Programa Interdisciplinar de Pós-Graduação em

Computação Aplicada

Mestrado Acadêmico

Felipe Rabuske Costa

Nuoxus - Um Modelo de Caching Proativo de Conteúdo Multimídia para Fog Radio Access Networks (F-RANs)

São Leopoldo, 2018

Felipe Rabuske Costa

NUOXUS - UM MODELO DE CACHING PROATIVO DE CONTEÚDO MULTIMÍDIA PARA FOG RADIO ACCESS NETWORKS (F-RANS)

Dissertação apresentada como requisito parcial para a obtenção do título de Mestre pelo Programa de Pós-Graduação em Computação Aplicada da Universidade do Vale do Rio dos Sinos — UNISINOS

Orientador:
Prof. Dr. Rodrigo da Rosa Righi

São Leopoldo
2018

C837n Costa, Felipe Rabuske
Nuoxus - um modelo de caching proativo de conteúdo multimídia para Fog Radio Access Networks (F-RANs) / por Felipe Rabuske Costa. – 2018.
86 f. : il. ; 30 cm.

Dissertação (Mestrado) — Universidade do Vale do Rio dos Sinos, Programa de Pós-Graduação em Computação Aplicada, São Leopoldo, RS, 2018.

"Orientador: Dr. Rodrigo da Rosa Righi".

1. Redes de acesso em FOG. 2. Redes de acesso de borda. 3. Fog Radio Access Networks (F-RAN). 4. Caching. 5. Nuoxus. 6. Similaridade de cosseno. I. Título.

CDU: 004.738.2

Catalogação na Publicação:
Bibliotecário Alessandro Dietrich - CRB 10/2338

Felipe Rabuske Costa

Nuoxus - Um Modelo de Caching Proativo de Conteúdo Multimídia para Fog Radio Access Networks (F-RANs)

Dissertação apresentada à Universidade do Vale do Rio dos Sinos – Unisinos, como requisito parcial para obtenção do título de Mestre em Computação Aplicada.

Aprovado em 28 de fevereiro de 2018.

BANCA EXAMINADORA

Prof. Dr. Rodrigo da Rosa Righi – UNISINOS

Prof. Dr. Cristiano André da Costa – UNISINOS

Prof. Dr. Cristiano Bonato Both – UFCSPA

Prof. Dr. Rodrigo da Rosa Righi (Orientador)

Visto e permitida a impressão
São Leopoldo, 2018

Prof. Dr. Rodrigo da Rosa Righi
Coordenador PPG em Computação Aplicada

"*Some people don't like change, but you need to embrace change if the alternative is disaster.*".
(Elon Musk)

RESUMO

Estima-se que até o ano de 2020, cerca de 50 bilhões de dispositivos móveis estarão conectados a redes sem fio e que 78% de todo o tráfego de dados gerado por esse tipo de dispositivos será conteúdo multimídia. Essas estimativas fomentam o desenvolvimento da quinta geração de redes móveis (5G). Uma das arquiteturas mais recentemente proposta, chamada de *Fog Radio Access Networks* (F-RAN), dá aos componentes localizados na borda da rede poder de processamento e armazenamento endereçados às atividades da rede. Um dos principais problemas dessa arquitetura é o intenso tráfego de dados no seu canal de comunicação centralizado chamado *fronthaul*, utilizado para conectar as antenas (F-APs) à rede externa. Dado esse contexto, esse trabalho apresenta o Nuoxus, um modelo de *caching* de conteúdo multimídia voltado para F-RANs que visa amenizar esse problema. Ao armazenar esse tipo de conteúdo nos nós de rede mais próximos ao usuário, o número de acessos concorrentes ao *fronthaul* é reduzido, sendo esse um dos fatores agravantes na latência de comunicação na rede. O Nuoxus pode ser executado em qualquer nó da rede que possua capacidade de armazenamento e processamento, ficando responsável por gerenciar o *caching* de conteúdo desse nó. Sua política de substituição de conteúdo utiliza a similaridade de requisições entre os nós filhos e o restante da rede como um fator para definir a relevância de armazenar o conteúdo requisitado em *cache*. Além disso, utilizando esse mesmo processo, o Nuoxus sugere, de forma proativa, aos demais nós filhos que apresentam um alto grau de similaridade que façam o *caching* desse conteúdo, visando um possível futuro acesso. A análise do estado da arte demonstra que até o momento não existe nenhum outro trabalho que explore o histórico de requisições para fazer *caching* de conteúdo em arquiteturas multicamadas para redes sem fio de forma proativa e sem utilizar algum componente centralizado para fazer coordenação e predição de *caching*. A fim de comprovar a eficiência do modelo, foi desenvolvido um protótipo utilizando o simulador ns-3. Os resultados obtidos demostram que a utilização do Nuoxus foi capaz de reduzir a latência de rede em cerca de 29.75%. Além disso, quando comparado com outras estratégias de *caching*, o número de acesso à *cache* dos componentes de rede aumentou em 53.16% em relação à estratégia que obteve o segundo melhor resultado.

Palavras-chave: Redes de Acesso em FOG. Redes de Acesso de Borda. F-RAN. Caching. Nuoxus. Similaridade de Cosseno.

ABSTRACT

It is estimated that by the year 2020, about 50 billion mobile devices will be connected to wireless networks and 78% of the data traffic of this kind of device will be multimedia content. These estimates foster the development of the 5th generation of mobile networks (5G). One of the most recently proposed architectures, named Fog Radio Access Networks or F-RAN, gives the components located at the edge of the network the processing power and storage capacity to address network activities. One of the main problems of this architecture is the intense data traffic in its centralized component named fronthaul, which is used to connect the antennas (F-APs) to the external network. Given this context, we propose Nuoxus, a multimedia content caching model for F-RANs that aims to mitigate this problem. By storing the content in the nodes closest to the user, the number of concurrent accesses to the fronthaul is reduced, which decreases the communication latency of the network. Nuoxus can run on any network node that has storage and processing capacity, becoming the responsible for managing the cache of that node. Its content replacement policy uses the similarity of requests between the child nodes and the rest of the network as a factor to decide the relevance of storing the requested content in the cache. Furthermore, by using this same process, Nuoxus proactively suggests to the child nodes whose degree of similarity is high to perform the caching of the content, assuming they will access the content at a future time. The State-of-the-art analysis shows that there is no other work that explores the history of requests to cache content in multi-layer architectures for wireless networks in a proactive manner, without using some centralized component to do coordination and prediction of caching. To demonstrate the efficiency of the model, a prototype was developed using the ns-3 simulator. The results obtained demonstrate that the use of Nuoxus reduced network latency in 29.75%. In addition, when compared to other caching strategies, the cache hit increased by 53.16% when compared to the strategy that obtained the second-best result.

Keywords: FOG Radio Access Networks. Edge Radio Access Networks. F-RAN. Caching. Nuoxus. Cosine Similarity.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

3GPP	*3rd Generation Partnership Project*
4G	Quarta Geração de Redes Móveis ou Quarta Geração de Internet Móvel
5G	Quinta Geração de Redes Móveis ou Quinta Geração de Internet Móvel
BBU	*Baseband Unit*
C-RAN	*Cloud Radio Access Network*
CPU	*Cloud Processing Unit*
CRP	*Chinese Restaurant Process*
CSMA	*Carrier Sense Multiple Access*
D2D	*Device to Device*
DCSN	*Distributed Cloud Service Networks*
EPC	*Evolved Packet Core*
F-AP	*Fog Access Point*
F-RAN	*Fog Radio Access Network*
F-UE	*Fog User Equipment*
H-CRAN	*Heterogeneous Cloud Radio Access Network*
HPN	*High Power Node*
HTTP	*Hypertext Transfer Protocol*
HetNet	*Heterogeneous Network*
LFU	*Least Frequently Used*
LRU	*Last Recently Used*
LTE	*Long Term Evolution*
PSO	*Particle Swarm Optimization*
RAN	*Radio Access Network*
RRH	*Remote Radio Head*
RR	*Random Replacement*
SCD	*State Based Content Distribution*
SLRU	*Segmented LRU*
SNR	*Signal-to-Noise Ratio*
URL	*Uniform Resource Locator*

SUMÁRIO

1 INTRODUÇÃO

O recente crescimento no número de dispositivos móveis conectados através de redes sem fio alterou a forma em que consumimos conteúdo. Tanto serviços considerados essenciais para vida contemporânea, como saúde a distância e bancos online (*Internet Banking*), quanto aqueles destinados ao entretenimento, como vídeos e jogos, são fornecidos através de dispositivos que dependem de uma conexão sem fio para sua utilização (CHEN, 2016).

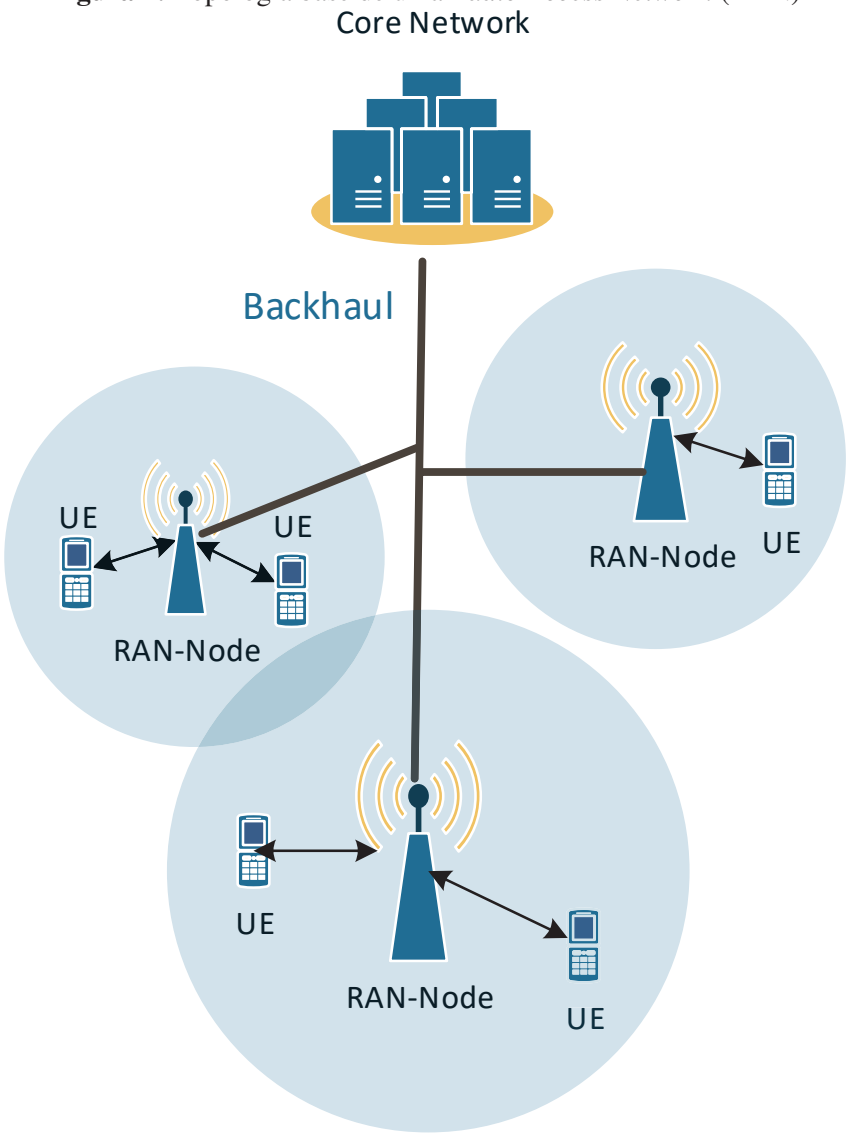
A simplificação ao acesso móvel à informação, entretenimento e colaboração social fez com que o número de *smartphones* crescesse significativamente. Um estudo conduzido por Poushter (2016) entre os anos de 2014 e 2015, conclui que 45% da população dos 40 países analisados possuí um *smartphone*, sendo que esses números são maiores em países desenvolvidos, como por exemplo, Coreia do Sul (88%) e Estados Unidos (72%). Além do crescimento dos dispositivos móveis de uso pessoal, a indústria também está cada vez mais dependente desse tipo de *hardware*, devido à *internet* das coisas, que pode ser vista como sendo o *framework* necessário para que produtos troquem informações entre si, possibilitando novas aplicações industriais. É esperado que a internet das coisas leve a humanidade à próxima revolução industrial (BHARDWAJ; KOLE, 2016).

A crescente adoção de objetos conectados tem como consequência um aumento da demanda de infraestrutura das redes sem fio. De acordo com Samsung Electronics (2015), estima-se que 50 bilhões de dispositivos estarão conectados à rede sem fio em 2020 e Cisco (2017) estima que o tráfego de dados gerados por dispositivos móveis chegará em 49 exabytes por mês em 2021. Esse aumento quase eminente na quantidade de dispositivos conectados requer avanços significativos na área de redes, especialmente nas redes sem fio. Visando adequar-se a essa realidade, esforços estão sendo colocados na definição da quinta geração de redes móveis (5G), a qual visa uma maior largura de banda e cobertura, e uma menor latência e consumo de energia (OSSEIRAN et al., 2014). A quinta geração de redes móveis não só possibilitará uma adoção em massa de dispositivos móveis, mas também será a real facilitadora de novas tecnologias, como carros autônomos, controle remoto de robôs e aplicações táteis de tempo real. Isso porque essas aplicações necessitam de uma latência na ordem dos milissegundos e uma alta confiabilidade (5G PPP Architecture Working Group, 2016).

Os requerimentos das redes 5G a nível de usuário estão sendo finalizados pela 3GPP (3GPP, 2016). Porém, isso não impediu com que as indústrias e comunidade científica começassem a trabalhar na arquitetura da próxima geração de redes sem fio. Essas novas arquiteturas focam no aperfeiçoamento das Redes de Acesso a Rádio (em inglês: *Radio Access Networks* ou RANs), que são responsáveis por implementar as tecnologias de rádio em um sistema de comunicação. A arquitetura base de uma RAN é formada por três tipos de componentes: equipamento de usuários (em inglês: user equipment ou UE); nós de rede de acesso a rádio (em inglês: *Radio Access Network Nodes* ou RAN-Nodes); e a rede de núcleo (em inglês: *core network* ou CN). Esses componentes ficam dispostos conforme a topologia apresentada na Figura 1. Os UEs se

conectam aos RAN-Nodes através de uma conexão sem fio com a finalidade de se comunicar com a CN. Dependendo da arquitetura RAN implementada, esses componentes são renomeados. Por exemplo, na arquitetura das redes LTEs (*Long Term Evolution*), os UEs são chamados de eUEs, os RAN-Nodes de eNB e a CN de EPC (XU et al., 2016).

Figura 1: Topologia base de uma *Radio Access Network* (RAN)



Fonte: Adaptado de (XU et al., 2016).

Uma das propostas mais recentes em termos de arquitetura para redes 5G é chamada de Rede de Acesso a Rádio em FOG (em inglês: *FOG Radio Access Networks* ou *F-RAN*). Nessa arquitetura, parte da capacidade de processamento e armazenamento é delegada aos dispositivos inteligentes localizados nas bordas da rede, ao contrário das demais arquiteturas 5G, onde essas funcionalidades estão completamente centralizadas (PENG et al., 2016). Essa descentralização dos componentes nas F-RANs não somente é necessária para mitigar os problemas de congestionamento inerentes a qualquer arquitetura centralizada, mas também permite a utilização da inteligência dos componentes de rede e dos seus diferentes modos de transmissão para otimizar

seu desempenho.

1.1 Motivação

Os resultados publicados por um recente estudo feito pela Cisco (2017) alegam que 60% dos dados consumidos por dispositivos móveis em 2016 foram formados por conteúdo multimídia. Esse mesmo estudo estima que esse tipo de conteúdo irá formar 78% dos dados consumidos em 2021 por esse tipo de dispositivos. Não é difícil imaginar a razão por trás desse aumento, dado a crescente popularidade de serviços de *streaming* de vídeo, como *Netflix*[1], *Youtube*[2] e até mesmo de emissoras de televisão que estão adotando essa nova plataforma (CISCO, 2017).

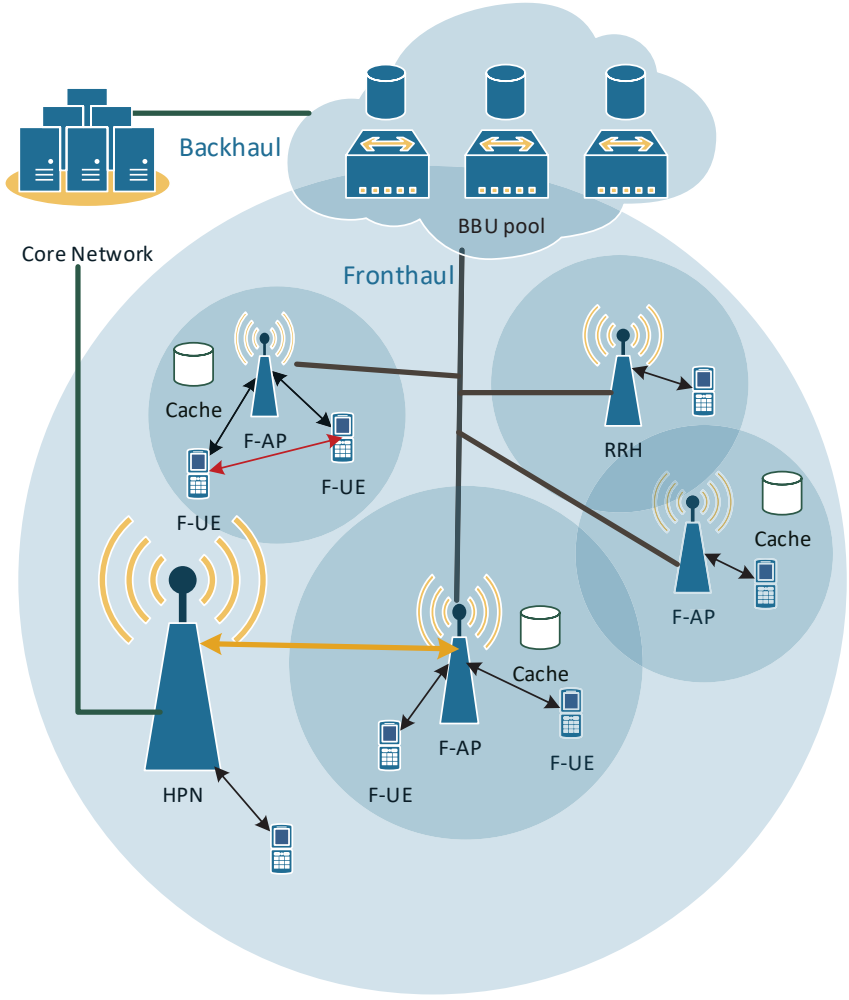
Como apontado por Brodersen, Scellato e Wattenhofer (2012), apesar de sua natureza global, o interesse em um dado conteúdo multimídia ocorre em sua maior parte de forma local, devido a diversos fatores. Por exemplo, esporte, política, e notícias tendem a ter um foco de interesse localizado e, portanto, sua popularidade acaba sendo impactada pela localização geográfica dos usuários. Além disso, como apresentado por Scellato et al. (2011), usuários de regiões próximas tendem a apresentar linguagem e cultura similares, facilitando a disseminação de conteúdo através de conexões em redes sociais entre aqueles localizados em uma distância relativamente pequena, como uma mesma cidade ou bairro.

Seguindo o modelo F-RAN, representado na Figura 2, é possível concluir que sua arquitetura vem ao encontro dos dados publicados por esses estudos (PENG et al., 2016). Em primeiro lugar, os componentes de borda dessa arquitetura, que são os Pontos de Acesso Baseados em Computação *Fog* (em inglês: *Fog-Computing-Based Access Points* ou F-APs) e os Equipamentos de Usuário Fog (em inglês: *Fog User Equipaments* ou F-UEs), possuem capacidade de armazenamento de dados, o que possibilita a criação de *cache* de conteúdo, diminuindo o tempo de acesso a esses dados em uma determinada região (WANG et al., 2014a). Em segundo lugar, a maior parte dos componentes dessa arquitetura são considerados inteligentes, ou seja, eles possuem conhecimento de sua localização dentro da rede e podem manter um histórico de contexto sobre o conteúdo requisitado pelos usuários em dado espaço de tempo. Esse tipo informação pode ser utilizado para melhorar a forma como os dados são transmitidos dentro da rede e também permite a proatividade na transmissão de conteúdos em horários onde a rede está menos carregada, para serem consumidos em horários onde possa haver mais demanda (BASTUG; BENNIS; DEBBAH, 2014).

As projeções para o futuro próximo em relação ao consumo de dados por dispositivos móveis deixam clara a necessidade de uma evolução das redes sem fio (OSSEIRAN et al., 2014). Diversos pesquisadores (PENG et al., 2016), (WANG et al., 2014a), (HUANG; ZHAO; ZHANG, 2016), (FENG et al., 2016), (YU; TSAI; PANG, 2016) não só apontam *caching* como um dos pilares fundamentais para atingir os objetivos estabelecidos para a nova geração de re-

[1] http://netflix.com
[2] http://youtube.com

Figura 2: Arquitetura de uma *Fog Radio Access Network* (F-RAN) como proposta por Peng et al. (2016). F-APs são os Pontos de Acesso Baseados em Computação *Fog* (em inglês: *Fog-Computing-Based Access Points* que podem ser vistos como antenas com capacidade de processamento e armazenamento; F-UEs são os Equipamentos de Usuário Fog (em inglês: *Fog User Equipaments*); BBU-Pool é a parte em nuvem de uma F-RAN, na qual as requisições para a rede de núcleo são processadas; HPNs são os Nós de Alta Potência (em inglês: *High Power Nodes*), presentes para garantir a mobilidade do usuário.



Fonte: Adaptado de (PENG et al., 2016).

des sem fio, mas também apontam a necessidade de uma estratégia de *caching* e de distribuição de conteúdo que explore toda a capacidade das novas arquiteturas, dando ênfase na proatividade e predição de conteúdo a ser armazenados em dispositivos de borda.

Conforme a análise feita nos trabalhos relacionados, apresentada no Capítulo 3, a comunidade cientifica está se dedicando na aplicação de diversas estratégias de *caching* de conteúdo multimídias em novas arquiteturas de rede sem fio. Porém, dos trabalhos relacionados, nenhum realiza o *caching* de conteúdo de forma proativa, sem que haja a necessidade de um sistema externo, centralizado e dedicado a fazer a predição de conteúdo. A diferença do modelo proposto, é que ele atua de maneira contínua conforme as requisições são realizadas, e toda a informação necessária para realizar a predição se encontra no nó executando o modelo.

1.2 Questão de Pesquisa

Frente aos avanços feitos até então para definição de uma arquitetura para redes 5G baseada em *Fog Computing*, surgem novas possibilidades de explorar os componentes dessa arquitetura para otimizar o desempenho da rede, especialmente vista a alta demanda por conteúdos multimídia (CISCO, 2017). Dado essas informações, surge a seguinte questão de pesquisa:

> "Como seria um modelo de *caching* proativo de conteúdo multimídia para redes 5G baseadas na arquitetura F-RAN, que explora o histórico de acesso dos usuários e que seja capaz de fornecer dados de *caching* para auxiliar a seleção do modo de transmissão de dados?"

A seguir apresenta-se os conceitos acima utilizados expandidos de acordo com o contexto da questão de pesquisa, visando sua melhor compreensão:

- ***Caching***: consiste no armazenamento de dados nos nós de rede intermediários, ou seja, naqueles localizados entre o usuário e a rede de núcleo;
- **Proatividade**: Em um modelo de *caching* tradicional, o sistema realiza uma análise das informações relacionadas ao conteúdo requisitado para decidir, através de uma política de substituição, se tal conteúdo dever ser armazenado em *cache* (FIORE et al., 2009). *Caching* proativo, como levantado pela questão de pesquisa, significa que o modelo deve ter a capacidade de armazenar conteúdo em sua *cache* antes mesmo dele ser requisitado, de forma que esse conteúdo já esteja disponível na sua primeira requisição;
- **Conteúdo Multimídia**: são conteúdos normalmente estáticos, como por exemplo áudio, vídeo ou imagens, e cujo tamanho é considerado maior que outras formas de informação, principalmente se tratando de vídeo;
- **Redes 5G Baseadas em F-RAN**: a questão de pesquisa tem como foco a arquitetura F-RAN por dois motivos. O primeiro deles é que essa é a mais recente abordagem para a nova geração de redes móveis. O segundo é que, por se tratar de *Fog Computing*, os nós dessa arquitetura permitem a execução de um modelo e o armazenamento de dados em *cache*. A proposta segue a arquitetura apresentada por Peng et al. (2016), que é explorada na Seção 2.1.
- **Componentes de Borda**: são os componentes localizados após o núcleo da rede. Na terminologia das F-RANs, esses seriam os F-APs, os F-UEs e os possíveis componentes fog intermediários, como distribuidores de conteúdo e roteadores;
- **Modos de Transmissão**: se tratam dos modos de transmissão propostos pela arquitetura F-RAN, que são: D2D, Relé, Coordenação Local Distribuída e HPN. Esses modos de transmissão são detalhados na Seção 2.1.4. O modelo proposto deve ser capaz de fornecer dados de *caching* para auxiliar nessa tomada de decisão.

1.3 Objetivos

Esse trabalho tem como objetivo geral a criação de um modelo de *caching* proativo de conteúdo multimídia para F-RANs, fazendo uso dos recursos disponíveis nessa arquitetura, como conhecimento do histórico de requisições. Com a finalidade de atingir esse resultado, os seguintes objetivos específicos são estabelecidos:

- Pesquisar o estado da arte relacionado a técnicas de *caching* aplicadas às redes 5G, com um foco maior na arquitetura das F-RANs;
- Desenvolver um modelo multicamadas de *caching* de conteúdo multimídia para F-RANs que permita o armazenamento de conteúdo de forma proativa.
- Estabelecer uma política de substituição que considere dados do histórico de acesso para estabelecer a similaridade entre os nós da rede;
- Avaliar o desempenho do modelo desenvolvido através de simulações de um protótipo;
- Analisar os resultados obtidos e compará-los com os resultados de outras técnicas de *caching* existentes;

Através desses objetivos individuais, espera-se obter um modelo que resulte na amenização do maior problema das F-RANs: o acesso excessivo ao *fronthaul*, o qual é um componente centralizado. Como consequência, espera-se que o modelo proposto contribua para a redução da latência de rede, já que dois dos fatores que corroboram com a latência de rede, a distância trafegada e a taxa de ocupação do canal de comunicação (*fronthaul*) (CHOI et al., 2004), serão reduzidos.

1.4 Estrutura do Texto

O restante do texto está organizado da seguinte forma: no Capítulo 2, são apresentados os conceitos fundamentais que serão utilizados durante o decorrer do trabalho. Entre eles estão o conhecimento sobre diferentes estratégias de *caching* utilizadas em redes e a evolução das arquiteturas de rede 5G, incluindo a arquitetura F-RAN, a qual é utilizada como arquitetura base desse trabalho.

