Uma avaliação do impacto de políticas de seleção de servidores cache na borda das redes móveis

Leonardo Fiório Soares[1], **Ian Vilar Bastos**[2], **Igor Monteiro Moraes**[1]

[1] Laboratório Midiacom – Instituto de Computação
Universidade Federal Fluminense (UFF)
Niterói – RJ – Brasil

[2]Departamento de Eletrônica e Telecomunicações - Faculdade de Engenharia
Universidade do Estado do Rio de Janeiro (UERJ)
Rio de Janeiro – RJ – Brasil

leonardofiorio@id.uff.br, ian.bastos@eng.uerj.br, igor@ic.uff.br

Abstract. *The objective of this paper is to show that cache server selection policies have a reduced impact on cache efficiency at the edge of mobile networks. To do this, simulations are performed by varying the server selection policies, the storage capacity, and the distribution of storage among cache servers. The efficiency metrics used are the average hit and the average delay for retrieving the content segments. The results show that the different server selection policies have less than 1% performance difference between each other for the average delay for obtaining the segments. On the other hand, storage capacity has a large impact on cache efficiency. A 7 times larger storage capacity can reduce the average delay by 36%. The results also show that concentrating the cache storage capacity on a single server instead of distributing it can reduce the maintenance cost by impacting the average delay by less than 3%.*

Resumo. *O objetivo deste artigo é mostrar que políticas de seleção de servidores cache têm impacto reduzido no desempenho do cache na borda das redes móveis. Além disso, este artigo mostra que concentrar a capacidade de armazenamento do cache em um único servidor pode reduzir o custo sem prejudicar o desempenho do cache. Para isso, são feitas simulações variando as políticas de seleção de servidores, a capacidade de armazenamento e a distribuição do armazenamento entre os servidores cache. As métricas de eficiência utilizadas são o acerto médio e o atraso médio para a obtenção dos segmentos de conteúdo. Os resultados mostram que as diferentes políticas de seleção de servidores apresentam menos de 1% de diferença de desempenho entre si para o atraso médio de obtenção dos segmentos. Por outro lado, a capacidade de armazenamento tem grande impacto no desempenho do cache. Uma capacidade de armazenamento 7 vezes maior pode reduzir 36% o atraso médio. Os resultados também mostram que concentrar a capacidade de armazenamento do cache em um único servidor ao invés de distribuí-la pode reduzir o custo de manutenção impactando em menos de 3% o atraso médio.*

1. Introdução

A distribuição de vídeo na Internet é uma aplicação que se torna mais popular a cada dia e que exige cada vez mais recursos da rede. Um exemplo disso é a exibição do programa

Big Brother Brasil em 2021 através do aplicativo Globoplay [Ravache 2021]. Segundo os dados divulgados, foram mais de 430 milhões de horas assistidas pelos usuários, das quais 350 milhões foram distribuição de vídeo ao vivo. Foi registrado um pico de requisições simultâneas 352% maior e um número de usuários 261% maior quando comparados a exibição do mesmo programa em 2020. Da mesma forma, cada vez mais usuários acessam a Internet através de redes móveis. Segundo previsões, mais de 70% da população global terá conectividade móvel em 2023 [Cisco 2021]. Consequentemente, o número de usuários de aplicações de distribuição de vídeo que usam redes móveis para receber o conteúdo de vídeo também aumenta.

Uma das propostas para lidar com o aumento do número de usuários de aplicações de distribuição de vídeo em redes móveis é a implementação de servidores de *cache* usando a computação de borda e a virtualização de funções de rede (Virtualized Function Network - NFV) [Ge et al. 2020] para armazenar os segmentos dos conteúdos em pontos da rede mais próximos dos usuários. A computação de borda permite que sejam implementadas tarefas de processamento e armazenamento na borda da rede, ou seja, uma computação mais próxima dos dispositivos finais. Uma proposta para a computação de borda nas redes móveis é a Multi-Edge Computing (MEC), originalmente definida pelo European Telecommunications Standards Institute (ETSI) em 2014 [Hu et al. 2015]. A arquitetura MEC oferece uma solução de computação de borda nativamente voltada para as Redes de Acesso via Radio (*Radio Access Network* - RAN)[Dolui and Datta 2017]. Tal solução permite que as aplicações tenham acesso direto a informações do rádio que podem ser usadas para reconhecimento da localização e mobilidade dos dispositivos.

