.id and u['migracao']):
                    if comandoOver:
                        comandoOver += ' echo %s >> /var/run/hostapd-wlan0.maclist;' % (u['mac'])
                    else:
                        comandoOver = 'echo %s > /var/run/hostapd-wlan0.maclist;' % (u['mac'])
            monta_blacklist(comandoOver, o.ipswitch)
        under = Switch.objects.filter(celula=c, carga='U')
        for u in under:
            comandoUnder = ''
            for us in usuarios:
                if (us['radio'] != u.id and not us['migracao']):
                    if comandoUnder:
                        comandoUnder += ' echo %s >> /var/run/hostapd-wlan0.maclist;' % (us['mac'])
                    else:
                        comandoUnder = 'echo %s > /var/run/hostapd-wlan0.maclist;' % (us['mac'])
            monta_blacklist(comandoUnder, u.ipswitch)
        balanced = Switch.objects.filter(celula=c, carga='B')
        for b in balanced:
            comandoBalanced = ''
            for u in usuarios:
                if (b.id != u['radio']):
                    if comandoBalanced:
                        comandoBalanced += ' echo %s >> /var/run/hostapd-wlan0.maclist;' % (u['mac'])
                    else:
                        comandoBalanced = 'echo %s > /var/run/hostapd-wlan0.maclist;' % (u['mac'])
            monta_blacklist(comandoBalanced, b.ipswitch)

        for s in switches:
            executa_blacklist(s.ipswitch)
```