No Capítulo 3, é apresentada a metodologia de seleção da literatura científica que contribui para a área de pesquisa desse trabalho, seguido de uma análise individual de cada um dos trabalhos selecionados. Por fim, esse capítulo também apresenta, com base em tais trabalhos, as lacunas na área de pesquisa e como o modelo proposto pretende preenchê-la. Já no Capítulo 4 é apresentado o modelo desenvolvido, com as decisões de projeto e sua arquitetura. Também são dados detalhes sobre o funcionamento da política de substituição desenvolvida para o Nuoxus

e de como o algoritmo de seleção do modo de transmissão proposto por Peng et al. (2016) pode ser ajustado para considerar os dados fornecidos pelo modelo.

O Capítulo 5 apresenta o protótipo desenvolvido para avaliação do modelo, seguido dos cenários de simulação criados para esse objetivo. Os resultados obtidos através das simulações são apresentados e discutidos no Capítulo 6. Por fim, no Capítulo 7 é apresentada a conclusão do trabalho, onde as contribuições científicas do modelo são destacadas juntamente com possibilidades para trabalhos futuros envolvendo o modelo.

2 FUNDAMENTAÇÃO TEÓRICA

Esse capítulo apresenta uma breve descrição dos conceitos e tecnologias relevantes para esse trabalho. Na primeira seção, são apresentados a arquitetura base para F-RANs, o funcionamento de seus componentes e como a comunidade cientifica chegou à essa arquitetura através de iterações em outros modelos propostos. Também são apresentadas estratégias de *caching* utilizadas em redes e uma classificação das políticas de substituição existentes. Em seguida, é apresentada a teoria do cálculo de similaridade de cosseno, utilizada pelo modelo proposto. Por fim, há uma introdução básica aos conceitos por trás do simulador ns-3, que será utilizado como ferramenta de avaliação do modelo.

2.1 FOG Radio Access Networks

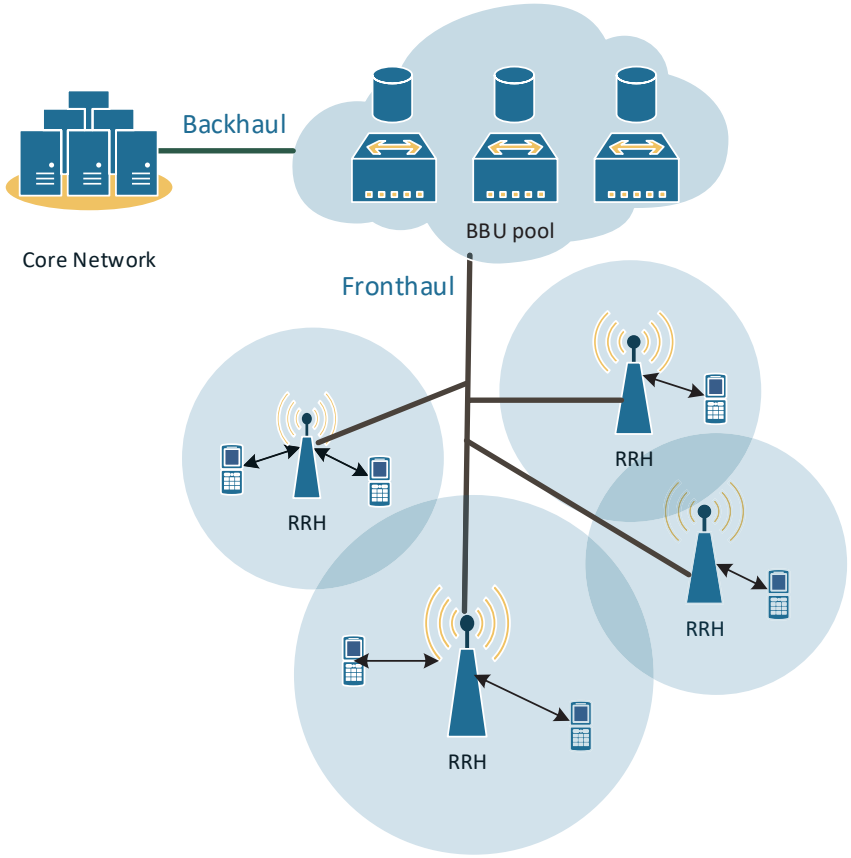
A arquitetura F-RAN referenciada nessa dissertação é apresentada por Peng et al. (2016) como uma evolução das Redes de Acesso Heterogêneas em Nuvem (em inglês *Heterogeneous Cloud Radio Access Networks* ou *H-CRAN*), que por sua vez é uma evolução das Redes de Acesso em Nuvem (em inglês *Cloud Access Networks* ou *C-RAN*). Portanto, para melhor descrever a arquitetura F-RAN, é essencial entender essas arquiteturas predecessoras e a necessidade de sua evolução visando atender os requerimentos das redes 5G.

2.1.1 *Cloud Access Networks*

A primeira arquitetura C-RAN foi apresentada pela **China Mobile** em 2010 como uma medida para diminuir os gastos de operação das redes de acesso (China Mobile, 2010). Essa arquitetura pode ser vista como uma solução centralizada composta de três componentes principais: 1) as unidades distribuídas de rádio, chamadas de *Remote Radio Heads* ou RRH, que contém também as antenas para realização de troca de sinal entre os dispositivos de usuário e a rede; 2) o agrupamento de unidades de base (*baseband unit pool* ou *BBU pool*), que realizam o processamento do sinal e a comunicação com a rede externa; 3) o canal de comunicação de alta largura de banda e baixa latência que conecta as RRHs e o BBU *pool*, conhecido como *fronthaul*. Normalmente o *fronthaul* consiste em uma rede de fibra ótica de alta taxa de transmissão.

O funcionamento dessa arquitetura se dá da seguinte maneira: primeiramente, o dispositivo de usuário se conecta em uma das antenas de uma das RRHs da rede. Ao enviar ou receber dados, a antena encaminha o sinal para a RRH, que faz o pré-processamento necessário para a sua digitalização e compressão. Em seguida, a RRH envia o sinal digitalizado e comprimido para o BBU *pool*, através do *fronthaul*. No BBU *pool*, o sinal é de fato processado e a comunicação com a rede externa é realizada. É importante ressaltar que o BBU *pool* consiste em um *cluster* de máquinas que recebem sinais de diversas RRHs, como demonstrado na Figura 3.

Figura 3: Arquitetura de uma *Cloud Radio Access Network* (C-RAN)



Fonte: Adaptado de (China Mobile, 2010).

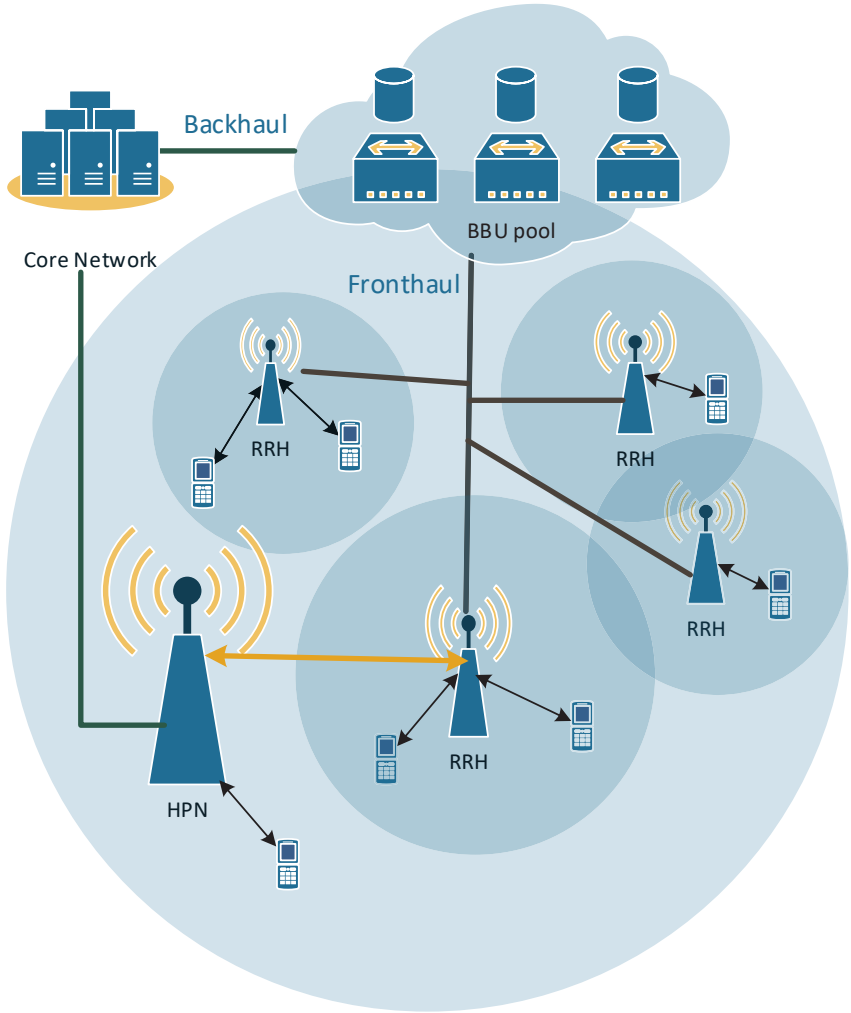
A principal vantagem dessa arquitetura é que, por ser baseada em sistemas em nuvem, ela tem seu custo de implantação e manutenção reduzido. As RRHs contêm menos componentes que possam apresentar problemas, evitando altos gastos com manutenção quando as unidades de base se encontram em áreas remotas. Além disso, as RRHs podem cobrir áreas menores, diminuindo o consumo de energia desses componentes. Sua grande desvantagem é que, dado um alto número de RRHs e de dispositivos de usuário, o *fronthaul* pode ficar congestionado, aumentando a latência de comunicação. O mesmo ocorre caso o BBU *pool* seja mal dimensionado, o que pode fazer com que ele fique sobrecarregado em horários de uso intenso (China Mobile, 2010).

2.1.2 *Heterogeneous Cloud Radio Access Networks*

A fim de diminuir o gargalo causado pelo *fronthaul*, as H-CRANs são propostas por Peng et al. (2014) como uma fusão entre as redes de acesso heterogêneas (HetNet), que são utilizadas como arquitetura de redes de quarta geração (4G), com as redes C-RANs. Nelas, nós de alta potência (*High Power Nodes* ou HPNs) são implementados como parte da rede para oferecer uma maior cobertura de sinal e realizar o controle das RRHs. Nesse tipo de arquitetura, assim como o BBUs *pool*, as HPNs têm acesso direto à rede externa. Porém, elas ainda possuem uma

conexão com o BBU *pool* para fins de controle e coordenação. O canal de comunicação entre as HPNs e o BBU *pool* é chamado de *backhaul*.

Figura 4: Arquitetura de uma *Heterogeneous Cloud Radio Access Network* (H-CRAN)



Fonte: Adaptado de (PENG et al., 2014).

A Figura 4 retrata uma arquitetura H-CRAN. Nota-se que as HPNs podem receber conexões tanto de um dispositivo de usuário, quanto de RRHs. Além disso, as RRHs continuam possuindo conexão com o BBU *pool*. Essa grande diversidade de tipos de roteamento faz com que essa arquitetura exija um nível maior de coordenação entre os componentes da rede (PENG et al., 2014).

2.1.3 Computação em *Fog* e F-RANs

Infelizmente, as H-CRANs como concebidas não são ideias para o futuro das redes sem fio. Primeiramente, pois com o crescimento das redes sociais e do consumo de conteúdo de mídia, boa parte dos dados trafegados entre as RRHs e o BBU *pool* acabam consistindo de conteúdo estático e redundante, como por exemplo vídeo e imagens, o que acaba afetando a capacidade do

fronthaul, mesmo com o auxílio das HPNs. Depois, as H-CRANs não aproveitam a capacidade de processamento presente nos componentes de borda da rede, como RRHs e dispositivos de usuário inteligentes (PENG et al., 2016).

Computação em *Fog*, também conhecida como computação de borda em nuvem (em inglês *edge cloud computing*), é uma extensão da *cloud computing* cujos serviços de rede, computação e armazenamento são tipicamente, mas não exclusivamente, localizados na borda da rede. Uma característica da computação em *Fog* é que seus componentes conhecem sua localização na rede, e portanto, podem tomar decisões baseados nesse fator. Ao remover a centralização dos serviços acima mencionados, a computação em nuvem acaba diminuindo a latência entre as operações e tem uma maior escalabilidade em relação ao número de componentes de borda suportados pelo sistema (BONOMI et al., 2012).

A arquitetura F-RAN é apresentada por Peng et al. (2016) como uma evolução das H-RANs, adicionando a elas conceitos de computação em *Fog*, como demonstrado na Figura 2. Nelas os dispositivos de usuário que possuem capacidade de processamento e armazenamento (considerados dispositivos inteligentes) são renomeados para equipamentos de usuário fog (*Fog User Equipment* ou F-UE, e as RRHs passa a ser chamadas de pontos de acesso baseados em computação *Fog* (*fog-computing-based access points* ou F-APs). Além de fazer a maior parte do processamento de sinal, uma das grandes vantagens desse tipo de arquitetura é a capacidade de fazer cache de conteúdos populares, evitando que esses dados tenham que percorrer o *fronthaul* múltiplas vezes. Com isso, o impacto resultante do aumento do consumo desse tipo de conteúdo é mitigado pela arquitetura. Por fim, a capacidade de possuir componentes inteligentes como parte da rede possibilita que ela se adapte a diferentes condições de uso e trabalhe em modos diferentes, dependendo da localização dos dados e do destino da comunicação (PENG et al., 2016).

2.1.4 Modos de transmissão de uma F-RAN

Como mencionado na Subseção anterior, os dispositivos inteligentes localizados na borda da rede possibilitam que ela se reconfigure e tome diferentes modos de operação dependendo das condições que a comunicação está ocorrendo. Aqui, apresentamos esses diferentes modos, e como eles influenciam a operação da rede (PENG et al., 2016).

- **Modo D2D:** No modo D2D (*Device to Device*), dois dispositivos inteligentes se comunicam sem a interferência de nenhum outro componente da rede. Esse modo garante a maior eficiência possível, dado que a distância entre eles não seja maior que o limite necessário para transmissão do sinal.

- **Modo Relé:** Esse modo é muito similar ao D2D, e é ativado quando a distância entre os dispositivos querendo se comunicar é maior que o limite de transmissão de ambos. Nesse modo, outros F-UEs agem como uma ponte entre os dispositivos, de modo que a

comunicação ainda é centralizada e processada em ambos, porém com auxílio de demais componentes de borda.

- **Modo de Coordenação Local Distribuída:** Nesse modo, múltiplos F-APs se coordenam a fim de enviar e receber dados de um F-UE sem a necessidade de comunicação com o BBU *pool*. Esse modo apresenta seus ganhos através do uso de técnicas de comunicação similares à tecnologia de coordenação multi-pontos (*Coordinated Multipoint*) usada nas redes 4G.

- **Modo HPN:** Nesse modo, as HPNs são responsáveis pela comunicação. Ele é usado quando o sinal de transmissão está fraco, ou o usuário está em movimento para garantir uma melhor experiência e não penalizar a rede com tarefas de coordenação entre múltiplos F-APs.

- **Modo C-RAN Global:** Nesse modo, a rede age como uma C-RAN tradicional. O dispositivo de usuário se conecta a um F-AP e o sinal é transmitido para o BBU *pool*, onde é processado por completo. Esse modo é utilizado quando há necessidade de comunicação com a rede externa.

Esses são os modos de rede apresentados por Peng et al. (2016). Porém, é importante ressaltar que F-RANs estão ainda em desenvolvimento, e outros autores podem ver esses modos de forma diferente ou apresentar outros requisitos para alternar entre esses diferentes modos de transmissão.

2.2 Estratégias de Caching Utilizadas em Arquiteturas de Rede

Nessa seção são apresentados os conceitos fundamentais em relação à *caching*. Primeiramente é apresentada a classificação das estratégias de substituição de conteúdo em *cache* utilizando a técnica mais comuns de cada grupo como exemplo. Em seguida, é abordado como é possível realizar o *caching* do conteúdo disponível na *World Wide Web* de uma maneira genérica. É importante ressaltar que os tópicos aqui discutidos têm o enfoque em *cache* para conteúdo em redes. A realização de *cache* em arquitetura de computadores, sistemas operacionais ou banco de dados podem apresentar técnicas similares, mas muitas vezes uma técnica aplicável à uma área pode não ser a melhor em outra, dado que o custo de comunicação em redes é o fator que mais influencia nas estratégias escolhidas.

2.2.1 Substituição de Cache

Em qualquer sistema real, o espaço reservado para o armazenamento de dados é limitado e não é capaz de armazenar todos os objetos requisitados. Portanto, uma das questões mais importantes ao implementar um sistema de cache é definir sua política ou metodologia de substituição

de conteúdo. Em termos gerais, todas as políticas de substituição podem ser classificadas em cinco grupos, de acordo com as características levadas em consideração em seu design. A seguir, são apresentados esses cinco grupos, com as principais políticas de cada (ELAARAG, 2013).

2.2.1.0.1 Baseado em Recência

As políticas desse grupo são derivadas de uma propriedade conhecida como "localidade temporal", que diz que um dado acessado recentemente tem mais chances de ser acessado novamente em um curto espaço de tempo (DENNING, 2005). A principal estratégia desse grupo é o *Least Recently Used* (LRU). Essa estratégia mantém uma lista com os objetos em cache e qual foi a última vez que eles foram acessados. Quando não há mais espaço em cache, o objeto acessado por último é descartado para dar lugar ao objeto da nova requisição (TANENBAUM, 1992).

2.2.1.0.2 Baseado em Frequência

Nesse grupo se encontram as estratégias que utilizam a popularidade de um arquivo para mantê-lo em cache, ou seja, arquivos mais acessados são mantidos em cache, enquanto que os menos acessados são descartados (ELAARAG, 2013). O principal representante desse grupo é o *Least Frequently Used* (LFU). Nessa estratégia, uma lista de objetos é mantida, contendo o número de acesso a esses objetos. Quando não há mais espaço para armazenamento, e uma nova requisição é feita, a estratégia substitui o arquivo com o menor número de acessos. Existem algumas variações do LFU que também consideram a idade do arquivo, para evitar com que um arquivo que foi muito acessado no passado, mas não é mais relevante, não fique muito tempo em cache (JAYAREKHA; NAIR, 2010).

2.2.1.0.3 Baseado em Recência e Frequência

Esse grupo contém os algoritmos que tentam combinar tanto a propriedade da localidade temporal, quando a popularidade do arquivo para determinar qual objeto deve ser substituído em cache. Uma das estratégias mais referenciadas desse grupo é a *Segmented LRU* (SLRU). Ela contém duas listas de objetos, uma chamada de "probatória" e outra de "protegida". Os objetos em ambas as listas são ordenados por ordem de acesso, onde os objetos mais recentemente acessados ficam localizados no início da lista. Quando um objeto é requisitado e ele não se encontra em cache, ele é armazenado em cache e adicionado à lista probatória. Quando um objeto acessado e já se encontra em cache, ele é removido do seu lugar atual e adicionado no início da lista protegida. Isso faz com que todos os objetos dessa lista já tenham sido acessados ao menos duas vezes, evitando o problema do LRU, onde objetos requisitados frequentemente

podem ser removidos somente pois não foram acessados recentemente. Quando a lista protegida está cheia, e um objeto tem que ser movido para ela, a estratégia remove o objeto acessado por último da lista protegida, e adiciona como sendo o objeto acessado por último na lista probatória, dando mais uma chance para esse objeto (KAREDLA; LOVE; WHERRY, 1994).

2.2.1.0.4 Baseado em Funções

As estratégias desse grupo utilizam uma função característica para definir o valor de requisição de um objeto e determinar se ele deve ser ou não armazenado em cache. Esse grupo contém a maior diversidade de estratégias, então não é possível indicar um representante claro. Para efeitos de exemplificação, pode-se citar a estratégia proposta por Yang, Zhang e Zhang (2001), onde, dado uma série de tempos $T(0), T(1), T(2)...T(n)$, onde $T(n)$ é o tempo do enésimo acesso a dado arquivo, é utilizado uma série de Taylor para predizer quando ocorrerá o próximo acesso ($T(n+1)$) ao objeto e, portanto, estabelecer se é relevante armazená-lo em *cache*.

2.2.1.0.5 Baseado em Aleatoriedade

As estratégias baseadas em aleatoriedade são mais simples de se implementar e não requerem guardar nem atualizar informações adicionais sobre os arquivos armazenados em *cache*. O algoritmo dessa categoria que é mais utilizado é chamado de *Random Replacement* (RR). Nesse algoritmo, caso um arquivo solicitado não esteja na *cache*, ele substitui um arquivo escolhido de forma aleatória (LIU; LAU; HAN, 2015).

2.2.2 Cache de Conteúdo Web

Nessa Subseção são apresentadas formas de identificar e realizar *caching* de conteúdo *Web*, dado sua grande diversidade de propriedades. Os pesquisadores Wang et al. (2014a) mencionam em seu trabalho as duas formas mais comuns de se armazenar em *cache* o conteúdo da *World Wide Web*, que são apresentados em seguida:

2.2.2.0.1 Através da URL

Como 82% do tráfego de dados é feito utilizando o protocolo HTTP (FIORE et al., 2009), é benéfico a utilização de *cache* para esse protocolo. Cada conteúdo *Web* possuí uma URL, que é um endereço único para aquele conteúdo na maioria das vezes. Então, o sistema de *caching* pode aproveitar esse identificador único para manter uma tabela das URLs de todos os objetos armazenados em *cache* e, caso um segundo usuário requisite um objeto cuja URL esteja armazenada, o objeto é retornado diretamente. Enquanto esse é o método mais eficaz em termos de tempo de resposta e utilização de banda, ele possui três limitações significantes:

- ***Aliasing of Content*:** Muitas vezes, conteúdos idênticos possuem diferentes URLs
- ***Uncacheability*:** Quando existem objetos que são temporários ou de acesso único
- ***Content Changes*:** Isso ocorre quando os objetos requisitados são atualizados, portanto, a mesma URL aponta para objetos diferentes dependendo de quando o objeto foi acessado.

2.2.2.0.2 Através do prefixo

Considerado uma evolução do método anterior, ele é capaz de identificar conteúdos duplicados não somente através da URL ou de parte da URL, mas também através de uma função *hash* aplicada nos N primeiros *bytes* do objeto. Nesse sentido, a escolha de N é crucial para o sucesso desse método. Enquanto que um N muito pequeno aumenta as chances de conteúdo duplicado, um N muito grande aumenta a comunicação e o tempo necessário para computar a *hash*.

2.3 Similaridade de Cosseno

Na área de modelos vetoriais em sistemas de recuperação da informação, a similaridade de cosseno é uma técnica bastante utilizada para determinar o quão similar são dois objetos representados como vetores. Considerando os vetores a e b, a similaridade de cosseno $cosine(a, b)$ é calculada da seguinte forma:

$$cosine(a, b) = \frac{a \bullet b}{\|a\| \times \|b\|} \tag{2.1}$$

Onde $a \bullet b$ é o produto cartesiano entre os vetores a e b, enquanto que $\|a\| \times \|b\|$ representa a multiplicação entre as magnitudes desses vetores.

Olhando os termos individualmente, temos que o produto cartesiano entre os vetores a e b de tamanho N é dado por

$$a \bullet b = \sum_{i=1}^{N} a_i b_i \tag{2.2}$$

E a magnitude do vetor x é dada por

$$\|b\| = \sqrt{x \bullet x} \tag{2.3}$$

Portanto, unindo a equação 2.1 com as equações 2.2 e 2.3, temos que a similaridade de cosseno entre a e b (ambos de tamanho N) é dada por

$$cosine(a, b) = \frac{\sum_{i=1}^{N} a_i b_i}{\sum_{i=1}^{N} a_i^2 \times \sum_{i=1}^{N} b_i^2} \tag{2.4}$$

Como efetivamente esse é o cálculo do cosseno formado entre os vetores, em um plano

euclidiano de valores positivos, o cosseno calculado sempre estará entre 0 e 1, ou seja, $0 \leq cosine(a, b) \leq 1$ (SIDOROV et al., 2014), onde 0 é gerado quando os vetores não possuem nenhuma similaridade e 1 é gerado quando os vetores são iguais.

Além disso, ao contrário de outras técnicas de cálculo de similaridade, como por exemplo, o coeficiente de Jaccard, a similaridade de cosseno não considera os membros dos vetores cujo valor é zero, sendo amplamente utilizado para comparar vetores binários onde o interesse está em valores 1 na mesma posição de ambos os vetores (HAMERS et al., 1989).

2.4 Simulador ns-3

O ns-3 é um simulador de eventos discretos focado em simulações de rede e Internet. Seu público alvo é pesquisadores e instituições de ensino para uso educacional. Ele é de código aberto e está licenciado sobre a *GNU GPLv2*[1]. Por esse motivo, é ideal para desenvolvimento de simulações de componentes novos, visto a facilidade de estender os componentes existentes, ou até mesmo criar novos componentes e integrá-los aos modelos existentes (RILEY; HENDERSON, 2010).

Em resumo, o ns-3 possui modelos de componentes de rede que simulam tanto seu funcionamento quanto seu desempenho a nível de pacote de dados. Ele também provém um motor de simulação que utiliza esses componentes (NS-3 Project, 2017). As principais abstrações do ns-3 são as seguintes:

- ***Node*** (nó): é um dispositivo básico que possui capacidade computacional. Um nó por si só não é capaz de fazer muita coisa. Cabe à simulação adicionar funcionalidades ao nó, como periféricos, aplicações, protocolos de rede, etc.

- ***Application*** (aplicação): o ns-3 não possui conceito de sistema operacional nem de chamadas de sistema. Toda a simulação é construída com base em aplicações, que devem ser construídas para simular atividades do mundo real. Um exemplo de aplicação é um cliente e servidor UDP que trocam pacotes em intervalos de tempo pré-determinados.

- ***Channel*** (canal): é o componente responsável por conectar dois nós utilizando uma certa tecnologia de rede. Por exemplo, algumas das implementações padrões simulam conexões ponto-a-ponto, Wi-Fi, CSMA, etc. Além disso, vários parâmetros podem ser configurados, como por exemplo, largura de banda, taxa média de perda de pacotes, etc.

- ***Net Devices*** (dispositivos de rede): no ns-3, um dispositivo de rede simula o tanto o *hardware* quanto o *driver* de uma interface de rede, e deve ser instalado no nó para habilitar sua comunicação com os demais nós. Assim como no mundo real, onde um computador pode ter múltiplas interfaces de rede, um nó pode ter diversos *Net Devices* instalados.

[1] http://www.gnu.org/copyleft/gpl

- ***Helpers*** (ajudantes): instalar dispositivos de rede nos nós, instalar canais nos dispositivos de rede, atribuir endereços IPs, etc., acabam sendo tarefas muito comuns em simulações. Para facilitar o uso do ns-3, muitas das tecnologias de rede possuem objetos que são um *facade* para essas operações corriqueiras.

Para realizar uma simulação, um *script* deve ser desenvolvido onde nós representando os componentes da rede são criados e populados com implementações das demais abstrações. Existem vários outros simuladores que utilizam o ns-3 como base, dado sua natureza de código-aberto. Entre eles, está o projeto LENA[2], que desenvolve o simulador de rede voltado para LTE e *Evolved Packet Core* (EPC). Devido ao potencial e a alta demanda desse tipo de rede, o código estável do projeto LENA é periodicamente combinado com o ns-3, e entregue como seu módulo LTE. Infelizmente, devido ao caráter de novidade da arquitetura F-RAN, não existe nenhum projeto aberto do ns-3 para esse tipo de rede ’(PIRO; BALDO; MIOZZO, 2011).