A tecnologia NFV consiste na implementação de serviços de rede de forma virtual e, por esse motivo, separa a implementação do serviço da necessidade de um dispositivo físico dedicado à função [de Souza and Duarte 2020]. Ao implementar servidores de *cache* como uma função virtual de rede nos servidores da borda da rede pode-se atender as requisições com conteúdos armazenados em *cache* e, assim reduzir o número de requisições atendidas através dos provedores dos conteúdos. Consequentemente, as requisições são atendidas sem a necessidade de encaminhá-las a uma rede externa, o que resulta em um menor atraso de recuperação.

Um desafio para a implementação de servidores MEC é a seleção de servidores. Existem trabalhos propondo políticas de seleção de servidores MEC executar aplicações e atender aos usuários. Para a implementação do *cache* em servidores MEC, a política de seleção de servidores *cache* consiste na forma de escolha de qual servidor MEC armazenará um novo segmento. O objetivo deste artigo é avaliar o impacto das políticas de seleção de servidores *cache* para o armazenamento de segmentos de conteúdo requisitados pelos usuários da rede móvel. Este artigo também avalia o desempenho do uso de múltiplas instâncias de *cache* com o de uma única instância de *cache* e o seu impacto no custo financeiro da implantação de tais instâncias. A implementação do *cache* deste trabalho baseia-se na proposta de união das tecnologias NFV e MEC definidas pelo ETSI.

As avaliações consistem em simulações variando as políticas de seleção de servidores *cache*, as capacidades de armazenamento total do *cache*, as formas de distribuição da capacidade de armazenamento e as políticas de substituição dos segmentos em cache. As políticas de seleção de servidores *cache* são baseadas em algoritmos round-robin, aleatório e a métrica de grau de centralidade e o objetivo deste artigo não é propor uma

nova política de seleção de servidores em particular. A estimativa de custo é obtido da calculadora de preços da Amazon Web Services para fins de comparação.

As topologias utilizadas nas avaliações são geradas sinteticamente usando parâmetros encontrados na literatura para representação de uma topologia real. A mobilidade dos usuários é gerada sinteticamente distribuída na área coberta pela topologia. Os resultados mostram que as diferentes políticas de seleção de servidores apresentam menos de 1% de diferença de desempenho entre si para o atraso médio de obtenção dos segmentos. A avaliação dos custos e da distribuição da capacidade de armazenamento do *cache* entre os servidores mostram que concentrar a capacidade de armazenamento do *cache* em um único servidor ao invés de distribuí-la pode reduzir o custo de manutenção impactando em menos de 3% o atraso médio.

A Seção 2 apresenta os trabalhos relacionados. Na Seção 3 são apresentadas as políticas usadas nas avaliações. A Seção 4 aborda todas as avaliações feitas neste artigo. E, por fim, a Seção 5 aborda as considerações finais e trabalhos futuros.

2. Trabalhos relacionados

Os trabalhos encontrados na literatura discutem o uso de diferentes métricas e políticas para seleção de servidores. Porém, até o momento, não são encontrados trabalhos que avaliem as políticas de seleção de servidores MEC para *cache*. Khan et al. propõem o Content-Based Centrality Metric (CBC) para a computação em névoa [Khan et al. 2017]. O CBC de cada nó é usado para priorizar alguns nós sobre outros para armazenar o conteúdo em cache. Esta proposta calcula o grau de centralidade de cada nó com base no número de caminhos mais curtos desde os nós dos usuários até cada servidor de origem. Os autores também consideram um fator de replicação de conteúdo entre os nós da rede que determina a parcela do armazenamento que é comum em todos os servidores.