[2]http://networks.cttc.es/mobile-networks/software-tools/lena/

3 TRABALHOS RELACIONADOS

Esse capítulo apresenta os trabalhos relacionados com a área dessa pesquisa e suas contribuições para a mesma. Na primeira seção, são apresentados os critérios de seleção dos trabalhos e detalhes sobre a execução da pesquisa. Nas seções subsequentes, são apresentados um resumo de cada trabalho relevante e os resultados obtidos pelos mesmos. Por fim, é feita uma análise comparativa entre os trabalhos e são apresentadas as lacunas na área de pesquisa.

3.1 Metodologia Utilizada na Seleção dos Trabalhos

Com objetivo de obter uma lista de publicações que apresentam o verdadeiro estado da arte relacionado a *caching* para F-RANs, a pesquisa foi restringida a base de dados bem-conceituadas pela comunidade científica. Através de seus motores de busca, as seguintes bibliotecas digitais foram pesquisadas: *IEEE Xplore Digital Library*[1], *ScienceDirect*[2], *Springer Link*[3] e *ACM Digital Library*[4].

A pesquisa foi restrita a termos de busca simples que, apesar de retornar um grande número de publicações, resultam em uma gama mais abrangente de trabalhos que se enquadram na área de pesquisa, dado que o termo *Fog* surgiu há pouco tempo. Além disso, foi utilizado os motores de busca avançados disponibilizados por essas bibliotecas digitais. Com isso, foi possível utilizar operadores booleanos nos termos de busca para obter melhores resultados. Os seguintes termos de busca foram utilizados:

1. *(Cache OR Caching) AND ("Radio Access Network" OR "F-RAN" OR "Fog" OR "Edge Radio Access Network")*
2. *"edge caching" AND Networks*
3. *(Cache OR Caching) AND 5G*

Além de restringir essa pesquisa inicial aos termos de busca acima mencionados, a pesquisa inicial também restringiu o idioma de publicação para inglês. As primeira busca ocorreu em março de 2017 e uma segunda em dezembro de 2017, e os resultados obtidos em cada execução foram extraídos no formado *BibTex* e importados na ferramenta StArt[5], que identifica automaticamente as publicações duplicadas e facilita o processo de revisão desses trabalhos.

Nessa pesquisa, os mecanismos retornaram 727 publicações únicas. Pelo fato do termo *Fog* ser muito genérico, muitos artigos de áreas não relacionadas com a pesquisa foram retornados. Portanto, o próximo passo foi realizar uma triagem baseada no título de cada trabalho, visando

[1] http://www.ieeexplore.ieee.org
[2] http://www.sciencedirect.com
[3] http://www.link.springer.com
[4] http://www.dl.acm.org
[5] http://lapes.dc.ufscar.br

eliminar aqueles trabalhos cuja área não era redes de computadores ou modelos de *caching*. Após essa análise, restaram 91 trabalhos.

Depois dessa primeira análise, foi realizada a leitura do resumo das 91 publicações com o objetivo de responder a seguinte pergunta: *"este trabalho apresenta um novo modelo de caching para uma arquitetura de rede 5G?"*. Após excluir as publicações que não apresentavam uma resposta positiva, restaram 17 trabalhos para análise textual das quais 3 foram eliminadas por não apresentarem um modelo de *caching* com resultados conclusivos. Com isso, 14 trabalhos foram considerados fortemente relacionados à essa pesquisa, e seus resumos são apresentados nas próximas subseções.

3.2 Trabalhos Analisados

Nessa seção, são apresentados os trabalhos levantados utilizando a metodologia descrita anteriormente. Cada trabalho é apresentado individualmente em uma subseção. Tenta-se destacar a abordagem utilizada por cada autor, os resultados obtidos e sua contribuição para área de *caching* em redes de acesso. É importante ressaltar que com a finalidade de manter a consistência textual, os termos utilizados pelos autores foram substituídos por seus equivalentes em uma F-RAN. Por exemplo, a expressão "estações de base com capacidade de *cache*", utilizada por muitos autores antes do surgimento das F-RANs, foi substituído por F-AP, que, por definição, são estações com esse tipo de capacidade.

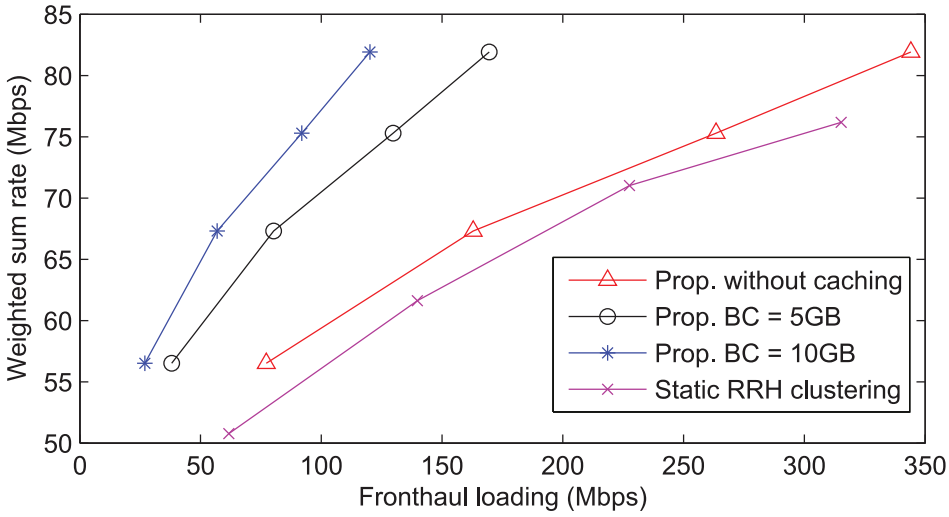
3.2.1 *Randomized User-Centric Clustering for Cloud Radio Access Network with PHY Caching*

O trabalho desenvolvido por Liu, Lau e Han (2015) apresenta duas contribuições para arquitetura C-RAN, que também podem ser aplicadas às F-RANs. A primeira contribuição consiste em uma formação dinâmica de *clusters* de F-APs para realizar transmissão conjunta e redundante de dados a fim de aumentar a potência de transmissão e diminuir a perda de dados. Como essa proposta apresenta conceitos e contribuições fora do escopo do trabalho, ela não será detalhada. A segunda contribuição, porém, está diretamente ligada ao conteúdo dessa dissertação. Ela consiste em uma estratégia de *caching* aleatória na camada física de uma C-RAN, que pode ser sumarizada em três diferentes passos:

- No primeiro passo, os pacotes de um arquivo requisitado por vários usuários (portanto popular) é divido em pacotes menores em um servidor de conteúdo.
- O segundo passo consiste em descarregar pacotes aleatoriamente escolhidos e os transmitir para os F-APs em horários de menor intensidade de uso da rede. Os F-APs salvam esses pacotes em sua *cache* local.
- O terceiro e último passo acontece quando um dispositivo de usuário requisita um arquivo.

Nele, o F-AP avalia e envia para os dispositivos os pacotes contidos em sua *cache* local e que fazem parte do arquivo requisitado. Os pacotes que não são encontrados em *cache* são requisitados para o servidor através do *fronthaul*

Figura 5: Comparação dos resultados da simulação utilizando clusterização estática, clusterização dinâmica, clusterização dinâmica com cache de 5 GB e 10 GB



Fonte: (LIU; LAU; HAN, 2015)

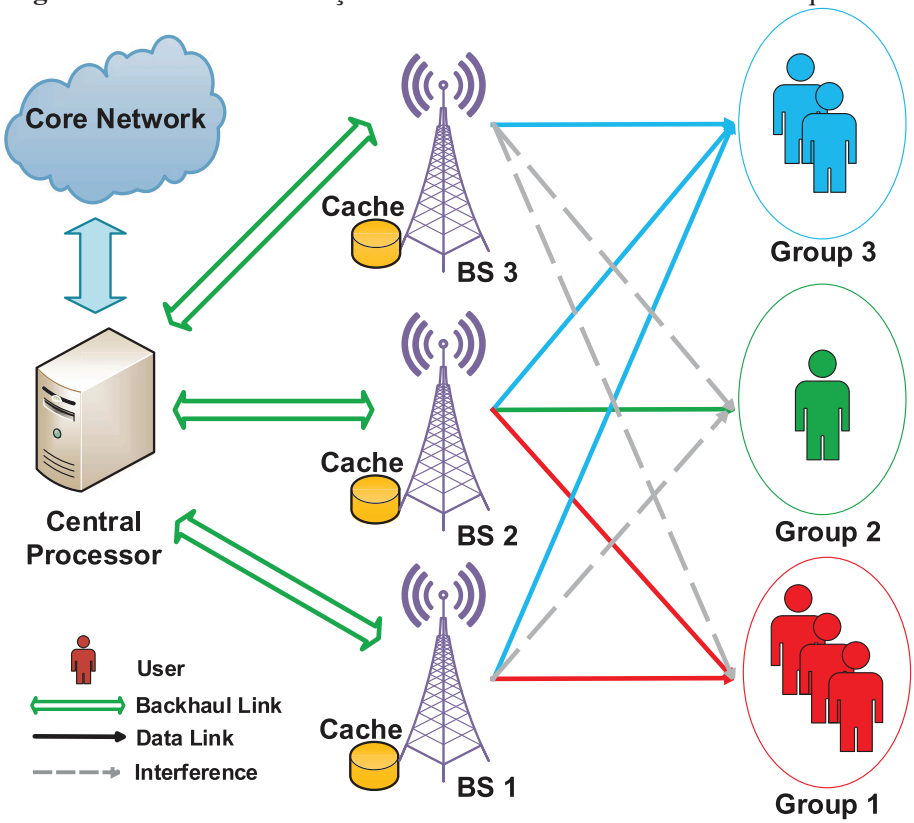
Os autores realizaram uma simulação considerando 49 F-APs de 2 antenas formando 7 hexágonos regulares. Sua simulação realiza a distribuição de 12 usuários de maneira aleatória e considera 50 arquivos de 1 GB cada. A requisição dos arquivos segue a distribuição Zipf com obliquidade 1,5. Os resultados mostram que a formação de *cluster* proposta apresenta um melhor desempenho em termos de tráfego no *fronthaul*. O mais interessante é que quando o cache é ativado, o tráfego no *fronthaul* diminui significativamente, como demostrado na Figura 5, o que reforça a importância da utilização de *cache* para redução desse gargalo.

3.2.2 *Content-Centric Sparse Multicast Beamforming for Cache-Enabled Cloud RAN*

Similar ao trabalho previamente analisado, a proposta apresentada por Tao et al. (2016) consiste em formações de *clusters* de F-APs baseadas no conteúdo requisitado por grupos de usuários. Esses *clusters* realizam a transmissão desse conteúdo de forma paralela através de *beamforming*, como demonstrada na Figura 6. Apesar do foco desse trabalho ser encontrar a melhor estratégia para essa paralelização, os autores utilizam três estratégias heurísticas de *caching* em sua simulação: baseado em popularidade, em probabilidade e aleatório, o que o torna relevante para essa dissertação.

De acordo com os resultados de sua simulação, a estratégia de *caching* por popularidade apresenta um melhor desempenho na maior parte dos casos, com exceção de quando há um menor números de usuários na rede, onde a estratégia com base na probabilidade demonstra os

Figura 6: Modelo de formação de clusters com base no conteúdo requisitado



Fonte: (TAO et al., 2016)

melhores resultados. Isso ocorre pois, com a estratégia de popularidade, os F-APs geralmente irão conter em suas *caches* conteúdos bastante similares e acabam sendo subutilizados por poucos usuários. Já com a estratégia probabilística, a diversidade de conteúdo é maior entre os F-APs que podem, em conjunto, servir mais conteúdos para esse pequeno número de usuários.

3.2.3 *Octopus: A Cooperative Hierarchical Caching Strategy for Cloud Radio Access Networks*

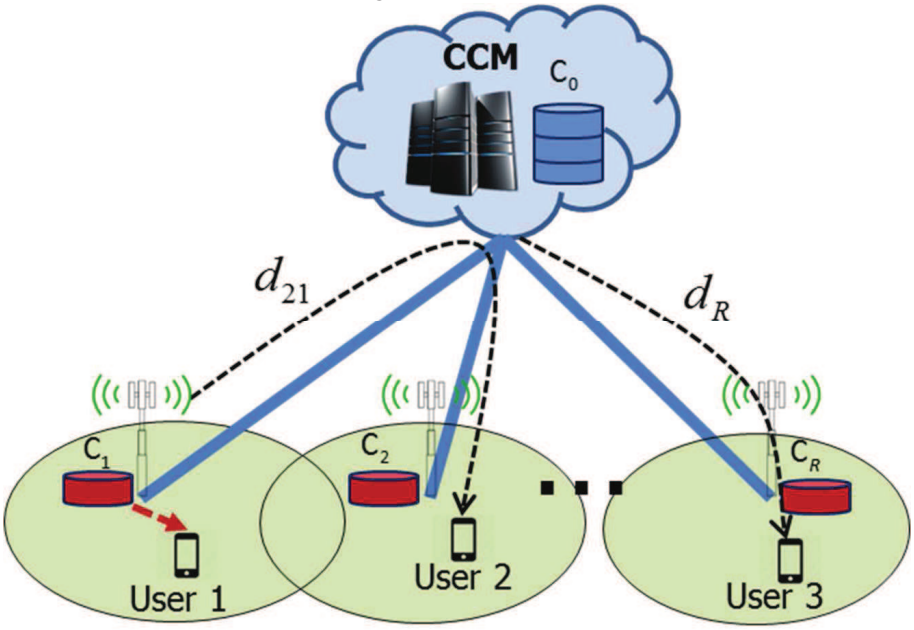
O trabalho de Tran e Pompili (2016) propõe uma estratégia hierárquica para realização de *caching* em RANs, chamada de *Octopus*. Essa estratégia possui duas locações distintas para realização de *caching*: na *Cloud Processing Unit* (CPU), que consiste em uma *cache* centralizada, próximo ao BBU *pool*; e nos dispositivos de borda, de modo distribuído. Os autores provam que o problema é NP-Difícil e propõe uma abordagem heurística de baixa complexidade que consiste em uma distribuição proativa dos arquivos em *caches* remotos. Essa abordagem foi construída usando um algoritmo guloso que, segundo os autores, consegue atingir pelo menos metade da solução ideal para esse problema.

Em seu trabalho, os autores não consideram a transmissão paralela de conteúdo e assumem que o sistema possui um controlador central, o qual tem acesso a todas as requisições dos usuários e é responsável por aplicar e atualizar a política de *caching* na rede. Eles também

mencionam a possibilidade desse controlador manter um índice local com os arquivos na *cache* distribuída e aplicar uma política de substituição de conteúdo.

O que diferencia esse trabalho de outros que propõem similar abordagem, como as apresentadas por Wang et al. (2014b) e Ahlehagh e Dey (2012), é o fato de que a estratégia de *caching* proposta não só adiciona uma camada a mais na arquitetura de rede, mas permite com que diferentes dispositivos de borda troquem dados gravados em *cache* através do *fronthaul*. Como pode ser observado na Figura 7, quando um usuário (*User 2*) requisita um arquivo armazenado em *cache* no dispositivo *C1*, o controlador central (CCM) coordena os componentes para que o arquivo seja transmitido de *C1* para *C2* através do *fronthaul*. Já o usuário *User 3*, requisita um arquivo que não está armazenado na *cache* de nenhum dispositivo de borda, então ele é resgatado da CPU.

Figura 7: Estratégia de *cache* utilizando *Octopus*



Fonte: (TRAN; POMPILI, 2016)

Os autores realizaram uma simulação comparando o *Octopus* com duas estratégias conhecidas, LFU e LRU (discutidas na Seção 2.2), e com duas estratégias recentes, ExMPC (BORST; GUPTA; WALID, 2010) e FemtoX (SHANMUGAM et al., 2013). Os resultados da simulação sugerem que o Octopus apresenta uma maior taxa de acerto das requisições de arquivos, o que também implica em uma menor latência e uma menor utilização do *fronthaul*.

3.2.4 *Joint Optimization of Cloud and Edge Processing for Fog Radio Access Networks*

O sistema de *caching* proposto por Park, Simeone e Shamai (2016) consiste em um modelo dividido em duas fases. Na primeira fase, denominada *pre-fetching*, nB_i bits da biblioteca de arquivos são escolhidos para serem armazenados nos F-APs. A política dessa fase utiliza

informações de longo prazo sobre a distribuição de popularidade, tamanho da *cache*, utilização do *fronthaul* e tamanho do arquivo para determinar a estratégia que será utilizada. Os autores mencionam as seguintes estratégias:

- ***Cache Most Popular***: Nessa estratégia, todos os F-APs irão fazer o *caching* do mesmo conjunto de arquivos mais populares. Os autores esperam que essa seja uma boa estratégia para quando apenas alguns poucos arquivos populares são requisitados pelos usuários.
- ***Cache Distinct***: Quando os arquivos não possuem grandes diferenças de popularidade, os autores afirmam que pode ser mais vantajoso armazenar a maior diversidade de arquivos possível. Nessa estratégia, cada F-AP realiza *caching* de diferentes arquivos, sendo que um arquivo não pode estar armazenado em mais de um F-AP.
- ***Fractional Cache Distinct***: A utilização de *Cache Distinct* impossibilita a transmissão paralela de um mesmo arquivo através de múltiplos F-APs, já que esse arquivo estará em somente uma delas. Isso é um problema em situações em que a capacidade do *fronthaul* é limitada. Uma solução é a utilização da *Fractional Cache Distinct*, onde os arquivos são divididos em segmentos que são aleatoriamente distribuídos entre os F-APs.

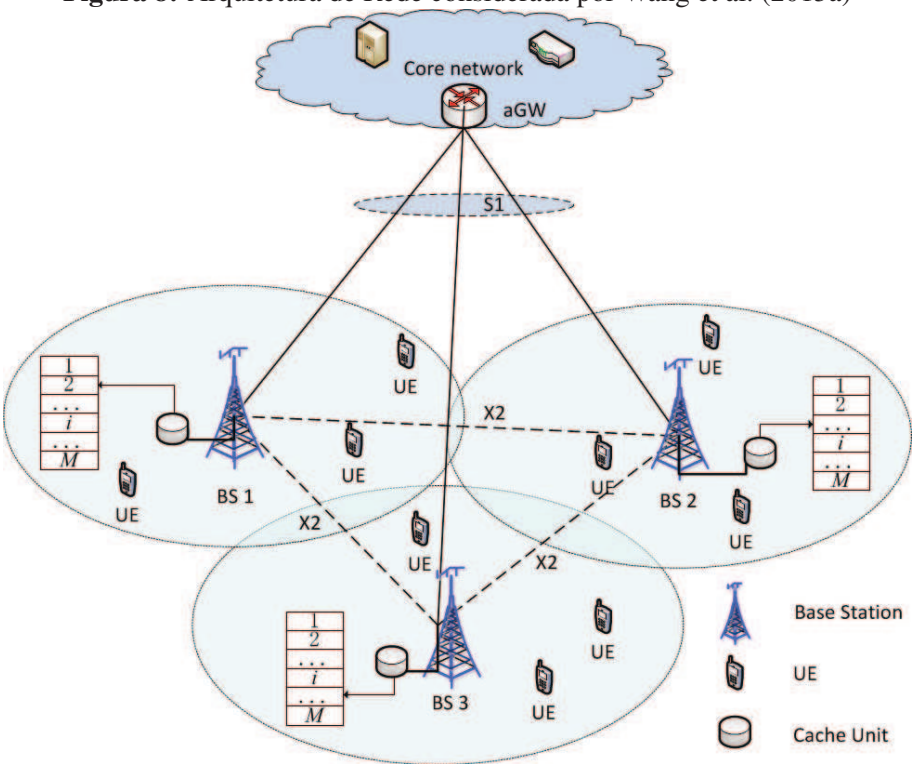
A fase de *pre-fetching* ocorre em intervalos maiores de tempo, onde assume-se que a popularidade dos arquivos se mantém constante. Já a segunda fase do modelo, chamada *delivery*, ocorre em cada intervalo de transmissão de dados e é responsável pela geração do sinal contendo tanto conteúdo em *cache*, quanto conteúdo buscado do BBU *pool*. Como a geração do sinal está fora do escopo desse trabalho, ela não será discutida.

Por fim, os autores realizaram uma simulação de seu modelo. Os resultados demonstram o ganho da utilização de *caching* em F-RANs e a importância da escolha correta da estratégia de *caching* dependendo da popularidade dos arquivos, já que quando a popularidade dos arquivos era similar, a rede obteve maiores taxas de transferência quando *Cache Distinct* era aplicada. Já quando alguns arquivos tinham uma maior popularidade, a utilização da *Cache Most Popular* resultava em maiores taxas de transmissão.

3.2.5 *Distributed Edge Caching Scheme Considering the Tradeoff Between the Diversity and Redundancy of Cached Content*

O trabalho apresentado por Wang et al. (2015a) faz uma análise do *trade-off* entre a diversidade e a redundância de conteúdo armazenado em *cache* nos dispositivos de borda. Para tal, os autores assumem uma arquitetura ligeiramente diferente da considerada por essa dissertação. Como pode ser observado na Figura 8, nessa arquitetura cada RHH (representada como uma *base station* (BS)) tem uma conexão de *fronthaul* exclusiva com a parte centralizada da rede. Já na arquitetura apresentada por Peng et al. (2016), todas as RHHs compartilham do canal de comunicação com o BBU *pool*.

Figura 8: Arquitetura de Rede considerada por Wang et al. (2015a)



Fonte: (WANG et al., 2015a)

Os autores abordam o problema utilizando os dois extremos como exemplo. Quando todos os F-APs fazem o *caching* dos arquivos mais populares, não há necessidade da troca de informação entre elas, diminuindo o custo de comunicação para zero. Porém, já que a diversidade de arquivos na borda da rede é menor, isso implica em um maior tráfego no *fronthaul*. Já quando cada um dos F-APs fazem *caching* de diferentes arquivos, o número de acessos ao *fronthaul* diminui, porém, o custo de comunicação entre os F-APs e sua latência aumentam. Essa situação é conhecida como o paradigma *exploitation vs exploration* (BASTUG; BENNIS; DEBBAH, 2014).

A solução dos autores utiliza um algoritimo heurístico de otimização chamado de otimização por enxame de partículas (em inglês: *particle swarm optimization* ou PSO) (POLI; KENNEDY; BLACKWELL, 2007), que foi adaptado para considerar os parâmetros de custo de comunicação, popularidade dos arquivos e tamanho da *cache*. Através de uma simulação Monte-Carlo, os autores chegaram à conclusão de que a diversidade de arquivos deve ser uma função que leve em consideração a distribuição de popularidade dos arquivos e a relação entre o custo de transmissão do *fronthaul* e custo de transmissão entre diferentes F-APs. De acordo com a simulação dos autores, o tamanho da *cache* não tem uma influência significativa nos resultados obtidos através da utilização o esquema proposto.

3.2.6 *iCacheOS In-RAN Caches Orchestration Strategy through Content Joint Wireless and Backhaul Routing in Small-Cell Networks*

No trabalho de Li, Hu e Ci (2015), os autores propõem o *iCacheOS*, uma estratégia de orquestração de *caching* em redes de acesso. Nela, objetos em *cache* local são compartilhados de foma cooperativa entre os F-APs através de suas conexões sem fio e do *fronthaul*. Nessa estratégia, um orquestrador central é responsável pela coordenação das *caches* distribuídas. Esse componente possui as seguintes informações: quais conteúdos estão em *cache* em cada F-AP, quais usuários estão conectados em cada F-AP, e quais são as conexões existentes entre eles. A partir dessa informação, quando um usuário requisita um arquivo, o orquestrador realiza os seguintes passos:

1. Caso o conteúdo requisitado esteja armazenado em *cache* no F-AP no qual o usuário está conectado, ele é enviado para o usuário usando somente os recursos locais. Caso contrário, o passo 2 é executado.

2. Nesse passo, orquestrador verifica se o arquivo solicitado está armazenado na *cache* de outro F-AP. Caso o arquivo não esteja, a requisição do usuário é então encaminhada para a rede central ou para um servidor de arquivos localizado fora da rede de acesso.

3. Caso arquivo solicitado esteja em alguma outro F-AP, o orquestrador coordena os componentes de rede necessário para que o arquivo seja transmitido dessa outro F-AP para o usuário, seja via rede sem fio, ou através de roteamento passando pelo *fronthaul*.

Os autores propõem a utilização de um algoritmo que faz uso de técnicas de redução de complexidade por métricas heurísticas, como por exemplo, Relaxação Lagrangeana, com o objetivo de obter a melhor solução possível para obtenção de uma rota para o conteúdo dentro da rede de acesso, visando obter uma menor latência na transmissão desses dados.

Para avaliação da estratégia proposta, os autores realizaram uma simulação comparando o iCacheOS com outras quatro estratégias em cenários onde os usuários eram estáticos e em cenários onde eles se movimentavam dentro da rede. As estratégias comparadas foram: a) *CluCacher*, que tenta agrupar os usuários que fazem o mesmo tipo de requisição com F-APs que possuem tal conteúdo, utilizando somente a rede sem fio (sem a participação do *fronthaul*); b) *POP+LRU*, que implementa uma política de *caching* utilizando parâmetros de popularidade e uma política de substituição baseada em LRU; c) *GlobeCacher*, que provém a melhor solução possível, pois essa estratégia possuí um pré-conhecimento de todas as requisições futuras, e portanto, é utilizada como um benchmark para a solução; d) *MobiCacher* (GUAN et al., 2014) que implementa uma estratégia de *caching* cooperativa com o pré-conhecimento da mobilidade do usuário na rede.

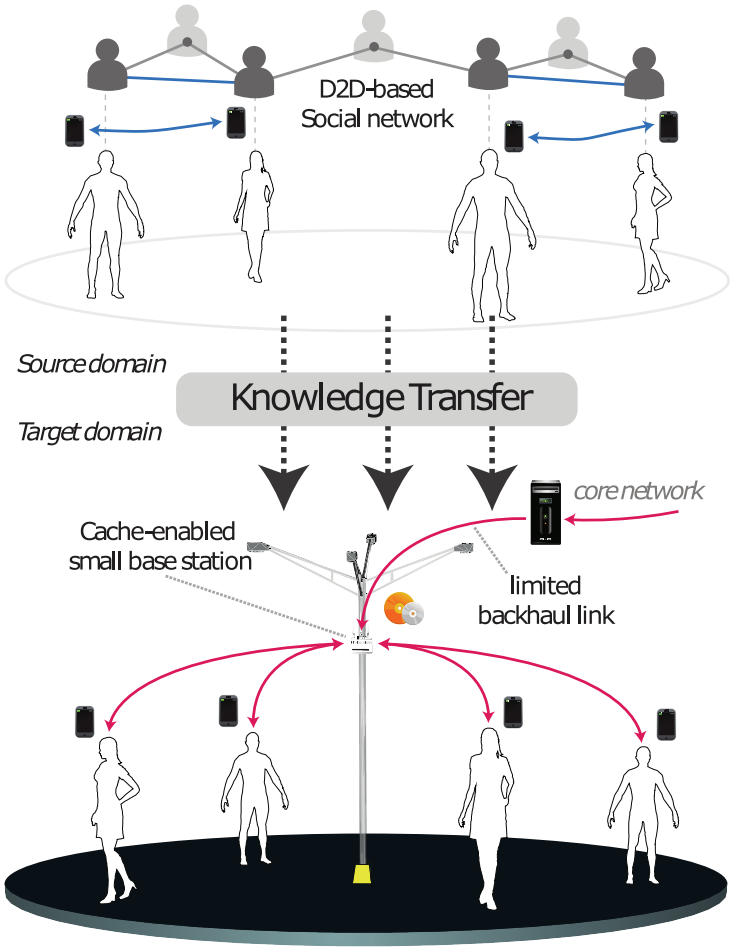
Os resultados obtidos através da simulação sugerem que, em média, iCacheOS obtém uma latência de transmissão 52,5% menor quando comparado ao *POP+LRU* e de 17,5% quando

comparado com *CluCacher*. Em cenários onde o usuário se movimenta na rede, iCacheOS obteve resultados muito similares aos obtidos pelo *MobiCacher*. Com isso, os autores concluem que o iCacheOS possui um melhor *trade-off* entre resultados e praticabilidade de implementação quando comparado com outras estratégias.

3.2.7 *A Transfer Learning Approach for Cache-Enabled Wireless Networks*

Os pesquisadores Bastug, Bennis e Debbah (2015) apresentam o primeiro trabalho na área de *caching* para redes sem fio a utilizar aprendizado de máquina para, através das interações dos usuários, determinar o conteúdo a ser armazenado em *cache*. Eu seu trabalho, eles alavancam a utilização de um *framework* de aprendizado de máquina não assistido chamado de transferência de conhecimento (em inglês: *transfer learning* ou *TL*). Com esse *framework*, é possível realizar a transferência de conhecimento entre um domínio com mais dados para um outro domínio onde é difícil, ou até mesmo impossível, realizar a coleta de dados de treino a fim de enriquecer os seus modelos de predição (PAN; YANG, 2010).

Figura 9: Utilização de transferência de conhecimento para determinar o conteúdo a ser armazenado em *cache*



Fonte: (BASTUG; BENNIS; DEBBAH, 2015)

Como demonstrado na Figura 9, os autores consideram dois domínios de conhecimento. O primeiro, consiste nas interações sociais diretas entre os usuários (D2D) em sua comunidade. Especificamente, esse domínio contém o comportamento dos usuários modelado de acordo com o processo estocástico do restaurante chinês (em inglês: *chinese restaurant process* ou CRP) (ALDOUS, 1985). Já o segundo domínio é formado pelos F-APs, os usuários conectados a elas e o conteúdo requisitado por esses usuários. A transferência de conhecimento do primeiro para o segundo ocorre em duas fases. Na primeira fase, é estabelecida a correspondência do conteúdo do primeiro domínio com o segundo. Na segunda fase, os dados de popularidade de arquivo obtidos através das interações sociais dos usuários são considerados juntamente com os valores obtidos em tempo real, a fim de obter uma melhor estimativa de quais arquivos serão requisitados em um futuro próximo e, portanto, dever ser incluídos na *cache* de dado F-AP.

Os autores realizaram uma simulação comparando sua proposta com outras estratégias, que sugere que a utilização de algum conhecimento prévio sobre a popularidade dos arquivos realmente gera uma menor carga no *fronthaul*. Eles enfatizam que os parâmetros da simulação que mais impactaram os resultados foram o tamanho disponível para *caching*, a distribuição de probabilidade de um arquivo ser requisitado, a intensidade com que os usuários fazem a requisição de arquivos, a capacidade de transmissão do *fronthaul* e o quão fiel é a correspondência entre o conhecimento do domínio fonte (D2D) e de destino (F-RAN).

3.2.8 *Backhaul Traffic Minimization under Cache-Enabled CoMP Transmissions over 5G Cellular Systems*

Os autores Yu, Tsai e Pang (2016) afirmam em seu trabalho que, mesmo possuindo *caching*, se não houver uma coordenação de conteúdo nos F-APs, a transmissão conjunta de dados ainda é uma das principais causas para um alto tráfego de dados no *fronthaul*. Para resolver esse problema, os autores propõem um algoritmo para formação de *clusters* de F-APs para realização da transmissão conjunta de um mesmo dado, considerando o conteúdo em *cache* presente em cada F-AP, com o objeto de diminuir o tráfego no *fronthaul*.

Os desafios enfrentados pelos pesquisadores consistem em determinar quantos F-APs devem ser agrupadas em um *cluster* de transmissão conjunta e quais F-APs devem fazer parte dessa transmissão. Seu algoritmo funciona da seguinte maneira: quando um usuário faz a requisição de um arquivo, todos os F-APs que contém o arquivo são adicionadas ao *cluster*. Se não há F-APs com o arquivo em *cache*, os F-APs escolhidas consiste no menor número de F-APs necessárias para satisfazer os requisitos de qualidade da transmissão. Ao escolher poucos F-APs, o tráfego no *fronthaul* é reduzido, já que menos F-APs fazem requisições para entregar o mesmo conteúdo para um usuário. Também visando diminuir o número de F-APs necessárias, o algoritmo prioriza as F-APs com a maior relação sinal-ruído (em inglês: *signal-to-noise ratio* ou SNR) (JOHNSON, 2006) para se juntar ao *cluster*, resultando em uma menor alocação de recursos.

Por fim, os autores compararam seu algoritmo proposto com um algoritmo similar, mas que não considera o conteúdo em *cache* nos F-APs através de um modelo de simulação. Eles utilizaram dados de uma implementação real de uma rede LTE com 132 estações de base. Cada estação de base consegue armazenar de 1 mil a 4 mil vídeos de uma biblioteca de 10 mil vídeos, que podem ser selecionados por até 600 usuários distribuídos aleatoriamente na região. Os resultados das simulações sugerem que a simples utilização do conhecimento sobre o conteúdo em *cache* consegue diminuir o tráfego do *fronthaul* em até 62%.

3.2.9 *Minimum Delay Guaranteed Cooperative Device-to-Device Caching in 5G Wireless Networks*

No seu trabalho, Amentie et al. (2016) apresentam um esquema de realização e substituição de conteúdo em *cache* que considera a capacidade limitada de armazenamento dos dispositivos de borda, a popularidade dos arquivos solicitados e o processo de acesso à rede. A grande diferença desse trabalho é a capacidade de colaboração entre os dispositivos dos usuários através da comunicação D2D, sob responsabilidade do F-AP, a qual coordena e gerencia o conteúdo armazenado em *cache*. Portanto, seu modelo foi limitado a redes formadas por um único F-AP.

Com a finalidade de realizar as tarefas de gerenciamento de *cache*, cada F-AP tem acesso às seguintes informações: 1) o histórico de requisições de cada usuário na rede; 2) a popularidade do conteúdo, tanto localmente quanto globalmente e 3) a densidade de usuários e sua localização da rede. Com essas informações, o seguinte esquema de *caching* é proposto. Para todos os usuários no *cluster*, o F-AP faz o *download* do conteúdo baseado em sua frequência de acesso. Quando esse conteúdo já não existe na rede interna, ele vem através da rede externa e é comparado com o conteúdo menos popular atualmente armazenado em *cache* para uma eventual substituição. Caso o conteúdo já exista no *cluster*, ele é transferido através de comunicação D2D e, dependendo de sua frequência de acesso, pode substituir um conteúdo de menor popularidade.

Os pesquisadores então criaram um modelo de simulação para avaliar o desempenho de seu esquema. Eles compararam a solução proposta com outros três esquemas. No primeiro (*D2D only*), o *caching* nos F-APs foi desabilitado, ou seja, quando um arquivo não estava em *cache* nos dispositivos de usuário, eles acessavam a rede externa para buscar o conteúdo. No segundo (*F-AP only*), os dispositivos de usuário não possuem capacidade de *caching*, somente os F-APs. No terceiro e último (*Random*), tanto F-APs quanto dispositivos de usuário possuem capacidade de *caching*, porém ele é feito de maneira aleatória. Os resultados das simulações sugerem que o esquema proposto diminui significativamente a latência na comunicação quando comparado com o *F-AP only* e moderadamente quando comparado com o *Random*. O esquema proposto apresentou resultados ligeiramente melhores que o *D2D only*, o que sugere que, para esse cenário, a estratégia de realizar *caching* dos dados nos dispositivos de usuário e transmiti-los usando comunicação D2D apresenta melhores resultados do que a realização do *caching*

nos F-APs.

3.2.10 *Exploiting Caching and Multicast for 5G Wireless Networks*

A publicação de Poularakis et al. (2016) apresenta o que os autores chamam de um paradigma novo para a próxima geração de redes sem fio, visando atender a explosiva demanda de dados móveis e minimizando o gasto de energia. Esse paradigma é construído em dois pilares: transmissão conjunta de dados e *caching*. O diferencial desse trabalho é que os autores esperam que, em uma mesma rede, múltiplos usuários podem fazer a requisição de um mesmo conteúdo em uma mesma janela de tempo. Portanto, a rede pode fazer a transmissão desse arquivo de maneira conjunta utilizando o mesmo bloco de sinal.

Os autores propõem um algoritmo heurístico da categoria guloso (LEISERSON, 1990) para endereçar o problema. O algoritmo começa com todos as *caches* sendo vazios. A cada iteração, o algoritmo verifica através de uma função de otimização qual o arquivo que resultaria em um menor consumo de energia e o armazena em *cache*. Essa função considera diversos parâmetros, como custo energético tanto para armazenar quanto para transmitir o arquivo, demanda média do arquivo na área afetada, probabilidade de o arquivo ser solicitado dado uma janela de tempo, etc. Quando a *cache* estiver cheia, o algoritmo encerra sua execução. Como pode ser observado, esse algoritmo requer uma série de parâmetros sobre o conteúdo, os quais os autores assumem haver um conhecimento prévio, e seu foco principal acaba sendo a redução do consumo energético.

Para avaliar o desempenho do algoritmo proposto, os autores criaram um modelo de simulação emulando a implantação da rede em uma arena esportiva. Os resultados sugerem que os seguintes fatores são os que mais impactam nos resultados em termos energéticos: tamanho do *caching*, distância de transmissão das antenas, capacidade do *fronthaul* e padrões de demanda de arquivos dos usuários.

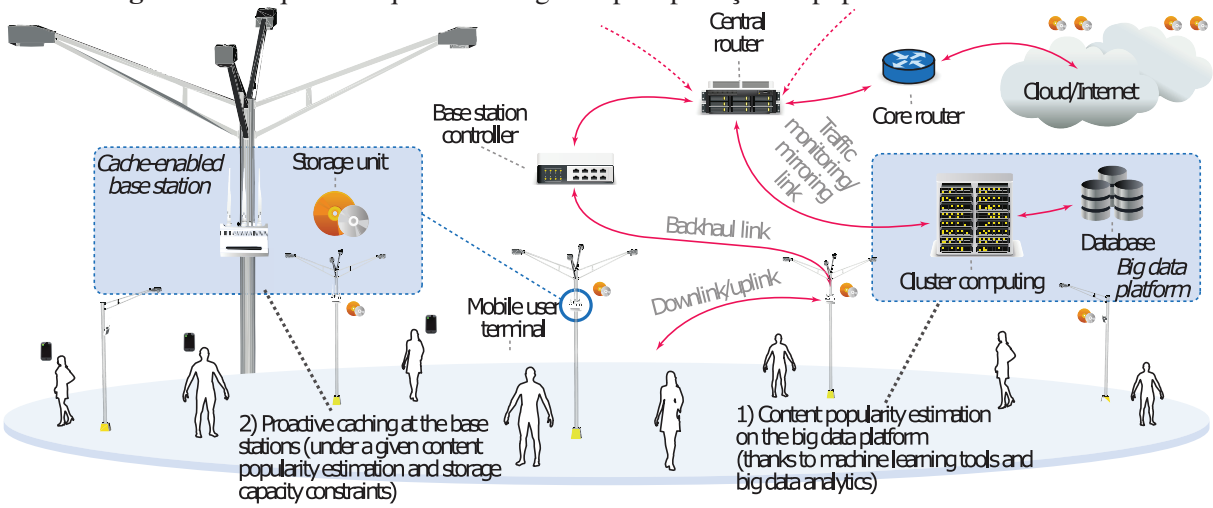
3.2.11 *Big Data Caching for Networking Moving from Cloud to Edge*

O trabalho proposto por Zeydan et al. (2016) apresenta uma arquitetura que integra técnicas de *big data* e explora a capacidade de *analytics* dessa tecnologia para estimar a popularidade dos conteúdos, permitindo seu armazenamento em *cache* de forma proativa.

Aproveitando-se da natureza previsível do comportamento humano, a arquitetura proposta pelos autores coleta dados contextuais dos usuários, como por exemplo histórico de navegação e localização, e realiza previsões sobre suas demandas espaço-temporais a fim de proativamente realizar o *caching* de conteúdo. A arquitetura proposta paraleliza a execução do algoritmo de predição no núcleo da F-RAN. Para tal, os autores sugerem a utilização de sistemas conheci-

dos para processamento massivo de dados, como por exemplo o *Hadoop*[6] e *Spark*[7], que seria localizado em um local centralizado da rede, como demonstrado na Figura 10. Sua arquitetura, então, utiliza uma abordagem gulosa para realizar o armazenamento dos arquivos com uma maior popularidade estimada em *cache* nos F-APs. Os autores não informam mais detalhes sobre como esse algoritmo ou ferramentas de análise utilizam os dados de contexto para estimar a popularidade de um arquivo.

Figura 10: Arquitetura que utiliza big data para predição de popularidade de conteúdo



Fonte: (ZEYDAN et al., 2016)

Para avaliação dessa arquitetura, os pesquisadores coletaram dados reais de tráfego em mais de 10 redes de núcleo na Turquia, que transferiu mais de 15 bilhões de pacotes para um sistema *Hadoop*, onde os dados passaram por o processamento necessário a fim de extrair informações relevantes para o estudo (BAŞTUğ et al., 2015). O resultado de suas simulações sugere que a capacidade de armazenamento é um parâmetro crucial para métricas de qualidade, sendo que quando há um espaço limitado, a proposta dos autores apresenta um resultado bastante similar ao ideal. Os autores relatam que quando existem arquivos grandes, mas com popularidade relativamente baixa, isso pode causar um tráfego muito grande no *fronthaul*. Portanto, os autores sugerem que trabalhos futuros também levem em consideração o tamanho do arquivo para conseguir lidar de forma melhor uma demanda realista de conteúdo.

3.2.12 *Socially Enabled Wireless Networks Resource Allocation via Bipartite Graph Matching*

Os autores Wang et al. (2015b) apresentam uma visão social de alocação de recursos através de grafos, que é estendido para redes sem fio, integrando características de comportamento dos usuários, assim como do conteúdo, para realização do *caching*. Para tal, os autores consideram

[6]http://hadoop.apache.org
[7]http://spark.apache.org

comunicação D2D para formação de pares e *clusters* via grafos bipartidos (DIESTEL, 2010).

Em termos de alocação de recursos, a solução proposta é descrita a seguir. Primeiramente, considerando um conjunto de usuários que possuem *caching* em seus dispositivos, cada usuário seleciona seus "parceiros" de transmissão caso eles sejam o destino dos dados ou caso eles sejam confiáveis e o canal de comunicação seja estável suficiente para suportar a formação de uma sub-rede. Então, esses parceiros são representados através de um grafo bipartido, que contém as características necessárias para aplicação do teorema de Hall (HALL, 1986). Os autores utilizam similaridades de interesse e relações sociais para formar os *clusters*, e pode ser que um usuário possa armazenar mais conteúdo em *cache* do que o necessário, a fim de servir um parceiro que está consumindo o mesmo dado. Após seleção de um parceiro, é feita a alocação de recursos de rede necessários para manter uma alta qualidade de comunicação entre esses dois dispositivos.

Os autores apresentam uma análise numérica do conceito apresentado, que sugere a viabilidade da proposta. Infelizmente, os pesquisadores não apresentam resultados práticos ou simulados de sua pesquisa, o que dificulta sua comparação com outros métodos. Porém, eles enfatizam que os próximos passos de sua pesquisa consistem em aumentar a robustez do modelo visto que as estimativas utilizadas podem não corresponder àquelas de um cenário real.

3.2.13 *Cooperative Content Distribution for 5G Systems Based on Distributed Cloud Service Network*

No seu trabalho, os autores Jiang, Feng e Qin (2015) propõem um *framework* para realização de *caching* de forma cooperativa implementando o algoritmo de distribuição de conteúdo baseado em estado (em inglês: *State based Content Distribution* OU SCD). Em seu modelo, os provedores fazem a instalação de diversos *cloudlets*, formando o que é chamado de *Distributed Cloud Service Networks* (DCSN), e o modelo proposto se encarrega de realizar a distribuição de conteúdo de forma otimizada afim de evitar transmissão redundante de dados da rede de núcleo para a DCSN.

No seu modelo, cada *cloudlet* serve uma determinada região e é, portanto, responsável por lidar com requestes locais, coletar informações sobre atualizações do conteúdo em *cache* e na rede externa e decidir o que será armazenado em *cache*. Duas *cloudlets* podem colaborar e compartilhar conteúdo entre si. Além disso, a rede externa pode proativamente enviar conteúdo a ser armazenado em *cache* nas *cloudlets*, porém, essas operações são feitas em momentos de menor uso da rede.

Como mencionado anteriormente, os autores também implementam um algoritmo de SCD com o objetivo de diminuir a latência média da rede. Nesse algoritmo, quando um usuário faz uma requisição de um arquivo de mídia, a *cloudlet* irá enviar o conteúdo diretamente para o usuário caso esse estiver armazenado em sua *cache*. Caso contrário, esse conteúdo é buscado da rede externa ou de outros *cloudlets*. Esses dados são periodicamente redistribuídos baseado

no estado desse conteúdo, que é atualizado pelos provedores de conteúdo e repassado para as *cloudlets* que o requisitaram.

Na avaliação de desempenho proposta pelos autores, eles compararam sua estratégia de *caching* com outras estratégias conhecidas como LRU, LFU e RR. Quando comparando o número de vezes que o conteúdo foi baixado da *cache* ao invés da rede externa, os resultados sugerem que a proposta dos autores superam LRU, LFU e RR em 15%, 27% e 76% respectivamente. Esses valores são 14%, 26% e 72% quando comparando a latência média na entrega de conteúdo. Os resultados também sugerem uma redução do tráfego independentemente do tamanho da *cache*, e os autores atribuem esse ganho à cooperação entre múltiplos *cloudlets*.

3.2.14 *Edge Caching for Layered Video Contents in Mobile Social Networks*

Os autores do trabalho (SU et al., 2017) afirmam que é previsto *caching* nos componentes de borda será um dos paradigmas mais importantes no futuro da Internet. Eles também afirmam que as técnicas existentes não podem ser aplicadas para *caching* de vídeos, visto poder de armazenamento limitado dos componentes de borda e a quantidade de dados ocupadas por esse tipo de conteúdo.

Para solucionar esse problemas, os autores propõem uma modelo teórico de armazenamento de vídeos em camadas, onde os dados são armazenados em nós de *caching* localizados entre os usuários e o servidor de vídeos. Nesse modelo, os usuários são agrupados em conjuntos baseados em sua interação através de redes sociais, e o seu diferencial é determinar o tamanho de armazenamento necessário para suprir a necessidade e manter a qualidade para um dado grupo de usuários. Para tal, a proposta baseia-se em relacionados aos interesses desses grupos, assim como na popularidade dos arquivos.

Uma vez determinado o espaço de armazenamento reservado ao grupo, o modelo armazena camadas dos vídeos requisitados, de forma que as camadas já armazenadas na *cache* são retornadas aos usuários que fazem requisição do vídeo, e as demais são então transmitidas pelo servidor de vídeo através da rede.

Os testes realizados através de simulações compararam o modelo de alocação proposto com o esquema de alocação uniforme (em inglês *Uniform Cache Allocation*, ou UCA), onde cada grupo de usuário recebe a mesma quantia de recursos e com o esquema de alocação aleatório (em inglês *Random Cache Allocation*, ou RCA). Os resultados obtidos sugerem que o esquema de alocação proposto é mais estável do que os demais esquemas, resultando em um menor atraso de rede.

3.3 Análise dos Trabalhos Relacionados e Lacuna de Pesquisa

Primeiramente, essa seção apresenta a análise dos trabalhos realizados considerando três aspectos levantados a área de pesquisa. O primeiro deles é a localização do *caching* dentro da

arquitetura da rede, ou seja, em quais nós da rede é feito o *caching* de conteúdo. O segundo diz respeito a capacidade de realizar o *caching* de conteúdo proativamente. Finalmente, o terceiro explora a capacidade da rede F-RAN de fornecer informações úteis dos componentes de rede, mais especificamente o histórico de requisições dos F-UEs para tomada de decisões relacionadas ao *caching* de conteúdo.

Ao fazer a análise da localização do *caching* nos trabalhos propostos, fica claro que eles concentraram seus esforços em uma das duas localizações possíveis: ou eles armazenam o cache em F-APs ou em F-UEs, sendo que a grande maioria foca os esforços em realizar o *caching* de conteúdo somente nas F-APs. Os resultados obtidos por esses trabalhos podem trazer ganhos significativos nesse primeiro momento das F-RANs. Porém, a partir do instante em que essa arquitetura de rede passar a apresenta múltiplas camadas (BYERS, 2017), os ganhos possíveis com a hierarquização da arquitetura da rede acabam sendo reduzidos, visto a falta de cooperação entre os nós da rede.

No contexto dessa análise, *caching* proativo foi considerado como a capacidade de um modelo de *caching* prever quais arquivos ou segmentos de arquivos serão mais requisitados por uma dada área ou um dado instante com a finalidade de armazená-los em *cache* antecipadamente. Poucos trabalhos analisados possuem algum tipo de proatividade, já que a maior parte deles realiza o *caching* conforme os usuários requisitam os arquivos, ou faz essa ação utilizando uma popularidade previamente sabida, o que não é verdade em uma aplicação real (TATAR et al., 2014). Além disso, todos os trabalhos que fazem o *caching* proativo utilizam algum componente centralizado para realizar a predição de qual conteúdo deve ser armazenado, não fazendo-a continuamente durante a operação da rede.

Tabela 1: Comparação dos trabalhos relacionados e a lacuna preenchida pelo modelo proposto.

Publicação	***Caching* em F-AP**	***Caching* em F-UE**	***Caching* Proativo**	**Utiliza Histórico**
(LIU; LAU; HAN, 2015)	✓	✗	✗	✓
(TAO et al., 2016)	✓	✗	✗	✗
(TRAN; POMPILI, 2016)	✓	✗	✗	✓
(PARK; SIMEONE; SHAMAI, 2016)	✓	✗	✗	✗
(WANG et al., 2015a)	✓	✗	✗	✗
(LI; HU; CI, 2015)	✓	✗	✗	✗
(BASTUG; BENNIS; DEBBAH, 2015)	✗	✓	✓	✓
(YU; TSAI; PANG, 2016)	✓	✗	✗	✗
(AMENTIE et al., 2016)	✓	✗	✗	✓
(POULARAKIS et al., 2016)	✓	✗	✗	✓
(ZEYDAN et al., 2016)	✗	✓	✓	✓
(WANG et al., 2015b)	✓	✗	✗	✗
(JIANG; FENG; QIN, 2015)	✓	✗	✗	✓
(SU et al., 2017)	✓	✗	✗	✓
Nuoxus	✓	✓	✓	✓

Fonte: Elaborada pelo autor.

Em uma F-RAN, os componentes podem trabalhar em diferentes modos de transmissão, como apresentado na Subseção 2.1.3. O modo de transmissão selecionado tem um enorme impacto na qualidade da transferência de dados, tornando fundamental sua correta escolha (PENG et al., 2016). Portanto, o terceiro e último aspecto analisado foi a capacidade do modelo de fornecer dados relacionados ao estado de *caching* e do histórico de requisições para componentes externos ao modelo, visando a utilização de tais dados para tomadas de decisões, incluindo a seleção do modo de transmissão.

A comparação dos trabalhos relacionados através desses três aspectos é apresentada de forma resumida através da Tabela 1. Considerando as limitações dos modelos propostos até então, fica possível identificar a necessidade do desenvolvimento de um modelo que aproveite a capacidade de processamento e armazenamento das F-RANs para realizar o *caching* de conteúdo de forma proativa e contínua, atuando em qualquer camada da rede e capaz de fornecer essas informações para tomadas de decisões externas.

4 MODELO PROPOSTO - NUOXUS

Nesse capítulo, é apresentado o modelo de *caching* proativo proposto, batizado de Nuoxus. A Seção 4.1 descreve quais são as decisões de projeto relevantes para o modelo seu impacto na arquitetura. A Seção 4.2 aborda a arquitetura do modelo, definindo seus componentes e dependências. A política de substituição de conteúdo desenvolvida é descrita em mais detalhes na Seção 4.3. A Seção 4.4 entra em mais detalhes como o modelo realiza o *caching* de forma proativa. Já a Seção 4.5 sugere um ajuste no algoritmo de seleção do modo de transmissão de rede proposto por Peng et al. (2016) a fim de explorar a capacidade do Nuoxus de fornecer dados de *caching* para tomada de decisões por componentes externos ao modelo.

4.1 Decisões de Projeto

Por estar ainda no início de seu desenvolvimento, a arquitetura F-RAN não possui muitas definições fixas. Por exemplo, ainda não há um *hardware* especifico para essa tecnologia e toda sua visão ainda está muito simplista quando comparada com arquiteturas de rede sem fio existentes atualmente, como a LTE (TRAN; POMPILI, 2016). Além disso, como o modelo proposto tem o foco em conteúdo multimídia, então muitas das questões que normalmente são levantadas ao abordar *caching* não se aplicam ao modelo. Como por exemplo, não há motivos para realizar a validação do estado desse tipo de dado, já que ele é considerado estático, e portanto, não é atualizado.

Dado esse contexto, as seguintes decisões foram tomadas:

1. O Nuoxus pode ser executado em qualquer nó da rede, desde que esse possua capacidade de processamento e armazenamento.

2. A arquitetura do Nuoxus não deve depender da topologia da rede, desde que essa possa ser interpretada como uma árvore de nós, onde cada nó tenha um pai e n filhos. Para mais detalhes em relação à essa decisão de arquitetura, veja a Seção 4.2.

3. Uma das características de conteúdo multimídia é que eles normalmente são estáticos, ou seja, não são alterados depois de publicados. Portanto, assumindo essa premissa, o modelo não contempla validação do estado dos dados;

4. O modelo não possuí foco em nenhuma tecnologia de comunicação nem de armazenamento de dados específica, pois as F-RANs ainda estão em desenvolvimento e não há nenhuma especificação em termos da capacidade *hardware* que os componentes dessa arquitetura de rede possuirão (PARK; SIMEONE; SHAMAI, 2016).

5. Nesse modelo, não é considerado os gastos energéticos da rede, devido a incerteza do consumo de energia pelo *hardware* desenvolvido para redes 5G, visto que estes supor-

tam novas características físicas, inclusive um novo espectro de frequências (OSSEIRAN et al., 2014).

6. Apesar de ser proposto uma extensão do algoritmo de seleção do modo de transmissão de dados das F-RANs, como explorado na Seção 2.1.4, para também considerar o conteúdo em *cache*, o modelo Nuoxus em si apenas fornece tais informações ao algoritmo, não sendo responsável por selecionar o modo de transmissão.