Liu et al. propõem uma solução para a seleção do local de implantação do servidor MEC [Liu et al. 2020]. O foco dos autores não é o *cache*, mas é citado o trabalho futuro para desenvolver uma solução para implantação e seleção de servidores MEC para vídeos de cache, considerando a localização, tamanho e popularidade dos vídeos de forma otimizada. Liu et al. consideram que os servidores MEC podem ser instanciados em uma unidade centralizada ou em uma unidade distribuída. As unidades distribuídas oferecem menores atrasos para os usuários, menor área de cobertura e custo maior. Já as unidades centralizadas oferecem maior área de cobertura e custo maior. Liu et al. mostram como é importante a relação entre a experiência de atraso do usuário e o custo dos servidores.

Zhu et al. introduzem um modelo para o problema de seleção de servidores MEC para executar instâncias de uma aplicação [Zhu and Huang 2017]. Este modelo considera a relação entre os custos de implantação, os requisitos de disponibilidade da aplicação baseando-se no tráfego entre aplicações, CPU e memória. A disponibilidade da aplicação é definida pelo número mínimo de instâncias executadas. Os autores desenvolvem um algoritmo heurístico que obtém uma solução aproximada. Este artigo é outro exemplo de trabalho que considera o custo de implantação para selecionar servidores MEC.

Ren et al. propõem um mecanismo de *cache* cooperativo e consciente da mobilidade, com o objetivo de conseguir a descarga de tráfego em redes de borda móvel [Ren et al. 2020]. O foco é o problema da colocação e substituição de segmentos de

conteúdo no cache. Para resolver o problema de substituição, os autores desenvolvem um algoritmo distribuído em cada servidor para encontrar o melhor segmento a ser substituído. O algoritmo consiste em avaliar todas as opções de substituição de segmentos e encontrar a substituição que resulta no maior ganho.

Huang et al propõem uma estratégia para substituir os segmentos em *cache* [Huang et al. 2020]. Os segmentos são classificados de acordo com a popularidade e a importância. O *cache* é dividido em três categorias: (a) armazenamento para os primeiros 15% dos 20% de conteúdo mais popular; (b) segmentos restantes dos 20% de conteúdo mais popular; (c) segmentos restantes. Para cada categoria é atribuída uma estratégia de substituição, sendo menos frequente substituições na categoria (a) do que em (b) e (c).

Até o momento, não é encontrado na literatura nenhum estudo focado na avaliação de diferentes políticas de seleção de servidores *cache* de conteúdo de vídeo no MEC. O objetivo deste trabalho é mostrar que a capacidade de armazenamento tem impacto maior do que as políticas de seleção de servidores *cache* e que concentrar a capacidade de armazenamento em um servidor reduz o custo mantendo a desempenho do cache.

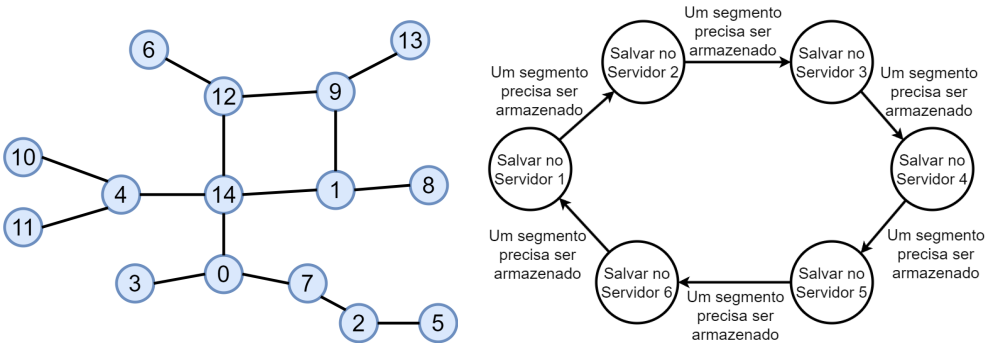
3. Políticas de seleção de servidores *cache*

A política de seleção de servidor *cache* é executada quando um segmento deve ser armazenado nos servidores MEC. Cada política de seleção de servidor *cache* pode utilizar diferentes métricas e algoritmos para determinar qual servidor da topologia armazenará um segmento. O objetivo deste trabalho é mostrar que a capacidade de armazenamento influencia mais na desempenho do *cache* do que o uso de diferentes políticas de seleção de servidores *cache*. Para isso, é o suficiente o uso de políticas que selecionem sequências de servidores diferentes para os mesmos cenários. Como o objetivo não é propor uma nova política de seleção de servidor cache em particular, as políticas usadas nas avaliações são baseadas em algoritmos e métricas comumente encontrados na literatura.