4.2 Arquitetura

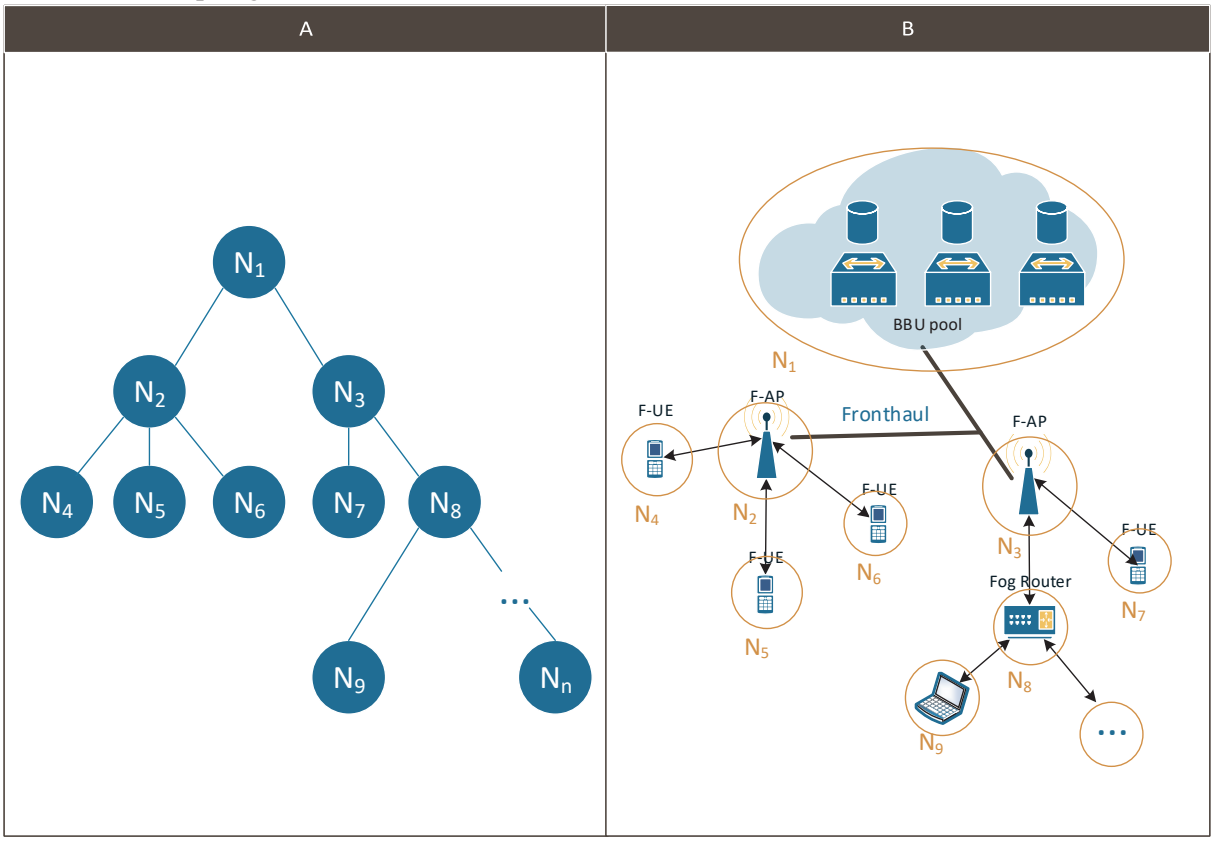
O modelo Nuoxus tem como principais objetivos a redução do tráfego de dados no *fronthaul* através do armazenamento de conteúdo multimídia nos nós intermediários da rede. Essa redução nos dados trafegados acarreta na diminuição da latência de comunicação entre os dispositivos de uma F-RAN devido a um menor número de colisões de pacotes. O Nuoxus foi concebido para ser executado individualmente em cada nó da rede, sendo que nem todos os nós precisam executá-lo. Isso porque não há um componente centralizado e cada nó é responsável por gerenciar seu próprio *caching* baseado somente nos dados de requisições locais. Por essa razão, o único conhecimento que o nó executando o Nuoxus precisa ter em relação ao restante da rede é qual é o seu nó pai e quais são seus nós filhos.

Como mencionado anteriormente, uma das decisões de projeto limita o Nuoxus a uma topologia de rede que pode ser interpretada como uma árvore, onde um nó possui um nó pai e qualquer número de nós filhos. Tal topologia é representada através da Figura 11, a qual contém um diagrama de uma rede como vista pelo Nuoxus e sua correspondência com uma F-RAN. Um nó N pode ter diversos filhos, e esses nós podem formar diversas camadas na árvore. Como cada nó contém um *cache* diferente, esses nós acabam criando uma arquitetura de *caching* em múltiplas camadas.

Utilizando o diagrama da Figura 11 para exemplificar um cenário real, pode-se dizer que N_1 representa o *BBU pool*, N_2 e N_3 representam F-APs, N_8 representa um dispositivo de comunicação intermediário com capacidade *Fog*, como um roteador (*F-Router*) e, finalmente, N_4, N_5, N_6, N_7 e N_9 representam F-UEs, que são os dispositivos de usuário. Como mencionado anteriormente, nem todos os nós precisam executar o Nuoxus. Nesse exemplo, caso N_3 não tenha capacidade de *caching* e, por consequência, não execute o Nuoxus, N_7 e N_8, assim como seus filhos, ainda podem executar o Nuoxus sem nenhum empecilho.

Cada nó N controla o histórico de requisições vindas de seus filhos diretos e, baseado em sua política de substituição, decide se o conteúdo requisitado é relevante para ser armazenado em sua *cache*, esperando que outros filhos conectados nesse nó também façam a requisição desse conteúdo. Além disso, baseado no histórico de requisições, o nó pode proativamente iniciar uma comunicação com os filhos com maior probabilidade de requisitar um dado conteúdo, sugerindo que esse filho faça tal requisição antecipadamente, a fim de salvar em sua *cache* antes mesmo do conteúdo ser requisitado por uma camada mais baixa da rede ou pelo usuário final.

Figura 11: a) Uma topologia de rede executando o Nuoxus. Cada nó N pode possuir n nós filhos, que podem formar uma árvore de m camadas. b) Correspondência entre os componentes que formam uma F-RAN com a topologia executando o Nuoxus.



Fonte: Elaborada pelo autor.

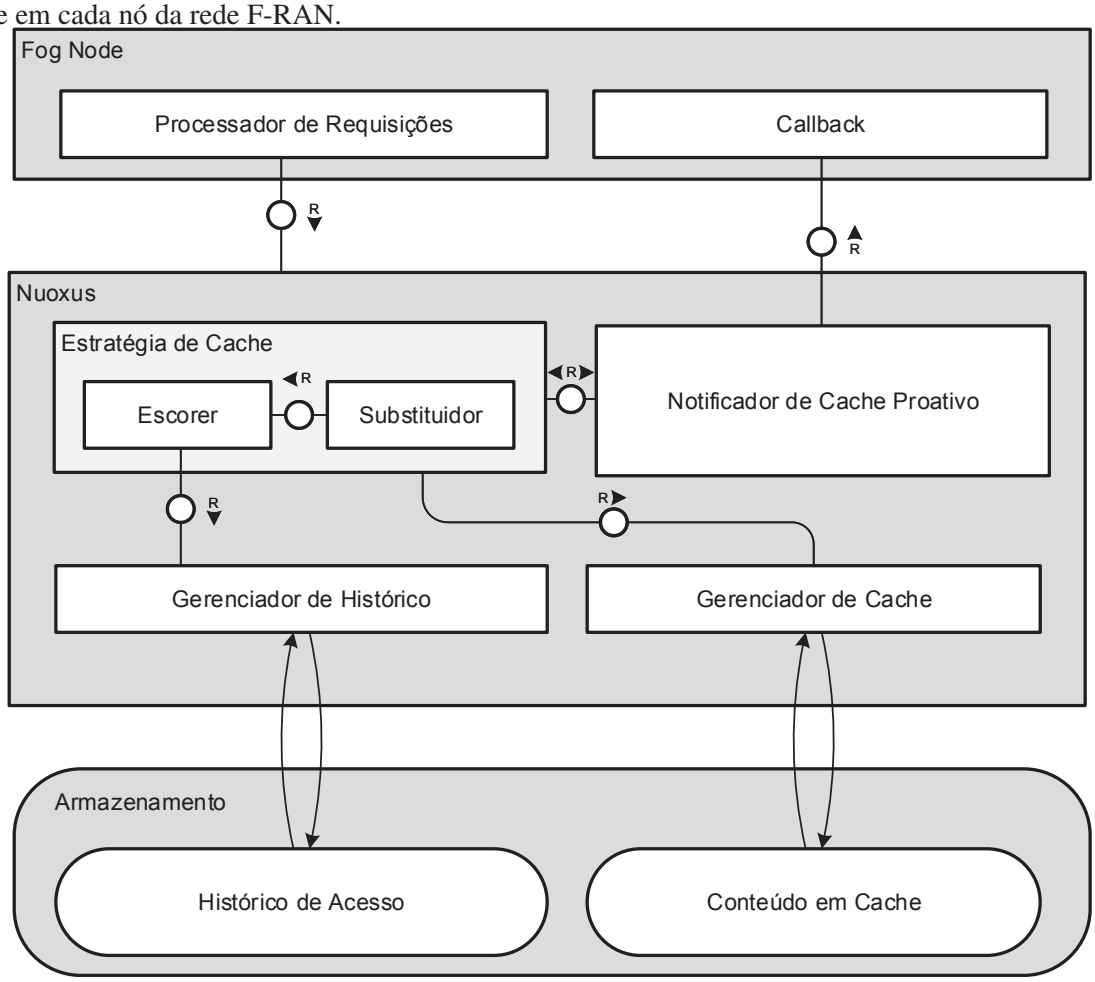
O Nuoxus também pode fornecer dados do histórico de requisições e do conteúdo atualmente armazenado em *cache* para que o algoritmo de seleção do modo de transmissão presente nos componentes da F-RAN utilize essas informações como parâmetro de decisão auxiliar. Explorar a utilização de tal informação por esse algoritmo que é executado pelas antenas de uma rede F-RAN se encontra fora do escopo do modelo Nuoxus e dessa dissertação.

A arquitetura do Nuoxus pode ser visualizada através da Figura 12. Ela está dividida em três camadas principais. A primeira delas é o ***Fog Node***, que representa a interface com o dispositivo da F-RAN que irá interagir com o Nuoxus. A segunda parte é o **Nuoxus**, que contém o núcleo de processamento do modelo. E por fim, o **Armazenamento** representa tanto o sistema de arquivos quanto a área de memória utilizada pelo modelo. Os componentes que fazem parte de cada uma das camadas estão listados a seguir, com uma breve descrição de sua responsabilidade.

- **Processador de Requisições**: Esse componente é responsável por chamar o Nuoxus quando o nó recebe uma requisição de um conteúdo multimídia. De modo geral, ele centraliza a comunicação entre a *F-RAN* e o Nuoxus.

- ***Callback***: Esse componente é responsável por receber chamadas de um nó pai durante o processamento de *caching* proativo. Caso um filho não tenha capacidade de armazena-

Figura 12: Componentes que formam a arquitetura do Nuoxus. Esse modelo roda de forma independente em cada nó da rede F-RAN.



Fonte: Elaborada pelo autor.

mento, ele simplesmente não irá registrar esse *callback* com o pai.

- **Estratégia de Cache**: Esse componente é responsável por determinar se um arquivo deve ou não ser armazenado em sua *cache*. Ele possui dois subcomponentes: o ***Escorer***, que dá uma pontuação (escore) para um arquivo e o **Substituidor**, que realiza a substituição dos arquivos em *cache* baseado em suas pontuações.

- **Notificador de Cache Proativo**: Utilizando-se do mecanismo de pontuação, esse componente determina se dado conteúdo é relevante para os nós filhos que possuem um *callback* registrado, e faz uma chamada aqueles que possam ter interesse de forma proativa

- **Gerenciador de Histórico**: Esse componente centraliza o armazenamento e obtenção do histórico de requisições dos nós filhos para o nó local. Ele possui a capacidade de fornecer tais informações para componentes externos ao Nuoxus.

- **Gerenciador de Cache**: Similar ao Gerenciador de Histórico, esse componente centraliza e atua como uma interface genérica para acesso aos arquivos de conteúdo multimídia

armazenados no nó local. Esse componente também possui capacidade de fornecer informações sobre o conteúdo armazenado na *cache* para entidades externas ao modelo.

4.3 Política de Substituição de Cache

Predizer a popularidade de um conteúdo online não é uma tarefa simples. Segundo Tatar et al. (2014), diversos estudos demonstram que o interesse dos usuários em um conteúdo online é transiente, heterogêneo e muitas vezes imprevisível. Considerando esse contexto, o Nuoxus não realiza a previsão da popularidade do conteúdo a fim de decidir se deve armazená-lo em *cache*. Ao invés disso, ele examina o histórico de requisições dos nós filhos para determinar a similaridade entre o nó requisitante e a rede. É esperado que sejam mais altas as chances de um arquivo ser solicitado novamente se o nó que o solicitou tenha um padrão de acesso mais similar com os demais nós da rede. Além disso, o Nuoxus também leva em consideração o número de acessos do conteúdo, dado o fenômeno de viralização dos conteúdos online (TATAR et al., 2014).

A política de substituição do Nuoxus pode ser dividida em duas partes distintas. A primeira parte, apresentada na Subseção 4.3.1 é responsável por atribuir um escore para o arquivo requisitado, utilizando como base os parâmetros descritos anteriormente. A segunda parte, apresentada na Subseção 4.3.2 determina se o arquivo requisitado deve ou não ser armazenado em *cache* com base i) no tamanho do arquivo ii) no espaço de armazenamento disponível iii) nos escores do arquivo requisitado e iv) nos escores dos arquivos atualmente armazenados.

4.3.1 Cálculo do Escore

O cálculo do escore de um arquivo é feito pelo **Escorer**, utilizando o Algoritmo 1. Ele é executado após o histórico de requisições já ter sido atualizado com a nova requisição para esse arquivo. Nesse algoritmo, um arquivo requisitado é identificado por $id\_arquivo\_requisitado$. Esse identificador pode ser obtido de diversas formas, uma delas é utilizando técnicas de identificação de conteúdo por prefixo (veja Subseção 2.2.2).

O escore do arquivo é obtido através da multiplicação de três fatores:

1. **C**: Esse fator é cálculo da similaridade de cosseno ($cosine$) entre os vetores binários $accessos\_1$ e $accessos\_2$. O vetor $accessos\_1$ representa o padrão de acesso do nó requisitante, e consiste de uma lista de mesmo tamanho do número de arquivos no histórico desse nó ($historico\_requisitante$). Todos os valores do $accessos\_1$ são 1. Já o vetor $accessos\_2$ é construído com base no histórico do nó executando o Nuoxus ($historico\_local$). Para cada arquivo no nó requisitante, o algoritmo atribui o valor 1 caso o arquivo já esteja no histórico local mais de uma vez (a fim de desconsiderar a requisição realizada originalmente por esse nó) e 0 caso o contrário. Isso significa que o cálculo de similaridade de cosseno é o mesmo abordado na Seção 2.3, desconsiderando a

Algoritmo 1: Atribuição do escore ao arquivo requisitado, criado para utilização no Nuoxus.

Entrada: $id\_arquivo\_requisitado, historico\_requisitante, historico\_local$
Saída: $escore$

```
1  início
       Dados: i, arquivo, accessos_1, acessos_2, R, C, A;
2      escore ← 0;
3      i ← 0;
4      para cada arquivo ∈ historico_requisitante faça
5          i ← i + 1;
6          accessos_1[i] ← 1;
7          se arquivo está mais de uma vez em historico_local então
8              acessos_2[i] ← 1;
9          senão
10             acessos_2[i] ← 0;
11         fim
12     fim
13     C ← cosine(acessos_1, acessos_2);
14     R ← tamanho(historico_requisitante) ÷ tamanho(historico_local);
15     A ← acessos(historico_local, id_arquivo_requisitado);
16     escore ← R × C × A;
17 fim
18 retorna escore;
```

grandeza escalar dos componentes;

2. **R**: Essa é a razão entre o número de arquivos distintos no histórico do nó requisitante ($tamanho(historico\_requisitante)$) e o número de arquivos distintos no histórico do nó atualmente executando o Nuoxus ($tamanho(historico\_local)$). Esse fator é utilizado para evitar que o algoritmo favoreça nós filhos que fazem requisições de poucos arquivos que, por coincidência, já tenham sido requisitados anteriormente;

3. **A**: Esse fator é o total número de acessos realizados ao arquivo requisitado. Esse fator é obtido através da função auxiliar $acessos$, que verifica quantas vezes o arquivo requisitado aparece no histórico de acessos do nó executando o Nuoxus ($historico\_local$). Esse fator é utilizado para que um arquivo popular, mesmo que sendo requisitado por usuários que não tenham um alto grau de similaridade com o restante da rede, ainda tenha chances de ser armazenado em *cache*

Em resumo, pode-se dizer que o principal fator utilizado no cálculo do escore é o total número de acessos ao arquivo (A). Porém, como ambos os fatores C e R sempre estarão no intervalo $[0, 1]$, eles são utilizados como variáveis reguladoras. O primeiro serve para que os arquivos requisitados por nós com maior similaridade obtenham um maior escore, e o segundo para dar um maior escore para os arquivos requisitados por nós com um maior histórico de atividades na rede.

Para exemplificar o efeito do cálculo, considere uma rede formada por um nó pai conectado a cinco nós filhos. Considere também o histórico de requisições como apresentado na Tabela 2. Essa tabela lista 10 arquivos e o número de vezes que cada um deles foi requisitado por cada nó filho, representando o $historico\_requisitante$. A coluna total contém a soma do número de requisições feitas pelos filhos, resultando no histórico de requisições do nó executando o Nuoxus, o qual representa o $historico\_local$.

Tabela 2: Exemplo: histórico de acesso de uma rede formada por um nó pai conectado a cinco nós filhos

	Nó 1	**Nó 2**	**Nó 3**	**Nó 4**	**Nó 5**	**Total**
Arquivo 1	0	0	0	0	1	1
Arquivo 2	1	0	1	0	1	3
Arquivo 3	1	0	0	0	0	1
Arquivo 4	2	0	1	0	3	6
Arquivo 5	0	1	0	0	3	4
Arquivo 6	0	0	1	0	0	1
Arquivo 7	0	1	0	0	0	1
Arquivo 8	0	1	0	0	0	1
Arquivo 9	0	0	1	0	0	1
Arquivo 10	0	0	1	0	0	1

Fonte: Elaborada pelo autor.

Ao simular a requisição do arquivo de número 6 por cada um dos nós filhos, aplicando o Algoritmo 1 em etapas, obtemos os resultados apresentados na Tabela 3. Nesse exemplo, é possível observar que tanto a similaridade de cosseno entre o nó filho e a rede (nó pai), quanto o número de arquivos requisitados afetam o escore dado pelo Nuoxus.

Tabela 3: Exemplo: cálculo de escore realizado pela política de substituição do Nuoxus. Foram omitidos os parâmetros $tamanho(historico\_local)$ e A, pois estes são 10 e 2 para todos os nós, respectivamente. T é uma abreviação para $tamanho(historico\_requisitante)$.

	$acessos\_1$	$accessos\_2$	$cousine(C)$	T	R	$Escore$
Nó 1	[1, 1, 1, 1]	[1, 0, 1, 1]	0,86602	4	0,4	0,692816
Nó 2	[1, 1, 1, 1]	[1, 1, 0, 0]	0,70710	4	0,4	0,56568
Nó 3	[1, 1, 1, 1, 1, 1]	[1, 1, 1, 0, 0, 0]	0,70710	5	0,5	0,70710
Nó 4	[1]	[1]	1,00000	1	0,1	0,2
Nó 5	[1, 1, 1, 1, 1]	[0, 1, 1, 1, 1]	0,89442	5	0,5	0,89442

Fonte: Elaborada pelo autor.

Por exemplo, ao requisitar o arquivo 6, que já havia sido requisitado pelo nó 3, o cosseno de similaridade entre o nó 4 e a rede é 1, pois esse único é arquivo requisitado pelo nó 4 até então. O que impede de que esse seja o escore mais alto, é razão R, a qual é baixa devido ao fato do nó ter requisitado apenas 1 arquivo até então. O mesmo é possível observar quando comparando os resultados dos nós 2 e 3, pois metade dos arquivos requisitados por ambos os nós já foram requisitados antes, portanto, ambos apresentam o mesmo cosseno de similaridade (0,70710) e

o fator decisivo é a razão R. Por ter requisitado mais arquivos, o nó 3 acaba obtendo um maior escore.

Em uma implementação real, alguns cuidados extras devem ser tomados. Por exemplo, devido ao grande número de requisições, os valores da similaridade de cosseno e da razão R podem ser de ordens muito baixas, o que pode causar erros dependendo do formato de dados em que esses valores são armazenados. Uma forma simples de resolver isso, é estabelecer um multiplicador que irá aumentar a ordem desses valores. Um outro problema que surge devido ao número alto de requisições, é o tempo de cálculo do Escore que pode se tornar alto, devido ao número de arquivos nos históricos. Para mitigar esse problema, pode-se implementar um ***Cleaner***, cuja responsabilidade principal seria limpar tais dados após um intervalo de tempo parametrizável. Isso também impediria que o modelo ficasse viciado com um comportamento antigo.

4.3.2 Substituição de Arquivos

A segunda parte da política de substituição de cache, consiste em efetivamente decidir quando um arquivo vai ou não ser armazenado em *cache*, devido a limitações no espaço de armazenamento, especialmente se o hardware das novas arquiteturas de rede armazenarem tal conteúdo em memória volátil, devido à sua maior velocidade de acesso.

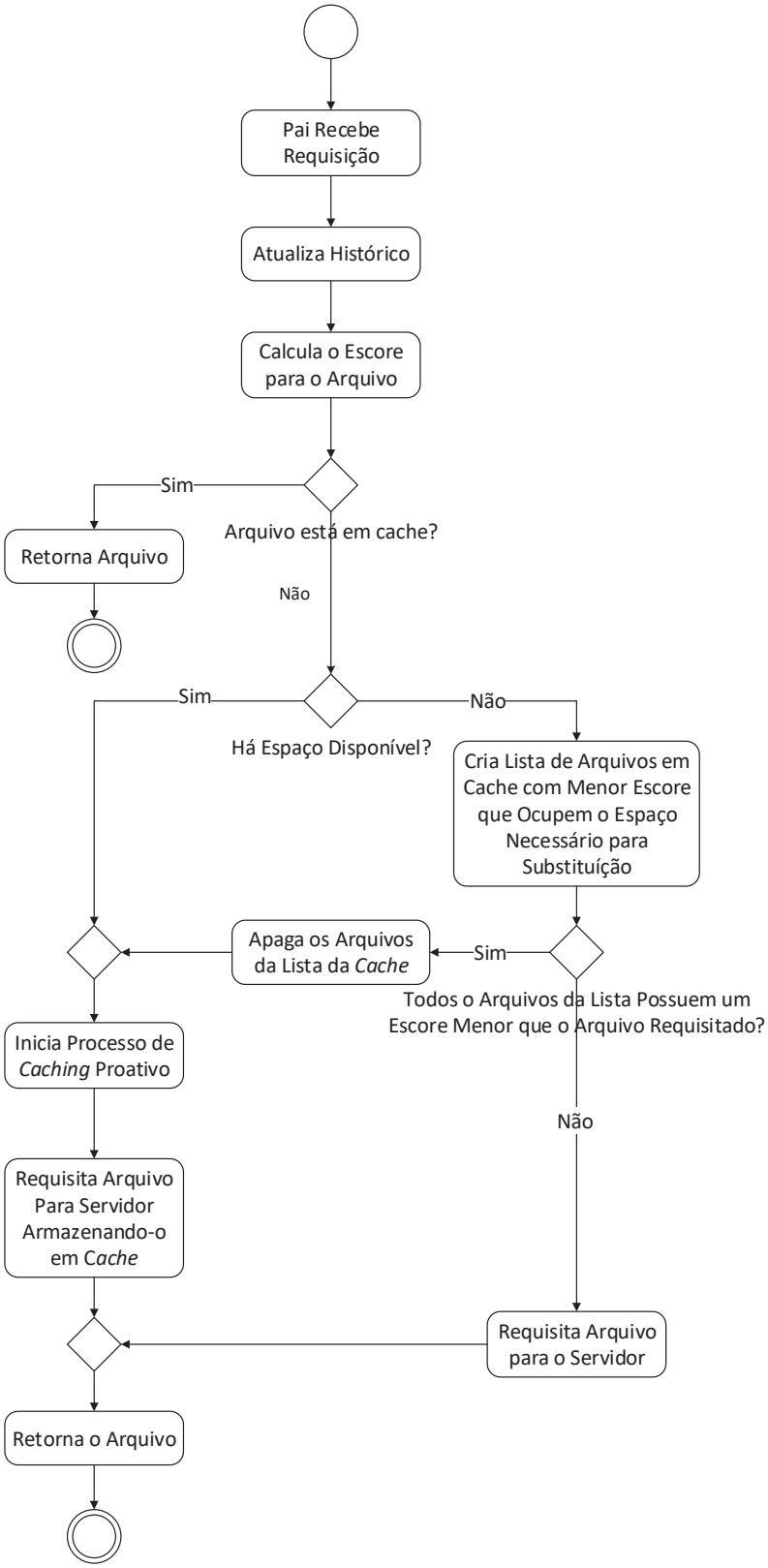
A lógica que decide se um arquivo vai ser colocado em *cache* é apresentada pelo diagrama de atividades da Figura 13. Com a finalidade de simplificar o diagrama, foram omitidos os componentes do Nuoxus envolvidos em cada uma das etapas de decisão. Porém, eles serão explorados durante o restante dessa subseção.

Inicialmente, o Nuoxus recebe uma requisição de um nó filho através do **Processador de Requisições**. Após isso, a primeira ação do Nuoxus é registrar essa requisição no histórico, tanto do nó executando o Nuoxus quanto do nó filho, utilizando o **Gerenciador de Histórico** e subsequentemente, fazer o cálculo do escore desse arquivo através do ***Escorer***. Caso esse arquivo já tenha sido requisitado anteriormente, o Nuoxus já possui um escore registrado para ele. Nesse caso, o escore mais alto será mantido no sistema.

A próxima etapa consiste em verificar se o arquivo requisitado está armazenado em *cache*. Para isso, o componente **Gerenciador de *Cache*** é utilizado. Caso o arquivo já esteja armazenado, ele simplesmente é retornado para o nó filho, sem que haja a necessidade de requisitá-lo para a camada superior, podendo apenas notificá-la de tal requisição para questões de métricas e similares.

Contudo, caso o arquivo requisitado não esteja em cache, o **Substituidor** irá requisitar para o **Gerenciador de *Cache*** uma lista de arquivos que precisariam ser removidos da *cache* para que seja possível armazenar o novo arquivo, dado o seu tamanho. A substituição irá ocorrer caso a lista esteja vazia (o que significa que já há espaço suficiente em disco), ou caso todos os arquivos da lista possuam um escore menor do que o novo arquivo requisitado. Caso ocorra

Figura 13: Diagrama de atividades que representa a lógica utilizada pelo Nuoxus para decidir se um arquivo requisitado será armazenado em *cache*.



Fonte: Elaborada pelo autor.

a substituição, o Nuoxus reserva o espaço para o arquivo e dá início ao processo de *caching* proativo, que será explicado na Seção 4.4.

Independentemente se houver substituição ou não, como o arquivo não está em *cache*, ele é requisitado para a camada superior. A única diferença, em caso do arquivo dever ser armazenado em *cache*, o conteúdo recebido da camada superior é salvo e enviado para o nó filho. Caso contrário, o nó funciona como uma relé, apenas direcionando os dados para o nó filho.

4.4 Caching Proativo

O Nuoxus prevê a utilização de *cache* proativo pela F-RAN, com a finalidade de redução do tráfego no *fronthaul* em momentos de utilização intensa da rede, o que, consequentemente, traz uma redução da latência de transmissão. Como o nó executando o Nuoxus não possui conhecimento sobre os demais nós na rede, inclusive se tais nós possuem ou não a capacidade de fazer *caching*, ele nunca obriga que um outro nó faça o *caching* de conteúdo.

O processo completo de *caching* proativo ocorre em três momentos distintos. O primeiro deles ocorre nos nós filhos, pois estes devem se registrar com o nó pai, informando que possuem interesse e capacidade de participar desse processo.

O segundo momento ocorre quando o nó pai recebe uma requisição de um nó filho. Ao verificar que o arquivo requisitado deve ser armazenado em seu *cache*, o nó pai dispara um gatilho que inicia o processo de calcular o escore do arquivo requisitado para cada um dos nós filhos registrados. O cálculo do escore ocorre da mesma forma como descrito na Subseção 4.3.1, porém, ao invés de utilizar a similaridade entre as requisições do nó pai com o nó requisitante, é utilizada a similaridade entre o nó requisitante e o nó registrado.

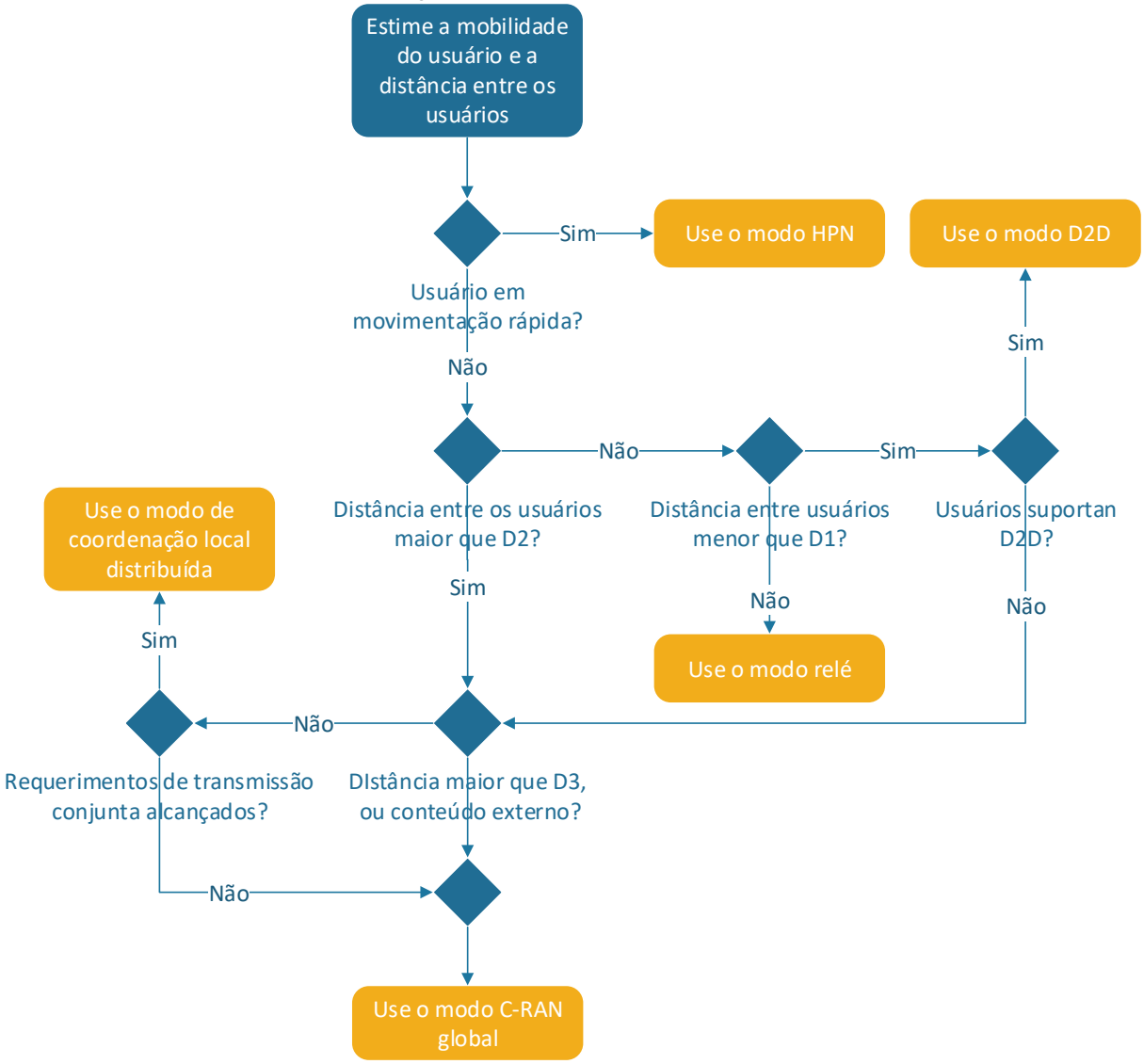
No terceiro e último momento, o nó pai envia através do **Notificador de Cache Proativo** uma mensagem para cada filho registrado, informando o escore e o tamanho do arquivo candidato. Essa mensagem é recebida através do componente batizado de ***Callback*** pelos nós filhos, que decidem ou não se o arquivo é interessante para sua *cache*. Para isso, os filhos utilizam a mesma lógica de substituição aplicada pelo Nuoxus, com a diferença que o *Escorer* é substituído por um componente que retorna o escore calculado pelo nó pai, ao invés de calculá-lo. Portanto, caso o arquivo tenha um escore maior do que todos os arquivos que precisariam ser substituídos para ceder espaço de armazenamento, o nó filho faz uma requisição do arquivo para o nó pai, que já possui o arquivo em sua *cache*, e então irá enviá-lo, sem precisar trafegar mais dados pelo *fronthaul*.

Uma abordagem que pode ser explorada nesse âmbito é parametrizar a razão entre o espaço reservado para *caching* regular e *caching* proativo, e possivelmente a utilização de diferentes políticas de substituição para cada um deles.

4.5 Algoritmo de Seleção do Método de Transmissão de Dados

Um dos artifícios trazidos pelas F-RANs a fim de diminuir o acesso ao *fronthaul* é a utilização de diferentes métodos de seleção para transmissão de dados, como descrito na Seção 2.1.4. O método para seleção do modo de transmissão apresentado por Peng et al. (2016) é representado pela Figura 14. Nesse método, $D1$, $D2$ e $D3$ são valores de distância estabelecidos pelo operador de rede, visto que os autores consideram que a distância ainda será o maior fator para estabelecer a qualidade da comunicação sem fio.

Figura 14: Seleção do modo de transmissão de uma F-RAN.

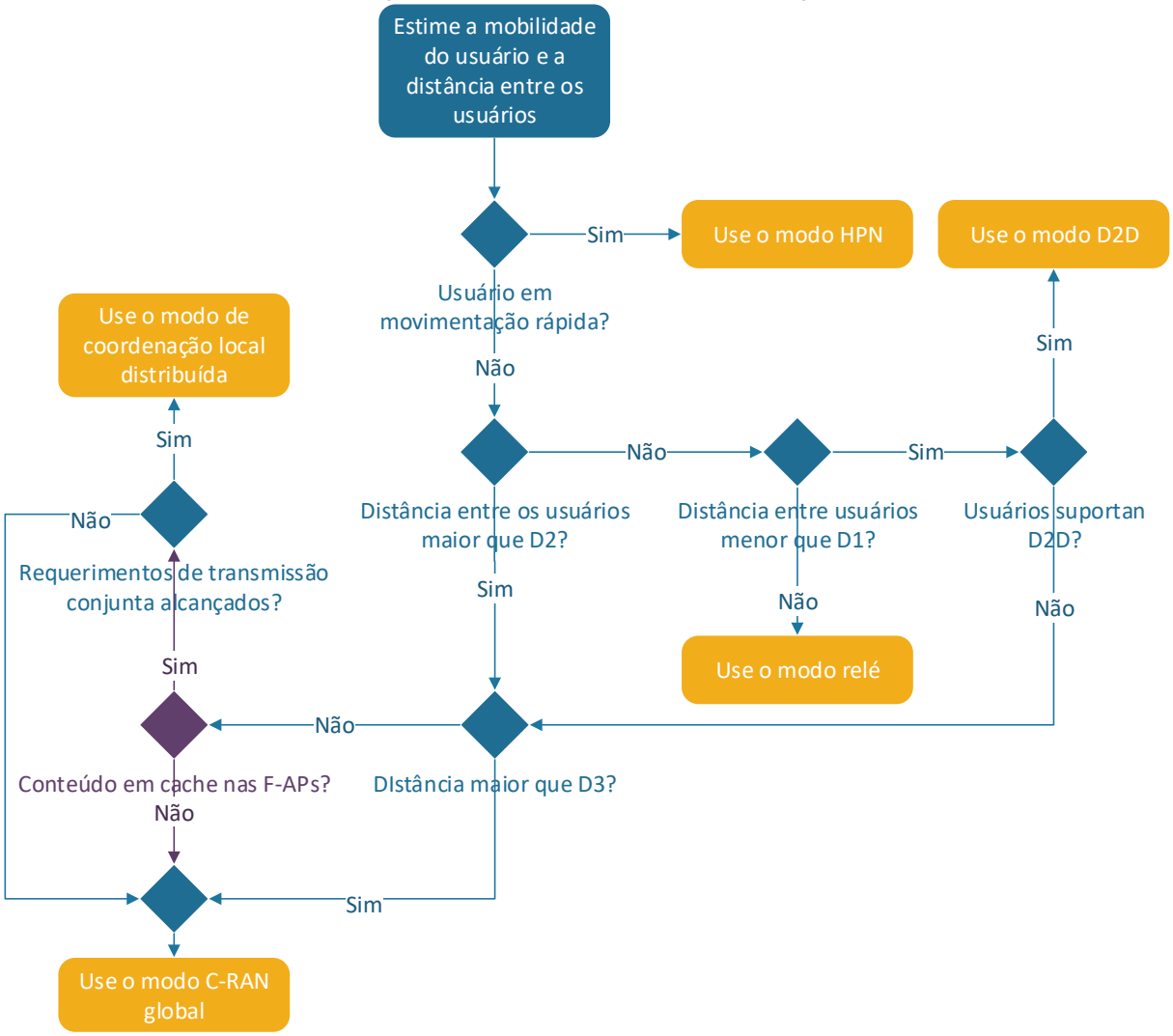


Fonte: Adaptada de (PENG et al., 2016).

A proposta do Nuoxus é adicionar o conteúdo armazenado em *cache* como mais um parâmetro a ser considerado pela rede ao fazer a escolha do modo de transmissão. A estratégia geral é pouco alterada, sendo que a principal diferença está na adição de mais um nó de decisão que seleciona entre o modo C-RAN global e o modo de coordenação local distribuída, como de-

monstrado na Figura 15. No método proposto por Peng et al. (2016), o modo de C-RAN global é sempre utilizado quando o conteúdo requisitado vem de fora da F-RAN. Porém, com a adição da capacidade de *cache* nos F-APs, é possível que múltiplos F-APs possuam o mesmo conteúdo em sua *cache*, o que possibilita a transmissão conjunta do mesmo, aumentando a velocidade de transmissão de conteúdo e a vazão da rede.

Figura 15: Seleção do modo de transmissão com adição do Nuoxus.



Fonte: Adaptada de (PENG et al., 2016).

Essa adição ao método de seleção do modo de transmissão pode ser vista como um bônus, consequente da capacidade do modelo de fornecer informações sobre os arquivos armazenados em *cache*. Porém, devido à impossibilidade de implementar uma F-RAN para comprovar sua eficiência, essa adição encontra-se fora do escopo desse trabalho, ficando citada aqui para efeito de documentação.

5 METODOLOGIA DE AVALIAÇÃO

Visto que a arquitetura de F-RANs foi proposta recentemente (PENG et al., 2016), ainda não há ferramentais disponíveis para realização de simulações para esse tipo de rede. Portanto, para validação do modelo, foi desenvolvido um protótipo na camada de aplicação do simulador ns-3[1]. Esse simulador provém modelos de componentes de redes que consideram suas características reais e que foram validados por diversos pesquisadores (PIRO; BALDO; MIOZZO, 2011). As próximas seções apresentam detalhes sobre a implementação do protótipo assim como a topologia de rede e os cenários de simulação utilizados para a validação do modelo. Por fim, são apresentadas as métricas utilizadas para avaliação do modelo e seu impacto em uma F-RAN.

5.1 Protótipo

A fim de integrar o Nuoxus com o ns-3, permitindo simular a sua aplicação em uma rede, um protótipo do modelo foi desenvolvido visando executá-lo na camada de aplicação do simulador. Com isso, foi possível injetar o novo comportamento na rede sem ter que alterar nenhum componente do núcleo do ns-3, garantindo a validação técnica já realizada por diversos membros da comunidade científica (ROUIL et al., 2017).

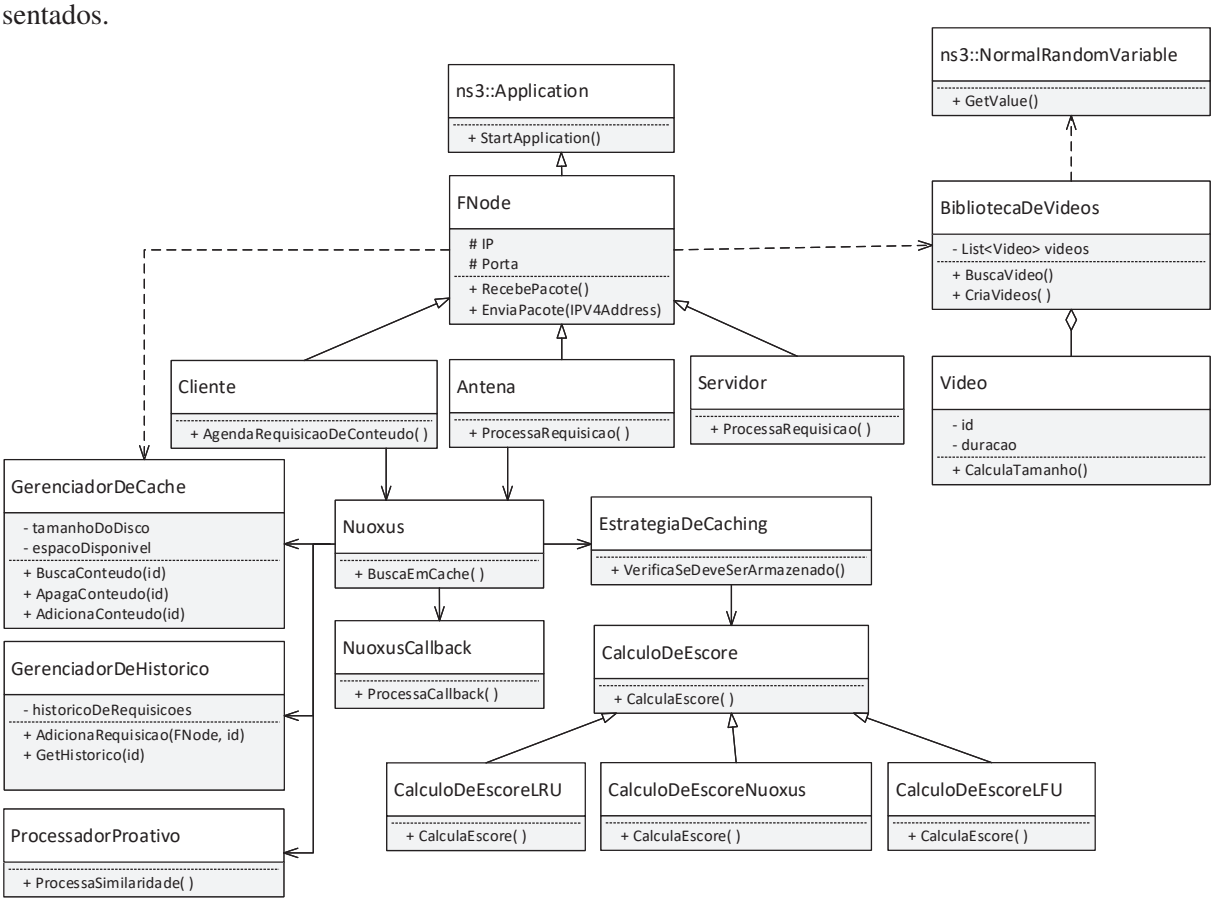
O protótipo, assim como o ns-3, foi implementado utilizando a linguagem de programação C++ e seu desenvolvimento focou-se em deixá-lo o mais fiel possível com aquilo que é esperado de uma implementação real do modelo. As únicas diferenças se dão em relação aos detalhes necessários para realizar sua integração com o *framework* do ns-3. A Figura 16 apresenta o diagrama de classes do protótipo, onde todas as classes e somente seus principais membros (atributos e métodos) estão sendo representados com a finalidade de tornar o diagrama mais claro.

As principais classes são a **Cliente**, **Antena** e **Servidor**, que representam os dispositivos F-UE, F-AP e *BBU Pool*, respectivamente. Todas essas classes herdam da **FNode**, permitindo que esses dispositivos sejam identificados na rede através de seu endereço TCP/IP e contenham funcionalidades para receber e enviar dados através da rede. Como a classe **FNode** herda da classe ***Application*** do ns-3, isso faz com que todas essas classes que representam dispositivos possam ser executadas como elas fossem aplicações nos nós do ns-3.

Por se tratarem de aplicações executando em nós, essas classes que representam dispositivos comunicam através da *API* de *sockets* fornecida pelo ns-3, que provém implementações para os protocolos TCP e UDP. O protótipo utilizou comunicação TCP, para evitar perda de pacotes. Como qualquer outra aplicação que utiliza *sockets*, não é necessário que as aplicações conheçam detalhes sobre a implementação dos *sockets*, mas é importante ressaltar que o ns-3 utiliza pacotes virtuais, ou seja, pacotes sem dados (*payload*) reais, a fim de não consumir muitos re-

[1] http://www.nsnam.org

Figura 16: Diagrama de classe do protótipo. Somente os membros mais relevantes estão sendo representados.



Fonte: Elaborada pelo autor.

cursos para simulação. As aplicações utilizam-se do cabeçalho dos pacotes para efetivamente trocarem informações.

Todo o **FNode** possui um *cache*, que é gerenciado através da classe **GerenciadorDeCache** e um histórico de requisições, que é gerenciado pela classe **GereciadorDeHistorico**. Os FNodes também possuem acesso à biblioteca de vídeos (**BibliotecaDeVideos**), que implementa o padrão de projetos *Singleton*, garantido que a mesma instância dessa classe, e portanto, a mesma coleção de vídeos seja usada durante toda a simulação. A biblioteca de vídeos inicializa todos os dados relacionados à sua coleção no primeiro acesso a essa instância única. Para isso, ela utiliza o *framework* de variáveis aleatórias do ns-3 para determinar a duração de cada vídeo. A utilização do *framework* de variáveis aleatórias do ns-3 garante que uma semente única, que é fornecida pelo *script* de simulação, será utilizada para geração de todos os números pseudo-aleatórios, garantido a reprodutibilidade dos resultados.

O modelo em si é implementado através da classe **Nuoxus**. Essa classe é utilizada pelas F-APS e pelos F-UEs, sendo que o segundo somente para o *caching* proativo. Cada instância do Nuoxus possui acesso à classe *EstrategiaDeCaching*, que é responsável por determinar se um arquivo requisitado deve ou não ser armazenado em cache baseado no escore retornado pela classe **CalculoDeEscore**. Para isso, ela utiliza uma das seguintes especializações dessa classe:

- **CalculoDeEscoreLRU**: Nessa implementação, cada chamada ao método **CalculaEscore** irá retornar um valor maior. Garantido, dessa forma, o arquivo mais recente tenha um maior escore.

- **CalculoDeEscoreLFU**: Nessa implementação, o método **CalculaEscore** irá retornar o número de vezes que o arquivo foi requisitado, implementando dessa forma o LFU.

- **CalculoDeEscoreNuoxus**: Essa é a implementação padrão do Nuoxus que utiliza a metodologia de pontuação descrita na Seção 4.3.

Caso o **script** de testes utilize **caching** proativo, as instâncias do Nuoxus possuem acesso às classes **ProcessadorProativo** e **NuoxusCallback**. A primeira é responsável por, na F-AP, executar o processo de *caching* proativo, conforme descrito na Seção 4.4 e a segunda por receber essas informações e notificar a instância do Nuoxus para fazer a requisição, caso o arquivo deva ser armazenado na *cache* do F-UE. Para os cenários onde não há *caching*, todas as instâncias do Nuoxus são desabilitadas.

Finalmente, as classes do protótipo são utilizadas para montar o cenário de testes através do que é chamado no ns-3 de *script* de simulação. Um *script* se trata de um programa que possui acesso a todos os modelos disponíveis no ns-3, inclusive as aplicações - e portanto o protótipo do Nuoxus. Nesse *script*, toda a topologia de rede é inicializada. Isso inclui criar os nós, instalar as interfaces e componentes de cada camada de comunicação, atribuir endereços e instalar as aplicações. Por fim, o *script* também recebe parâmetros e configura os componentes varáveis através do *framework* de atributos do ns-3. Esses parâmetros incluem a semente que será utilizada para geração dos números aleatórios, a estratégia de *caching*, o tipo de distribuição de popularidade que será utilizada para realizar as requisições de vídeo, etc. Mais detalhes sobre a topologia de rede e dos parâmetros utilizados em cada cenário serão apresentados na próxima seção.;

5.2 Topologia de Rede e Cenários de Simulação

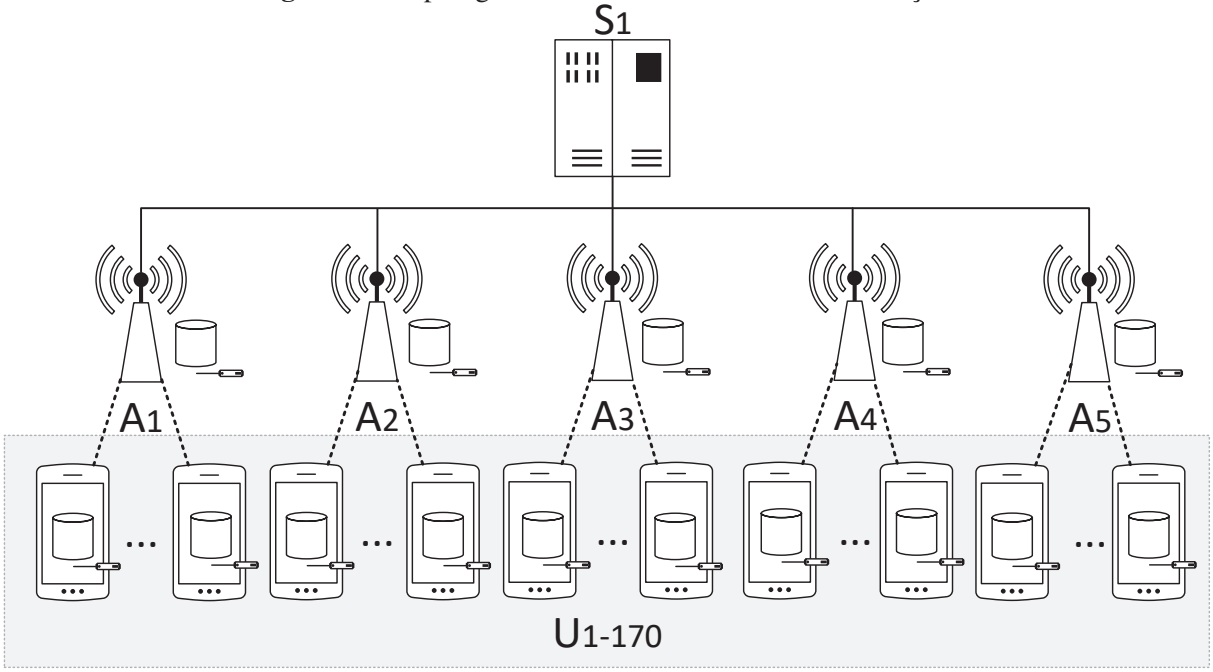
Para realizar a validação do modelo, utilizou-se durante as simulações a mesma topologia de rede entre todos os cenários de teste. Essa topologia foi baseada na arquitetura padrão de uma F-RAN, com exceção da ausência de um nó de alta potência. Esse tipo de nó não foi incluído na topologia pois eles são heranças das H-CRAN e são utilizados somente quando a cobertura de sinal não é suficiente (PENG et al., 2016), o qual caracteriza um cenário fora do escopo desse trabalho. Essa topologia é formada por 1 dispositivo agindo como o servidor de vídeo (S_1) (que também pode ser visto como o núcleo de rede (EPC ou *content provider*)), 5 dispositivos agindo como F-APs (A_{1-5}) e 170 dispositivos agindo como F-UEs (U_{1-170}).

Considerando tais dispositivos, sua organização na rede se dá da seguinte forma: cada F-AP é conectada ao núcleo de rede através de uma conexão compartilhada com largura de banda de $100Gb/s$ atraso físico de $5ms$. Os 170 F-UEs foram conectados às F-APs de maneira aleatória,

utilizando uma distribuição uniforme, através de uma conexão com largura de banda de $1Gb/s$ e atraso físico de $10ms$. Esses valores foram determinados conforme capacidades de um canal de fibra ótica (ALFIAD et al., 2008) e dos valores médios de uma rede LTE (3GPP, 2015). Além disso, cada dispositivo possui $40Gb$ de espaço em disco para realizar *caching* de conteúdo. O tamanho do disco foi definido de forma arbitrária, sendo que média cerca de 30 arquivos ficam armazenados na *cache* por vez. A Figura 17 contém uma representação da topologia de rede utilizada.

A rede utilizada para a simulação faz uso de conexões cabeadas ao invés de conexões sem fio, pois o módulo de LTE do ns-3 realiza diversas simulações de sinal, as quais consumem recurso computacional. Além disso, devido à natureza dessa validação, a tecnologia de comunicação acaba não influenciando no resultado das comparações, desde que todos os cenários utilizem a mesma topologia de rede.

Figura 17: Topologia de rede utilizada durante a simulação.



Fonte: Elaborada pelo autor.

Para as requisições de vídeo, os dispositivos possuem acesso a uma biblioteca composta de 10 mil vídeos. Como descrito anteriormente, essa biblioteca é fixa, ou seja, ao inicializar a simulação, todo os modelos do ns-3 possuem o conhecimento prévio sobre os arquivos disponíveis. A duração de cada vídeo, em minutos, se deu através de uma distribuição normal com parâmetros de média 4.4 e variância 3, baseado nos dados publicados por comScore (2014), sendo que o tempo mínimo de cada vídeo será de um minuto. O tamanho do arquivo é dado pela multiplicação do *bitrate* padrão de um vídeo HD usando o *codec* H.264, que é 5000 kb/s (ENCODING, 2016), pela duração do vídeo, em segundos.