Uma das políticas de seleção de servidores *cache* avaliadas é baseada nas métricas de centralidade. A centralidade é um conceito muito abordado na literatura de teoria de grafos e análise de redes cujo o objetivo é indicar a importância de cada vértice no grafo. Existem diferentes formas de determinar a centralidade de um vértice. Uma das formas é a grau de centralidade. A grau de centralidade baseia-se no número de arestas do vértice. Quanto maior o número de arestas do vértice maior o valor do grau de centralidade.

A Figura 1(a) mostra um exemplo de grafo cíclico não-direcionado com 15 nós. O vértice 14 deste grafo possui quatro nós vizinhos e não há outro vértice com número de vizinhos maior. Por esse motivo, o vértice 14 é o que possui o maior grau de centralidade no grafo. Em contrapartida, os vértices folha 3, 5, 6, 8, 10, 11 e 13 possuem o menor grau de centralidade. No contexto da política de seleção de servidor cache, o grau de centralidade dos nós pode ser usado para definir o nó servidor mais importante para armazenar conteúdos em *cache*, sendo mais importante os nós com maior grau de centralidade.

Um algoritmo muito encontrado na literatura é o round-robin, comumente aplicado em problemas de balanceamento de carga. Para seleção de servidores *cache* este algoritmo pode ser usado para distribuir equilibradamente os segmentos entre os servidores *cache*. A Figura 1(b) mostra o funcionamento do algoritmo round-robin na política de seleção de servidores *cache* em uma topologia com 6 servidores. Quando um segmento



(a) Grafo acíclico não-direcionado.

(b) Seleção de servidores cache usando round-robin.

Figura 1. Funcionamento das políticas de seleção de servidores *cache*.

precisa ser armazenado em *cache*, um servidor é escolhido seguindo uma sequência. Quando o último servidor da sequência é escolhido, a política retorna ao Servidor 1.

Buscando variar a seleção de servidores *cache*, a política chamada grau de centralidade espalhado também é implementada. Inicialmente, esta política ordena os servidores da topologia do maior para o menor grau de centralidade. À medida que novos segmentos precisem ser armazenados em *cache*, os servidores são ocupados sempre passando para o próximo quando o servidor atual está cheio. Quando todos os servidores estão cheios, o algoritmo round-robin é usado para selecionar em qual servidor ocorre a substituição dos segmentos já armazenados pelo novo segmento. Como os critérios de seleção das políticas avaliadas são diferentes, as sequências de servidores selecionados também tendem a diferir o suficiente para a avaliação do impacto das políticas no desempenho do *cache*. Por fim, também é avaliada a política de seleção de servidores aleatória.

As políticas de seleção de servidores *cache* são avaliadas variando 3 políticas de substituição de segmentos no cache: aleatório, *Least Recently Used* (LRU) e popularidade. A política de substituição aleatória escolhe aleatoriamente segmentos no *cache* para serem removidos até que haja espaço disponível suficiente para o novo segmento ser armazenado. A política LRU substitui o segmento menos acessado no *cache* pelo novo segmento. Já política popularidade substitui o segmento de conteúdos de menor popularidade até que seja possível inserir o novo segmento. Se todos os segmentos do *cache* são de conteúdos mais populares que o novo segmento, o novo segmento não é inserido.

Para simular a duplicação de conteúdos em *cache*, é aplicada em todas as avaliações uma política de cópia de segmentos. Esta política consiste em copiar um segmento requisitado para o melhor servidor para usuário quando o servidor que atende a requisição é diferente do melhor servidor.

4. Avaliações

A Seção 4.1 apresenta a metodologia das avaliações. A Seção 4.2 aborda o impacto das políticas de seleção de servidores *cache* na eficiência do *cache*, a Seção 4.3 aborda o impacto da capacidade de armazenamento e a Seção 4.4 aborda impacto da distribuição da capacidade de armazenamento.

Número de servidores	2	3	4	5	6	7	8
Custo (USD)	99,88	149,82	199,76	249,70	299,64	349,58	399,52

Tabela 1. Custo dos servidores obtidos da calculadora da Amazon.