Todos os cenários simulados trabalham da mesma forma. No primeiro momento, a rede é configurada na topologia descrita anteriormente e a estratégia de *caching* a ser utilizada é

instanciada e passada para o Nuoxus, que atua em cada um dos dispositivos, exceto pelo servidor de vídeo, cuja única finalidade é fornecer os pacotes dos vídeos requisitados. Após esse primeiro momento, para cada F-UE, é agendado um evento de requisição de vídeo para ser executado no instante 0.01 da simulação. A decisão de qual vídeo será requisitado segue o que foi chamado aqui de distribuição de popularidade. Essa distribuição depende de qual cenário está sendo simulado, podendo ser:

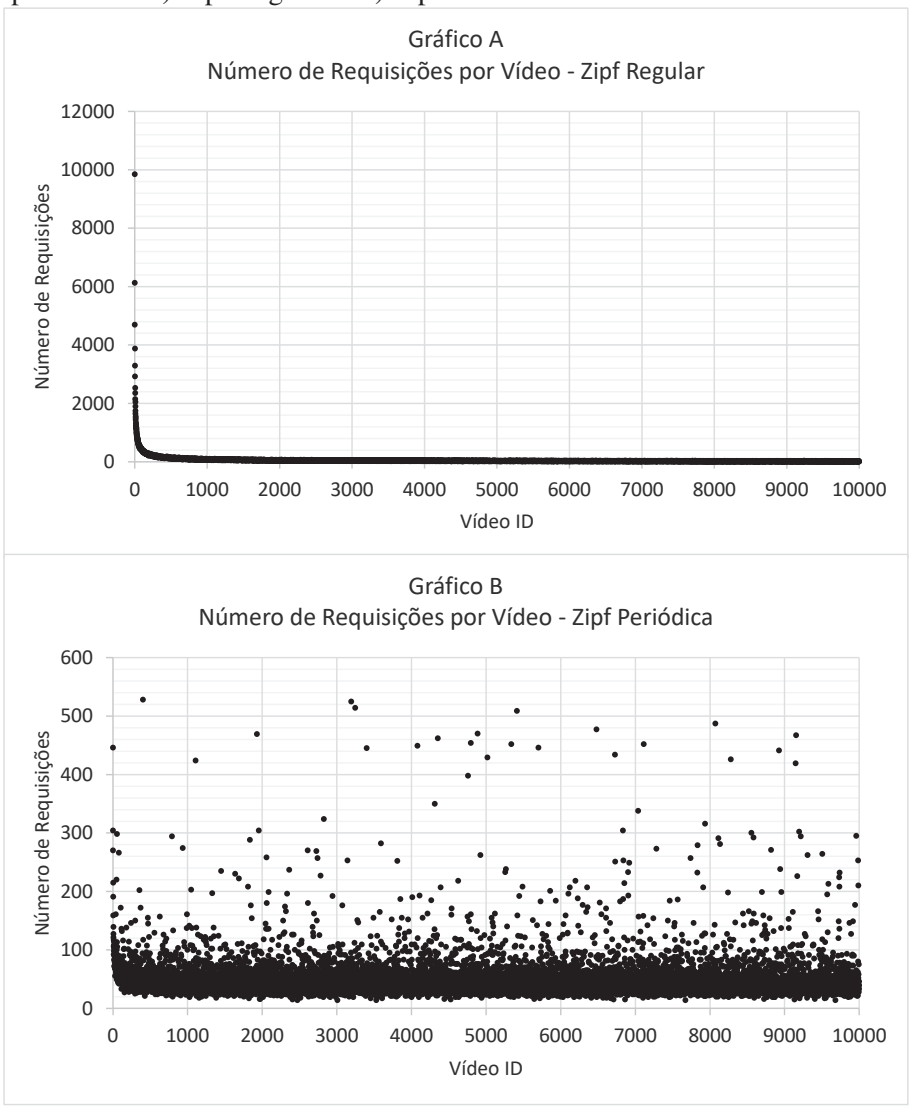
- **Zipf Regular**: nessa distribuição, os F-UEs escolhem o arquivo a ser requisitado utilizando a distribuição de probabilidade Zipf com $obliquidade = 0,7$. Essa distribuição de e o seu parâmetro foram escolhidos com base em dois estudos. No primeiro, publicado por Gill et al. (2007) os autores realizaram medições do tráfego local para o Youtube e acharam a que a requisição de conteúdo poderia ser modelada utilizando a distribuição Zipf com $obliquidade = 0,56$. Já o segundo, utilizando os dados de tráfego de um *Web Proxy*, sugere que o acesso a conteúdo web segue a distribuição Zipf com $obliquidade = 0,56$ (MAHANTI; WILLIAMSON; EAGER, 2000). Portanto, a $obliquidade = 0,7$ escolhida para a validação do modelo é a média aritmética dos valores obtidos por esses estudos. Além disso, outros trabalhos também indicam a distribuição Zipf como sendo a mais fiel para representar esse tipo de informação ((BRESLAU et al., 1999), (TATAR et al., 2014)).

- **Zipf Periódica**: essa distribuição também utiliza Zipf com os mesmos parâmetros descritos para **Zipf Regular**. A diferença é que passado um certo período de tempo, que para esses testes foi de uma hora do tempo simulado, é realizado um deslocamento do arquivo mais requisitado pela rede. A intenção é simular canais de mídia que lançam conteúdos periodicamente gerando assim interesse em usuários que recebem notificações. Com isso, podemos comparar como diferentes estratégias de *caching* se adaptam ao haver uma alteração na distribuição de popularidade dos arquivos e mais incerteza em termos de requisição de conteúdo.

O intuito de utilizar duas distribuições é verificar o comportamento das diferentes estratégias de *caching* em duas situações distintas, sendo uma mais regular e outra mais imprevisível. Com a finalidade de visualizar o impacto que cada distribuição de popularidade tem sobre as requisições feitas, a Figura 18 apresenta os gráficos de 500 mil utilizações dessas duas diferentes distribuições de popularidade. O gráfico A representa a utilização da distribuição Zipf Regular e o gráfico B da distribuição Zipf Periódica. O eixo horizontal do gráfico contém o ID de cada vídeo, que se encontra entre 1 e 10 mil, enquanto que o eixo vertical apresenta o número de requisições realizadas àquele arquivo.

Através dos gráficos é possível ver claramente o efeito que cada uma das distribuições tem sobre os arquivos requisitados. Na distribuição Zipf Regular, representada no gráfico A, o arquivo de ID 1 foi requisitado cerca de 10 mil vezes, o arquivo de ID 2 cerca de 6 mil vezes, o de ID 3 cerca de 5 mil vezes e assim sucessivamente. Já na distribuição Zipf Periódica,

Figura 18: Gráficos contendo o número de requisições realizadas para cada vídeo utilizando as distribuições de popularidade A) Zipf Regular e B) Zipf Periódica.



Fonte: Elaborada pelo autor.

representada através do gráfico B, é possível perceber que as requisições são mais diversificadas, mas ainda há um número limitado de arquivos com maior popularidade.

Após cada *F-UE* selecionar um vídeo utilizado uma das distribuições, eles fazem as requisições para a F-AP na qual eles estão conectados. A F-AP irá então processar através no Nuoxus a requisição e repassá-la para o servidor de vídeo que irá então fazer o caminho oposto com os pacotes que compõem o arquivo requisitado. Cada F-UE fará uma nova requisição ao se passar metade do tempo de duração do vídeo requisitado anteriormente. Como cada vídeo possui uma duração diferente, depois da primeira requisição os F-UEs então passam a requisitar arquivos em instantes diferentes. A simulação é interrompida ao passar 24 horas simuladas.

Para cada cenário de teste apresentado na Tabela 4, a simulação foi executada 10 vezes. Segundo a documentação do ns-3 ((NS-3 Project, 2017)), não há garantia que sementes de geração de número aleatórios não produzam *stream* de números que se sobreponham. Por esse

motivo durante todas as execuções de simulações a semente se manteve fixa o estado da *stream* de geração foi incrementado a cada nova execução de um mesmo cenário. Por exemplo, o cenário C_1 foi executado 10 vezes, utilizando a semente 1 e os estados de *stream* de 1 à 10. Similarmente, o C_2 também foi executado 10 vezes utilizando a semente 1 e os estados de *stream* de 1 à 10.

Tabela 4: Cenários de simulação utilizados para validação do modelo Nuoxus.

Nome	Distribuição de Popularidade	Estratégia de *Caching*
$\mathbf{C}_1$	Zipf Regular	Sem Cache
$\mathbf{C}_2$	Zipf Regular	LFU
$\mathbf{C}_3$	Zipf Regular	LRU
$\mathbf{C}_4$	Zipf Regular	Nuoxus Sem Proatividade
$\mathbf{C}_5$	Zipf Regular	Nuoxus Com Proatividade
$\mathbf{C}_6$	Zipf Periódico	Sem Cache
$\mathbf{C}_7$	Zipf Periódico	LFU
$\mathbf{C}_8$	Zipf Periódico	LRU
$\mathbf{C}_9$	Zipf Periódico	Nuoxus Sem Proatividade
$\mathbf{C}_{10}$	Zipf Periódico	Nuoxus Com Proatividade

Fonte: Elaborada pelo autor.

5.2.1 Métricas de Avaliação

O ns-3 trabalha a nível de pacotes, provento mecanismos para acompanhá-los nos seus trajetos durante a simulação. Para validação do modelo, foram definidas métricas para avaliação da efetividade da solução. No total foram definidas quatro métricas principais, as quais podem ser classificadas em duas categorias: eficiência da política de substituição e latência de rede, apresentadas a seguir.

5.2.1.0.1 Eficiência da Política de Substituição

A política de substituição pode ser vista como o núcleo do Nuoxus. Portanto, medir o quão efetiva ela se demonstra, é um fator muito importante a ser considerado na avaliação do modelo. Para tal, foram estipuladas duas métricas: o percentual de requisições cujo arquivo requisitado estava armazenado em *cache* (C), também conhecido como percentual de *cache hit*, e a segunda é o percentual de requisições que geraram substituições de arquivo em *cache* (S).

A primeira visa demonstrar o quão efetivo de fato é a política de substituição, pois quanto maior ela for, mais vezes a F-RAN está deixando de acessar o núcleo de rede através do *fronthaul* pois está acessando o seu *cache* local. Ela é obtida através do seguinte cálculo:

$$C = \frac{\text{Cache Hit F-AP} + \text{Cache Hit F-UE}}{\text{Total Requisições}} \tag{5.1}$$

Onde *Cache Hit F-AP* representa o número de arquivos requisitados que estava armazenado na *cache* da F-AP que tratou a requisição e *Cache Hit F-UE* representa o número de arquivos requisitados que já foram armazenados na F-UE através do *caching* proativo. Por fim, *Total Requisições* é auto-descritivo, representando o número total de vezes que os F-UEs fizeram alguma requisição de vídeo.

Já a segunda métrica demonstra o gasto da rede em substituição de conteúdo. Idealmente, com uma capacidade de *cache* e com uma taxa de transferência infinita, a política de substituição nunca iria substituir o conteúdo armazenado. Porém, como esses fatores são limitados, uma boa política iria armazenar apenas os conteúdos mais relevantes, evitando a sua rotatividade. Sua obtenção se dá através da seguinte fórmula:

$$S = \frac{\text{Substituições em } \textit{Cache}}{\text{Total Requisições}} \tag{5.2}$$

Onde *Substituições em Cache* representa o número de vezes que um arquivo foi removido da *cache* de um dispositivo para disponibilizar espaço para um outro arquivo, e *Total Requisições* mais uma vez representa o número total de vezes que os *F-UEs* fizeram alguma requisição.

5.2.1.0.2 Latência de Rede

A grande vantagem que as F-RANs possuem sobres as arquiteturas propostas anteriormente, como H-CRANs, é a possibilidade dos componentes de rede trabalharam de uma forma mais inteligente a fim de diminuir o tráfego de dados no *fronthaul* e diminuir a latência na comunicação. O Nuoxus surge como um modelo para realização de *caching* em F-RANs, visando utilizar o seu potencial para contribuir com os objetivos da F-RAN.

A latência de rede é determinada por inúmeros fatores em uma implementação de rede (CHOI et al., 2004). Porém, como entre os cenários avaliados a única alteração é a estratégia de *caching* utilizada, espera-se obter uma redução da utilização e do tráfego de dados no *fronthaul*, diminuindo o número de colisões e, portanto, reduzindo a latência na comunicação e da rede em geral. A latência média (L) da rede será dada pela seguinte equação:

$$L = \frac{\sum_{i=0}^{n} t_c(i) - t_s(i)}{n} \tag{5.3}$$

Onde $t_s(i)$ e $t_c(i)$ são os tempos de saída e de chegada do pacote i ao seu destino, respectivamente.

6 RESULTADOS

Esse capítulo apresenta os resultados obtidos através da simulação dos cenários de testes descritos no capítulo anterior. Primeiramente, são apresentados os resultados relacionados à aplicação da estratégia de *caching* e uma análise de sua eficiência quando comparado com as demais estratégias. Após isso, o impacto da utilização do Nuoxus na latência de rede será apresentado através dos resultados obtidos para essa métrica.

6.1 Eficiência da Política de Substituição

O primeiro aspecto avaliado é a eficiência da nova política de substituição proposta através do Nuoxus. A fim de facilitar a compreensão da análise dos resultados, cada métrica associada com esse aspecto de é apresentado em uma subseção a seguir.

6.1.1 Utilização de Cache

Os resultados relacionados com a primeira métrica (C), são apresentadas através da Tabela 5. Essa tabela traz o total número de requisições, o número de vezes que o conteúdo requisitado estava armazenado na *cache* da F-AP e o número de vezes que o conteúdo requisitado estava na *cache* do F-UE. Ela também traz a métrica de avaliação C, seguido de seu desvio padrão σ **C**. Essa métrica está também representada através do gráfico da Figura 19.

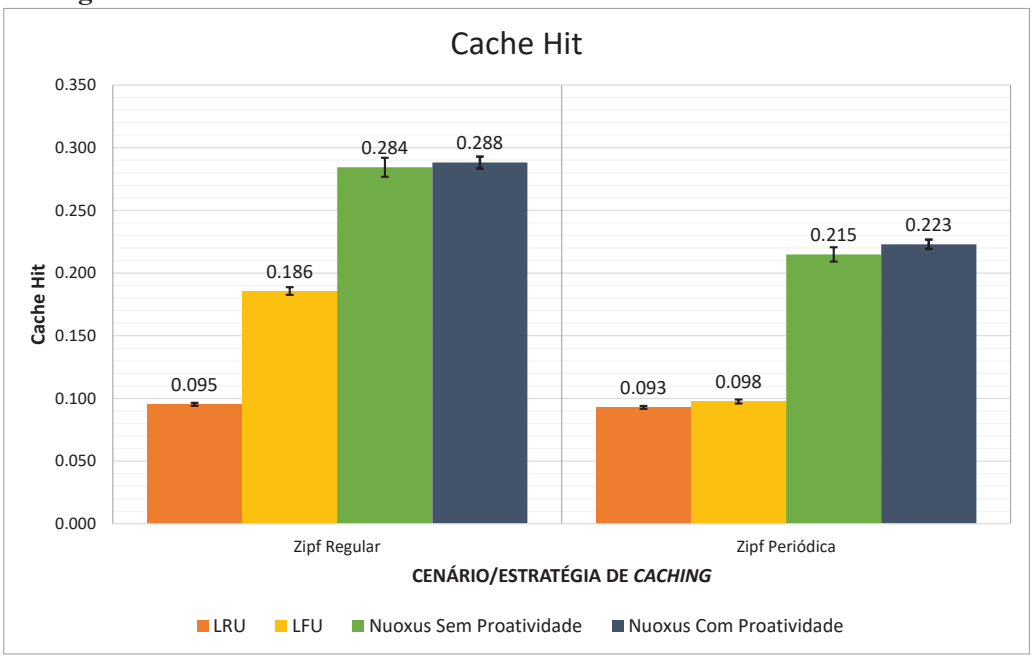
Tabela 5: Resultados relacionados ao acesso de arquivos em *cache* (**C**).

Cenário	Requisições	*Cache Hit - F-AP*	*Cache Hit F-UE*	C	σ C
$\mathbf{C}_1$	158 543	0	0	0,000 0	0,000 00
$\mathbf{C}_2$	158 543	29 446	0	0,185 7	0,003 03
$\mathbf{C}_3$	158 543	15 111	0	0,095 3	0,001 19
$\mathbf{C}_4$	158 543	45 101	0	0,284 4	0,004 74
$\mathbf{C}_5$	158 543	15 634	30 057	0,288 1	0,003 49
$\mathbf{C}_6$	157 640	0	0	0,000 0	0,000 00
$\mathbf{C}_7$	157 640	15 374	0	0,097 5	0,001 11
$\mathbf{C}_8$	157 640	14 637	0	0,092 9	0,000 73
$\mathbf{C}_9$	157 640	33 877	0	0,214 9	0,005 80
$\mathbf{C}_{10}$	157 640	20 114	15 035	0,223 0	0,003 60

Fonte: Elaborada pelo autor.

A primeira informação relevante para a análise é o número total de requisições. Nota-se que os cenários C_1 a C_5, os quais utilizam a distribuição de popularidade Zipf Regular, e os cenários C_6 a C_{10}, os quais utilizam a distribuição de popularidade Zipf Periódica, possuem o mesmo número médio de requisições entre si. Esse é o comportamento esperado, visto que todos os cenários foram executados utilizando a mesma semente e o mesmo conjunto de avanços na *stream* de geração número aleatórios. Portanto, esse número comprova que os F-UEs requi-

Figura 19: Gráfico contento os resultados relacionados à métrica de *Cache Hit*.



Fonte: Elaborada pelo autor.

sitaram exatamente a mesma sequência de arquivos entre os cenários que utilizaram a mesma distribuição de popularidade.

A segunda informação é o número de *cache hit*. Como já esperado, é possível notar que os cenários C_1 e C_6, que são os que não possuem nenhum tipo de estratégia de *caching*, não há nenhum *cache hit*. Já os cenários C_5 e C_{10}, que são os que utilizam o *caching* proativo, além de apresentarem acesso à *cache* das F-APs, também apresentam acesso à *cache* dos F-UEs. Para o cenário que utiliza o Nuoxus com proatividade e Zipf Regular, cerca de 66% dos acessos à *cache* ocorreram nos F-UEs, e para o cenário que Nuoxus com proatividade e Zipf Periódica, o percentual de acesso que foram supridos pelo conteúdo armazenado na *cache* dos F-UEs foi de cerca de 43%. Apesar da métrica C não refletir o local do conteúdo acessado, a métrica L acaba sendo impactada.

Os resultados sugerem que a estratégia de *caching* do Nuoxus **sem proatividade**, apresenta um aumento no aproveitamento de *cache* de 53,16% ao comparar com o LFU e de 198,46% ao compará-lo com o LRU nos cenários onde a distribuição de popularidade é a Zipf Regular. Já nos cenários onde a distribuição de popularidade é a Zipf Periódica, esses números tornam-se 120,41% e 131,32%, respectivamente.

Ao comparar o Nuoxus com proatividade e sem proatividade, pode-se perceber que o aproveitamento de *cache* de ambas as estratégias são bem similares. O Nuoxus com proatividade apresentou um desempenho ligeiramente pior que o Nuoxus sem proatividade nos cenários Zipf Regular. Porém, considerando o desvio padrão de ambos os resultados, é possível afirmar que não há diferenças no aproveitamento de *cache*. Similarmente, apesar do Nuoxus com proatividade ter apresentado um melhor aproveitamento de *cache* nos cenários com distribuição Zipf

Periódica, devido ao desvio padrão dos resultados obtidos, pode-se apensas afirmar que seu desempenho é similar ao Nuoxus sem proatividade. Como já descrito anteriormente, a principal diferença relacionado ao acesso ao conteúdo armazenado em *cache* entre o Nuoxus sem proatividade e o Nuoxus com proatividade acaba sendo a localização do conteúdo armazenado, o que impacta a latência de rede.

Por fim, é importante observar o quão resiliente as estratégias do Nuxous se mostraram ao trocar a distribuição de popularidade dos cenários. O Nuoxus sem proatividade teve uma queda de 24,89% na utilização de *cache* ao comparar o cenário C_4 com o C_9. Já a segunda estratégia com melhores resultados, o LFU, teve uma redução de 47,79% ao comparar o cenário C_2 com o cenário C_7.

6.1.2 Substituição de Arquivos

O segundo ponto avaliado é o quão frequentemente os arquivos são substituídos em *cache*, pois isso implica em um maior gasto em operações de entrada e saída. Os resultados obtidos são apresentados através da Tabela 6. Nessa tabela, novamente é apresentado o número médio de requisições realizadas, seguido do número de substituições realizadas. Por fim, a tabela apresenta a métrica definida **S**, seguido de seu desvio padrão σ **S**. Os resultados da métrica também são apresentados de maneira visual através do gráfico da Figura 20.

Tabela 6: Resultados relacionados ao número de substituições de arquivos em *cache* (**S**).

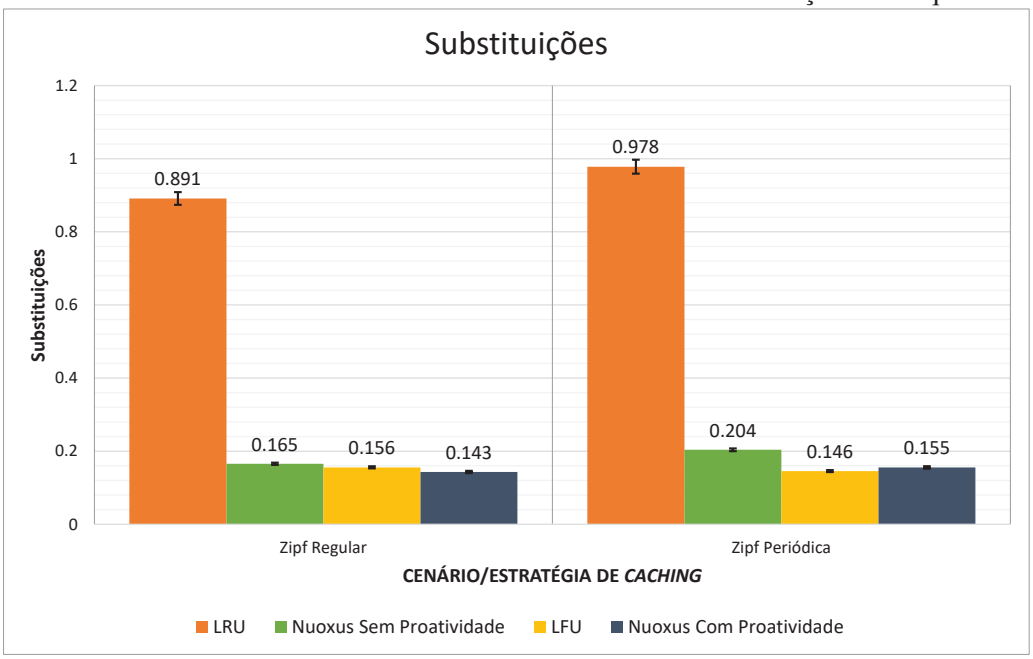
Cenário	Requisições	Substituições	S	σ S
$\mathbf{C}_1$	158 543	0	0,000	0,000 0
$\mathbf{C}_2$	158 543	24 700	0,156	0,002 9
$\mathbf{C}_3$	158 543	141 298	0,891	0,017 4
$\mathbf{C}_4$	158 543	26 238	0,165	0,003 1
$\mathbf{C}_5$	158 543	22 667	0,143	0,002 6
$\mathbf{C}_6$	157 640	0	0,000	0,000 0
$\mathbf{C}_7$	157 640	22 939	0,146	0,001 2
$\mathbf{C}_8$	157 640	154 207	0,978	0,008 3
$\mathbf{C}_9$	157 640	32 116	0,204	0,002 5
$\mathbf{C}_{10}$	157 640	24 499	0,155	0,001 4

Fonte: Elaborada pelo autor.

É evidente que a estratégia com piores resultados nesse quesito é o LRU (cenários C_3 e C_8),s endo que cerca de 89,1% das requisições resultaram em substituição nos cenários com Zipf Regular (cenários C_1 a C_5). Nos cenários com Zipf Periodica (cenários C_6 a C_{10}), esse número sobe para 97,8%. Esse comportamento já era esperado, pois nessa estratégia o único fator considerado é o quão recente um arquivo foi requisitado. Então, mesmo que um arquivo foi requisitado apenas uma única vez, ele é armazenado em *cache*.

Já ao comparar a estratégia Nuoxus com a LFU, os resultados sugerem que ambas têm um desempenho bastante similar. Nos testes com distribuição de popularidade Zipf Regular, o

Figura 20: Gráfico contento os resultados relacionados à métrica de Substituições de Arquivos em Cache



Fonte: Elaborada pelo autor.

Nuoxus sem proatividade aumentou o número de substituições em 6,22% ao compará-lo com o LFU. Já o Nuoxus com proatividade reduziu o número de substituições em 8,23% ao compará-lo com o mesmo algoritmo. As maiores diferenças surgem nos cenários com Zipf Periódica, onde o percentual de substituições do Nuoxus sem proatividade é 40% maior que o LFU e do Nuoxus com proatividade é 6,8% maior.

É importante destacar nessa métrica é que, mesmo apresentando os melhores resultados de *cache hit*, os cenários que utilizam a estratégia de *caching* do Nuoxus não apresentam uma taxa de substituição muito maior. O que indica que as operações de entrada e saída do dispositivo de armazenamento não seriam um impeditivo para a adoção dessa estratégia.

6.2 Latência de Rede

Como mencionado anteriormente, a latência de rede é formada por uma série de fatores (CHOI et al., 2004). Porém, dado as condições dos testes e visto que o único fator que é alterado entre os cenários é a estratégia de *caching* utilizada, podemos dizer que essa métrica é um indicativo direto do impacto da redução de requisições de arquivos ao servidor de *cache*, visto que esse ato diminui o tráfego de dados no *fronthaul*, reduzindo o atraso da rede. A Tabela 7 apresenta os resultados obtidos relacionados à essa métrica e a Figura 21 apresenta os mesmos resultados de uma maneira gráfica.

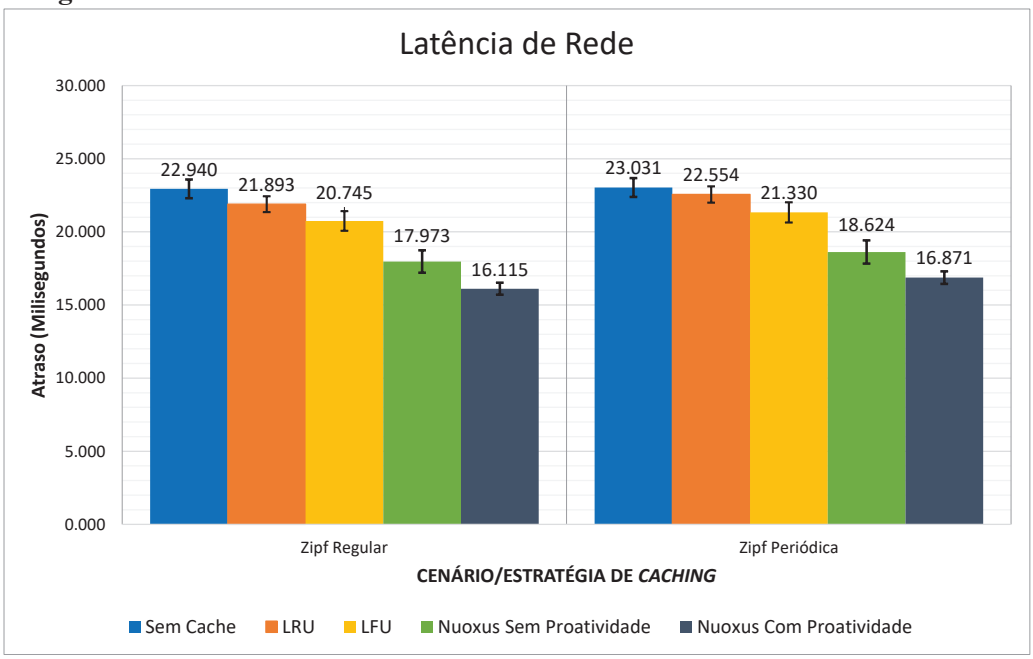
Os resultados obtidos demostram que, para os cenários Zipf Regular, o Nuoxus sem proatividade apresentou uma redução de 21,65% na latência quando comparado com o cenário sem *caching*, uma redução de 13,36% quando comparado com o cenário que utiliza LFU e 17,09%

Tabela 7: Resultados relacionados à Latência de Rede (**L**) em milissegundos.

Cenário	L	σ L
$\mathbf{C}_1$	22,940 3	0,187 05
$\mathbf{C}_2$	20,745 3	0,601 99
$\mathbf{C}_3$	21,892 5	0,434 11
$\mathbf{C}_4$	17,973 2	0,302 17
$\mathbf{C}_5$	16,114 6	0,443 06
$\mathbf{C}_6$	23,031 5	0,643 53
$\mathbf{C}_7$	21,330 5	0,691 70
$\mathbf{C}_8$	22,553 8	0,558 19
$\mathbf{C}_9$	18,624 3	0,794 01
$\mathbf{C}_{10}$	16,871 1	0,429 23

Fonte: Elaborada pelo autor.

Figura 21: Gráfico contento os resultados relacionados à métrica Latência de Rede



Fonte: Elaborada pelo autor.

quando comparado com o LRU. Ao utilizarmos o Nuoxus com proatividade como base, o percentual de redução no atraso aumenta para 29,75%, 22,32% e 26,39%, respectivamente.

Já nos cenários Zipf Periódica, a redução de atraso do Nuoxus sem proatividade é de 19,13% quando comparado com o cenário sem *caching*, de 12,69% quando comparado com o LFU e de 17,42% quando comparado com o LRU. Esses números são 26,75%, 20,91% e 25,20% ao utilizarmos o Nuoxus com proatividade como base de comparação.

Conforme mencionado anteriormente, o impacto do local em que o *cache hit* ocorre pode ser melhor observado através dessa métrica. Por exemplo, analisando os cenários onde o Zipf Regular foi utilizado, o Nuoxus sem proatividade e com proatividade obtiveram resultados bastante similares em termos de *cache hit* (28,4% e 28,8%, respectivamente). Porém, como cerca

de 66% dos arquivos resgatados de *cache* estavam armazenados nos F-UEs, a redução na latência de rede foi de cerca de 10% quando comparamos o Nuoxus com proatividade com o Nuoxus sem proatividade.

Similarmente, nos cenários que utilizam Zipf Periódica, o percentual de *cache hit* foi de 21,5% para o Nuoxus sem proatividade e de 22,3% para o Nuoxus com proatividade. Porém, como cerca de 43% dos acessos à *cache* no Nuoxus com proatividade ocorreu diretamente nos F-UEs, a redução da latência de rede foi de 9,4%.

6.3 Considerações do Capítulo

Considerando os resultados obtidos para os cenários testados, é possível afirmar que o Nuoxus proativo é a estratégia que apresenta o maior ganho. Porém, mesmo em situações onde ela não possa ser utilizada, por exemplo, enquanto os dispositivos de borda não possuírem capacidade de *caching*, a estratégia do Nuoxus sem proatividade ainda se sobressai quando comparada com outras estratégias populares, como LFU e LRU.

A única métrica em que o Nuoxus não apresenta os melhores resultados é a relacionada com o número de substituições de arquivos, onde o aumento do número de substituições ao utilizar o Nuoxus sem proatividade ao invés do LFU é de 40% para o cenário que utiliza Zipf Periódica. Porém, ao analisar a métrica, verifica-se que o percentual de requisições que geraram substituições aumentou de 14.6% para 20.4%, então, no total o aumento real é de cerca de 5.8% das requisições. Infelizmente, o tempo de acesso ao disco não foi simulado durante os testes, então não há como afirmar qual é o impacto real desse aumento no tempo total de processamento das requisições.

Porém, para uma F-RAN, a métrica mais importante é a redução do número de acessos ao *fronthaul* (PENG et al., 2016), que nesse trabalho foi medido de duas formas: através do acesso ao cache e do impacto desse acesso à latência de rede. Em ambas as medidas, o Nuoxus se sobressaiu em relação as demais estratégias de *caching*, sugerindo que F-RANs se beneficiariam ao aplicar o modelo.

7 CONCLUSÃO

Nos próximos anos, espera-se que o número de dispositivos móveis conectados a uma rede sem fio de longa distância tenha um aumento significativo suficiente para esgotar os recursos oferecidos pelas tecnologias atuais (Samsung Electronics, 2015).Devido a essas projeções, a comunidade cientifica e as indústrias estão colocando esforços na definição das redes 5G, e uma das arquiteturas mais recentes proposta para atender às suas especificações é chamada de F-RAN (PENG et al., 2016) Além disso, estudos de tráfego de rede relatam que a maior parte dos dados consumidos por dispositivos móveis consiste em conteúdo multimídia (CISCO, 2017).

Dado esse contexto, esse trabalho apresentou o modelo Nuoxus, cujo objetivo principal é a realização de *caching* de conteúdo multimídia em dispositivos de bordas em uma arquitetura F-RAN, com a finalidade de amenizar um dos maiores problemas desse tipo de arquitetura: o intenso tráfego de dados no *fronthaul*. Diferente de trabalhos relacionados, o Nuoxus oferece *caching* de forma proativa sem a necessidade de um componente centralizado para fazer a coordenação, atuando de forma contínua no nó que o executa.

As contribuições cientificas desse trabalho podem ser resumidas nos seguintes dois pontos:

- Um modelo de *caching* multicamadas, voltado para arquiteturas F-RAN, com capacidade de realizar o *caching* de conteúdo de forma proativa, onde as camadas superiores possuem capacidade de sugerir o armazenamento de conteúdo na *cache* de camadas inferiores antes mesmo do usuário final requisitá-lo;

- Uma nova estratégia de *caching* onde a similaridade de cosseno entre o nó requisitante e o restante da rede é um dos fatores considerados para determinar qual arquivos tem maiores chances de serem requisitados no futuro.

Os resultados das simulações demostram que o modelo proposto apresenta os melhores resultados entre as estratégias de *caching* avaliadas. A aplicação da estratégia de *caching* do Nuoxus foi capaz de aumentar o número de acessos ao conteúdo armazenado na *cache* em até 54,62% (Nuoxus sem proatividade) e em 56,27% (Nuoxus com proatividade). O aumento do número de acessos ao conteúdo na *cache* dos nós implica na redução no número de acessos ao *fronthaul*, mitigando um dos maiores problemas das F-RANs, e resultando em uma redução na latência da rede.

Como trabalhos futuros, sugere-se a integração do Nuoxus com um modelo real de F-RAN quando houver ferramental disponível, fazendo o teste com uma carga real de rede. Além disso, há a possibilidade de explorar o impacto do tamanho do espaço de armazenamento dos dispositivos na eficiência de *caching* do modelo, assim como adaptá-lo para armazenar partes dos arquivos, possibilitando uma maior diversidade de conteúdo. Por fim, dado a visão de um baixo consumo de energia para essas futuras redes, também é necessário testar o impacto da utilização do modelo nesse aspecto.

REFERÊNCIAS

3GPP. **Requirements for further advancements for Evolved Universal Terrestrial Radio Access (E-UTRA) (LTE-Advanced)**. [S.l.]: 3rd Generation Partnership Project (3GPP), 2015. TS. (36.913).

3GPP. **Study on New Services and Markets Technology Enablers**. [S.l.]: 3rd Generation Partnership Project (3GPP), 2016. TS. (22.891).

5G PPP Architecture Working Group. **View on 5G Architecture**. Heidelberg, Germany: [s.n.], 2016.

AHLEHAGH, H.; DEY, S. Hierarchical video caching in wireless cloud: approaches and algorithms. In: IEEE INTERNATIONAL CONFERENCE ON COMMUNICATIONS (ICC), 2012., 2012, Washington, DC, USA. **Anais...** IEEE, 2012. p. 7082–7087.

HENNEQUIN, P. L. (Ed.). Exchangeability and related topics. In: _____. **École d'Été de Probabilités de Saint-Flour XIII — 1983**. Berlin, Heidelberg: Springer Berlin Heidelberg, 1985. v. 1117, p. 1—198.

ALFIAD, M. S.; BORNE, D. van den; WUTH, T.; KUSCHNEROV, M.; LANKL, B.; WEISKE, C.; MAN, E. de; NAPOLI, A.; WAARDT, H. de. 111-Gb/s POLMUX-RZ-DQPSK transmission over 1140 km of SSMF with 10.7-Gb/s NRZ-OOK neighbours. In: EUROPEAN CONFERENCE ON OPTICAL COMMUNICATION, 2008., 2008, Brussels, Belgium. **Anais...** [S.l.: s.n.], 2008. p. 1–2.

AMENTIE, M. D.; SHENG, M.; SONG, J.; LIU, J. Minimum delay guaranteed cooperative device-to-device caching in 5G wireless networks. In: INTERNATIONAL CONFERENCE ON WIRELESS COMMUNICATIONS SIGNAL PROCESSING (WCSP), 2016., 2016, Washington, DC, USA. **Anais...** IEEE, 2016. p. 1–5.

BASTUG, E.; BENNIS, M.; DEBBAH, M. Living on the edge: the role of proactive caching in 5g wireless networks. **IEEE Communications Magazine**, Washington, DC, USA, v. 52, n. 8, p. 82–89, August 2014.

BASTUG, E.; BENNIS, M.; DEBBAH, M. A transfer learning approach for cache-enabled wireless networks. In: INTERNATIONAL SYMPOSIUM ON MODELING AND OPTIMIZATION IN MOBILE, AD HOC, AND WIRELESS NETWORKS (WIOPT), 2015., 2015, Washington, DC, USA. **Anais...** IEEE, 2015. p. 161–166.

BAşTUğ, E.; BENNIS, M.; ZEYDAN, E.; KADER, M. A.; KARATEPE, I. A.; ER, A. S.; DEBBAH, M. Big data meets telcos: a proactive caching perspective. **Journal of Communications and Networks**, Washington, DC, USA, v. 17, n. 6, p. 549–557, December 2015.

BHARDWAJ, S.; KOLE, A. Review and study of internet of things: it's the future. In: INTERNATIONAL CONFERENCE ON INTELLIGENT CONTROL POWER AND INSTRUMENTATION (ICICPI), 2016., 2016, Kolkata, India. **Anais...** IEEE, 2016. p. 47–50.

BONOMI, F.; MILITO, R.; ZHU, J.; ADDEPALLI, S. Fog Computing and Its Role in the Internet of Things. In: FIRST EDITION OF THE MCC WORKSHOP ON MOBILE CLOUD COMPUTING, 2012, New York, NY, USA. **Proceedings...** ACM, 2012. p. 13–16. (MCC '12).

BORST, S.; GUPTA, V.; WALID, A. Distributed Caching Algorithms for Content Distribution Networks. In: PROCEEDINGS IEEE INFOCOM, 2010., 2010, Washington, DC, USA. **Anais...** IEEE, 2010. p. 1–9.

BRESLAU, L.; CAO, P.; FAN, L.; PHILLIPS, G.; SHENKER, S. Web caching and Zipf-like distributions: evidence and implications. In: INFOCOM '99. EIGHTEENTH ANNUAL JOINT CONFERENCE OF THE IEEE COMPUTER AND COMMUNICATIONS SOCIETIES. PROCEEDINGS. IEEE, 1999, Washington, DC, USA. **Anais...** IEEE, 1999. v. 1, p. 126–134 vol.1.

BRODERSEN, A.; SCELLATO, S.; WATTENHOFER, M. YouTube Around the World: geographic popularity of videos. In: INTERNATIONAL CONFERENCE ON WORLD WIDE WEB, 21., 2012, New York, NY, USA. **Proceedings...** ACM, 2012. p. 241–250. (WWW '12).

BYERS, C. C. Architectural Imperatives for Fog Computing: use cases, requirements, and architectural techniques for fog-enabled iot networks. **IEEE Communications Magazine**, Washington, DC, USA, v. 55, n. 8, p. 14–20, 2017.

CHEN, C. J. User Adoption Decisions in Self-Service Technologies: a study of the internet banking. In: IIAI INTERNATIONAL CONGRESS ON ADVANCED APPLIED INFORMATICS (IIAI-AAI), 2016., 2016, Kumamoto, Japan. **Anais...** CPS Publishing Inc.EEE, 2016. p. 1207–1208.

China Mobile. **The Road Towards Green RAN - White Paper**. China: China Mobile Research Institute, 2010. TS.

CHOI, B. K.; MOON, S.; ZHANG, Z.-L.; PAPAGIANNAKI, K.; DIOT, C. Analysis of point-to-point packet delay in an operational network. In: IEEE INFOCOM 2004, 2004, Hong Kong. **Anais...** [S.l.: s.n.], 2004. v. 3, p. 1797–1807 vol.3.

CISCO. **Cisco Visual Networking Index**: global mobile data traffic forecast update, 2016–2021 white paper. Washington, DC, USA: [s.n.], 2017. TS. (1454457600805266).

COMSCORE. **comScore Releases January 2014 U.S. Online Video Rankings**. Disponível em: <http://www.comscore.com/Insights/Press-Releases/2014/2/comScore-Releases-January-2014-US-Online-Video-Rankings>. Acesso em: 06 de junho de 2017.

DENNING, P. J. The locality principle. **Communications of the ACM**, New York, NY, USA, v. 48, n. 7, p. 19–24, 2005.

DIESTEL, R. **Graph Theory (Graduate Texts in Mathematics)**. Heidelberg, Germany: Springer, 2010.

ELAARAG, H. A Quantitative Study of Web Cache Replacement Strategies Using Simulation. In: _____. **Web Proxy Cache Replacement Strategies**: simulation, implementation, and performance evaluation. London, UK: Springer London, 2013. p. 17–60.

ENCODING. **Understanding Bitrates in Video Files**. Disponível em: <http://help.encoding.com/knowledge-base/article/understanding-bitrates-in-video-files>. Acesso em: 06 de junho de 2017.

FENG, B.; JIANG, L.; FENG, G.; QIN, S.; GUO, Y. Network Coding Based Content Caching in Hierarchical Cloud Service Network for 5G. In: IEEE GLOBAL COMMUNICATIONS CONFERENCE (GLOBECOM), 2016., 2016, Washington, DC, USA. **Anais...** IEEE, 2016. p. 1–6.

FIORE, M.; MININNI, F.; CASETTI, C.; CHIASSERINI, C. F. To Cache or Not To Cache? In: IEEE INFOCOM 2009, 2009, Washington, DC, USA. **Anais...** IEEE, 2009. p. 235–243.

GILL, P.; ARLITT, M.; LI, Z.; MAHANTI, A. Youtube Traffic Characterization: a view from the edge. In: ACM SIGCOMM CONFERENCE ON INTERNET MEASUREMENT, 7., 2007, San Diego, CA, USA. **Proceedings...** ACM, 2007. p. 15–28. (IMC '07).

GUAN, Y.; XIAO, Y.; FENG, H.; SHEN, C. C.; CIMINI, L. J. MobiCacher: mobility-aware content caching in small-cell networks. In: IEEE GLOBAL COMMUNICATIONS CONFERENCE, 2014., 2014, Washington, DC, USA. **Anais...** IEEE, 2014. p. 4537–4542.

HALL, M. **Combinatorial Theory (Wiley Classics Library)**. New York, NY, USA: Wiley-Interscience, 1986.

HAMERS, L.; HEMERYCK, Y.; HERWEYERS, G.; JANSSEN, M.; KETERS, H.; ROUSSEAU, R.; VANHOUTTE, A. Similarity measures in scientometric research: the jaccard index versus salton's cosine formula. **Information Processing & Management**, Oostende, Belgium, v. 25, n. 3, p. 315–318, 1989.

HUANG, X.; ZHAO, Z.; ZHANG, H. Latency Analysis of Cooperative Caching with Multicast for 5G Wireless Networks. In: IEEE/ACM 9TH INTERNATIONAL CONFERENCE ON UTILITY AND CLOUD COMPUTING (UCC), 2016., 2016, Washington, DC, USA. **Anais...** IEEE, 2016. p. 316–320.

JAYAREKHA, P.; NAIR, T. R. G. An Adaptive Dynamic Replacement Approach for a Multicast based Popularity Aware Prefix Cache Memory System. **CoRR**, Washington, DC, USA, v. abs/1001.4135, 2010.

JIANG, L.; FENG, G.; QIN, S. Cooperative content distribution for 5G systems based on distributed cloud service network. In: IEEE INTERNATIONAL CONFERENCE ON COMMUNICATION WORKSHOP (ICCW), 2015., 2015, London, UK. **Anais...** IEEE, 2015. p. 1125–1130.

JOHNSON, D. H. Signal-to-noise ratio. **Scholarpedia**, [S.l.], v. 1, n. 12, p. 2088, 2006. revisão #91770.

KAREDLA, R.; LOVE, J. S.; WHERRY, B. G. Caching strategies to improve disk system performance. **Computer**, Washington, DC, USA, v. 27, n. 3, p. 38–46, March 1994.

LEISERSON, C. E. Greedy Algorithms. In: _____. **Introduction To Algorithms**. Washington, DC, USA: THE MIT PRESS, 1990. p. 329.

LI, H.; HU, D.; CI, S. iCacheOS: in-ran caches orchestration strategy through content joint wireless and backhaul routing in small-cell networks. In: IEEE GLOBAL COMMUNICATIONS CONFERENCE (GLOBECOM), 2015., 2015, Washington, DC, USA. **Anais...** IEEE, 2015. p. 1–7.

LIU, A.; LAU, V.; HAN, W. Randomized user-centric clustering for cloud radio access network with PHY caching. In: IEEE GLOBAL CONFERENCE ON SIGNAL AND INFORMATION PROCESSING (GLOBALSIP), 2015., 2015, Washington, DC, USA. **Anais...** IEEE, 2015. p. 948–952.

MAHANTI, A.; WILLIAMSON, C.; EAGER, D. Traffic Analysis of a Web Proxy Caching Hierarchy. **Netwrk. Mag. of Global Internetwkg.**, Washington, DC, USA, v. 14, n. 3, p. 16–23, May 2000.

NS-3 Project. **NS-3 Documentation**. Disponível em: <www.nsnam.org/documentation>. Acesso em: 04 de junho de 2017.

OSSEIRAN, A.; BOCCARDI, F.; BRAUN, V.; KUSUME, K.; MARSCH, P.; MATERNIA, M.; QUESETH, O.; SCHELLMANN, M.; SCHOTTEN, H.; TAOKA, H.; TULLBERG, H.; UUSITALO, M. A.; TIMUS, B.; FALLGREN, M. Scenarios for 5G mobile and wireless communications: the vision of the metis project. **IEEE Communications Magazine**, Washington, DC, USA, v. 52, n. 5, p. 26–35, May 2014.

PAN, S. J.; YANG, Q. A Survey on Transfer Learning. **IEEE Transactions on Knowledge and Data Engineering**, Washington, DC, USA, v. 22, n. 10, p. 1345–1359, October 2010.

PARK, S. H.; SIMEONE, O.; SHAMAI, S. Joint optimization of cloud and edge processing for fog radio access networks. In: IEEE INTERNATIONAL SYMPOSIUM ON INFORMATION THEORY (ISIT), 2016., 2016, Washington, DC, USA. **Anais...** IEEE, 2016. p. 315–319.

PENG, M.; LI, Y.; JIANG, J.; LI, J.; WANG, C. Heterogeneous cloud radio access networks: a new perspective for enhancing spectral and energy efficiencies. **IEEE Wireless Communications**, Washington, DC, USA, v. 21, n. 6, p. 126–135, December 2014.

PENG, M.; YAN, S.; ZHANG, K.; WANG, C. Fog-computing-based radio access networks: issues and challenges. **IEEE Network**, Washington, DC, USA, v. 30, n. 4, p. 46–53, July 2016.

PIRO, G.; BALDO, N.; MIOZZO, M. An LTE module for the ns-3 network simulator. In: INTERNATIONAL ICST CONFERENCE ON SIMULATION TOOLS AND TECHNIQUES, 4., 2011, Washington, DC, USA. **Proceedings...** ACM, 2011. p. 415–422.

POLI, R.; KENNEDY, J.; BLACKWELL, T. Particle swarm optimization. **Swarm Intelligence**, NY, USA, v. 1, n. 1, p. 33–57, 2007.

POULARAKIS, K.; IOSIFIDIS, G.; SOURLAS, V.; TASSIULAS, L. Exploiting Caching and Multicast for 5G Wireless Networks. **IEEE Transactions on Wireless Communications**, Washington, DC, USA, v. 15, n. 4, p. 2995–3007, April 2016.

POUSHTER, J. **Smartphone Ownership and Internet Usage Continues to Climb in Emerging Economies**. Washington, DC, USA: PewResearchCenter, 2016.

WEHRLE, K.; GÜNEŞ, M.; GROSS, J. (Ed.). The ns-3 Network Simulator. In: _____. **Modeling and Tools for Network Simulation**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. p. 15–34.

ROUIL, R.; CINTRóN, F. J.; BEN MOSBAH, A.; GAMBOA, S. Implementation and Validation of an LTE D2D Model for Ns-3. In: WORKSHOP ON NS-3, 2017, New York, NY, USA. **Proceedings...** ACM, 2017. p. 55–62. (WNS3 '17).

Samsung Electronics. **5G Vision**. Korea: [s.n.], 2015.

SCELLATO, S.; MASCOLO, C.; MUSOLESI, M.; CROWCROFT, J. Track Globally, Deliver Locally: improving content delivery networks by tracking geographic social cascades. In: INTERNATIONAL CONFERENCE ON WORLD WIDE WEB, 20., 2011, New York, NY, USA. **Proceedings...** ACM, 2011. p. 457–466. (WWW '11).

SHANMUGAM, K.; GOLREZAEI, N.; DIMAKIS, A. G.; MOLISCH, A. F.; CAIRE, G. FemtoCaching: wireless content delivery through distributed caching helpers. **IEEE Transactions on Information Theory**, Washington, DC, USA, v. 59, n. 12, p. 8402–8413, December 2013.

SIDOROV, G.; GELBUKH, A.; GÓMEZ-ADORNO, H.; PINTO, D. Soft similarity and soft cosine measure: similarity of features in vector space model. **Computación y Sistemas**, Gustavo A. Madero, Mexico, v. 18, n. 3, p. 491–504, 2014.

SU, Z.; XU, Q.; HOU, F.; YANG, Q.; QI, Q. Edge Caching for Layered Video Contents in Mobile Social Networks. **IEEE Transactions on Multimedia**, Washington, DC, USA, v. 19, n. 10, p. 2210–2221, Oct 2017.

TANENBAUM, A. **Modern Operating Systems**. Upper Saddle River, NJ, USA: Prentice Hall, 1992. (Prentice-Hall international series).

TAO, M.; CHEN, E.; ZHOU, H.; YU, W. Content-Centric Sparse Multicast Beamforming for Cache-Enabled Cloud RAN. **IEEE Transactions on Wireless Communications**, Washington, DC, USA, v. 15, n. 9, p. 6118–6131, September 2016.

TATAR, A.; AMORIM, M. D. de; FDIDA, S.; ANTONIADIS, P. A survey on predicting the popularity of web content. **Journal of Internet Services and Applications**, Washington, DC, USA, v. 5, n. 1, p. 8, 2014.

TRAN, T. X.; POMPILI, D. Octopus: a cooperative hierarchical caching strategy for cloud radio access networks. In: IEEE 13TH INTERNATIONAL CONFERENCE ON MOBILE AD HOC AND SENSOR SYSTEMS (MASS), 2016., 2016, Washington, DC, USA. **Anais...** IEEE, 2016. p. 154–162.

WANG, L.; WU, H.; WANG, W.; CHEN, K. C. Socially enabled wireless networks: resource allocation via bipartite graph matching. **IEEE Communications Magazine**, Washington, DC, USA, v. 53, n. 10, p. 128–135, October 2015.

WANG, S.; ZHANG, X.; YANG, K.; WANG, L.; WANG, W. Distributed edge caching scheme considering the tradeoff between the diversity and redundancy of cached content. In: IEEE/CIC INTERNATIONAL CONFERENCE ON COMMUNICATIONS IN CHINA (ICCC), 2015., 2015, Shenzhen, China. **Anais...** IEEE, 2015. p. 1–5.

WANG, X.; CHEN, M.; TALEB, T.; KSENTINI, A.; LEUNG, V. C. M. Cache in the air: exploiting content caching and delivery techniques for 5g systems. **IEEE Communications Magazine**, Washington, DC, USA, v. 52, n. 2, p. 131–139, February 2014.

WANG, X.; CHEN, M.; TALEB, T.; KSENTINI, A.; LEUNG, V. C. M. Cache in the air: exploiting content caching and delivery techniques for 5g systems. **IEEE Communications Magazine**, Washington, DC, USA, v. 52, n. 2, p. 131–139, February 2014.

XU, F.; YAO, H.; ZHAO, C.; QIU, C. Towards next generation software-defined radio access network–architecture, deployment, and use case. **EURASIP Journal on Wireless Communications and Networking**, Cham, Switzerland, v. 2016, n. 1, p. 264, 2016.

YANG, Q.; ZHANG, H. H.; ZHANG, H. Taylor series prediction: a cache replacement policy based on second-order trend analysis. In: ANNUAL HAWAII INTERNATIONAL CONFERENCE ON SYSTEM SCIENCES, 34., 2001, Washington, DC, USA. **Proceedings...** IEEE, 2001. p. 7 pp.–.

YU, Y. J.; TSAI, W. C.; PANG, A. C. Backhaul Traffic Minimization under Cache-Enabled CoMP Transmissions over 5G Cellular Systems. In: IEEE GLOBAL COMMUNICATIONS CONFERENCE (GLOBECOM), 2016., 2016, Washington, DC, USA. **Anais...** IEEE, 2016. n. 3, p. 1–7.

ZEYDAN, E.; BASTUG, E.; BENNIS, M.; KADER, M. A.; KARATEPE, I. A.; ER, A. S.; DEBBAH, M. Big data caching for networking: moving from cloud to edge. **IEEE Communications Magazine**, Washington, DC, USA, v. 54, n. 9, p. 36–42, September 2016.