4.1. Metodologia

Um conjunto de avaliações é feito para mostrar que o impacto da capacidade de armazenamento no desempenho do *cache* é maior do que o impacto das políticas de seleção de servidores *cache*. As avaliações consistem em simulações realizadas em um computador equipado com CPU Intel Core i9-11900K e 128GB de memória no Laboratório Midiacom da Universidade Federal Fluminense. O simulador é desenvolvido na linguagem Python e reproduz o funcionamento do *cache* na borda das redes móveis e as políticas de seleção de servidores *cache* apresentadas na Seção 3.

As avaliações são divididas em três partes. A primeira avaliação visa compreender o impacto das políticas de seleção de servidores no desempenho do *cache* executando as políticas round-robin, aleatória e grau de centralidade espalhado em ambiente simulado. A segunda avaliação busca compreender o impacto da capacidade de armazenamento no desempenho do *cache*. Para isso, são feitas simulações variando o número de servidores *cache* usados pela política round-robin. A terceira avaliação aborda a diferença entre concentrar e distribuir a capacidade de armazenamento nos servidores associando ao custo.

Nas três avaliações o desempenho do *cache* é avaliado pelas métricas de acerto médio do *cache* e a redução do atraso médio para a obtenção do segmento. A redução do atraso médio é calculada como a porcentagem de redução com relação ao atraso da Internet. Quanto maior o acerto médio e a redução do atraso médio, maior é o desempenho do *cache*. Essas métricas são apresentadas no eixo y dos gráficos de resultados enquanto o eixo x apresenta as políticas de seleção de servidores ou cenários avaliados. As políticas de substituição de segmentos são identificadas por cor conforme as legendas.

A avaliações também abordam os custos dos servidores apresentados na Tabela 1. São considerados no mínimo 2 servidores pois 1 é o nó de controle e outro instancia o *cache*. Os valores dos custos são obtidos pela calculadora da Amazon Web Services e são apenas uma referência para a avaliação. Cada servidor é considerado como uma instância EC2 a1.xlarge equipada com 4 vCPUs, 8 GB de memória e 30 GB de armazenamento. O custo de 1 servidor é de USD 49,69/mês.

Para as avaliações são geradas sinteticamente 3 topologias modeladas como grafos cujo os nós de rede são os vértices e os enlaces da rede são as arestas. As 3 topologias são compostas por 20 nós, sendo 4 estações-base fixas, 8 servidores e 8 encaminhadores. Os nós servidores e encaminhadores são representados por um grafos de Waxman gerados usando os parâmetros alfa=0,1, beta=0,4 e L=70. O valor de alfa determina a incidência de arestas entre os nós mais distantes. Beta define a densidade das arestas e L define a distância entre os nós. As estações-base são criadas com posições obtidas aleatoriamente do *dataset* OpenCellID e interligada com o nó com menor distância euclidiana. A largura de banda dos enlaces é definida na geração da topologia em função de seu comprimento [Garcia-Saavedra et al. 2018].

Atraso da Internet	Número de rodadas	Número de slots	Tamanho dos segmentos	Capacidade do cache	Número de usuários
500 ms	10	5000	1 MB	100 MB	99

Tabela 2. Tabela dos parâmetros usados nos experimentos.

Os experimentos usam um *trace* de mobilidade sintético composto por 99 usuários. Cada usuário possui 5000 posições consecutivas geradas aleatoriamente dentro de uma área definida por um intervalo de latitude e longitude. O intervalo é definido para que os usuários se movimentem na mesma área em que os elementos das topologias. O número de 5000 posições é motivado pelos 5000 instantes de tempos discretos que são considerados nos experimentos, sendo uma posição para cada instante de tempo.

4.2. O impacto de diferentes políticas de seleção de servidores *cache*.

Esta seção aborda os experimentos que usam todos os servidores da topologia. Este é o cenário com o maior custo para manter os servidores funcionando. De acordo com a Tabela 1, o custo para manutenção dos servidores é de USD 399,52. As política de seleção de servidores *cache* usadas são aleatório, round-robin e grau de centralidade espalhado.

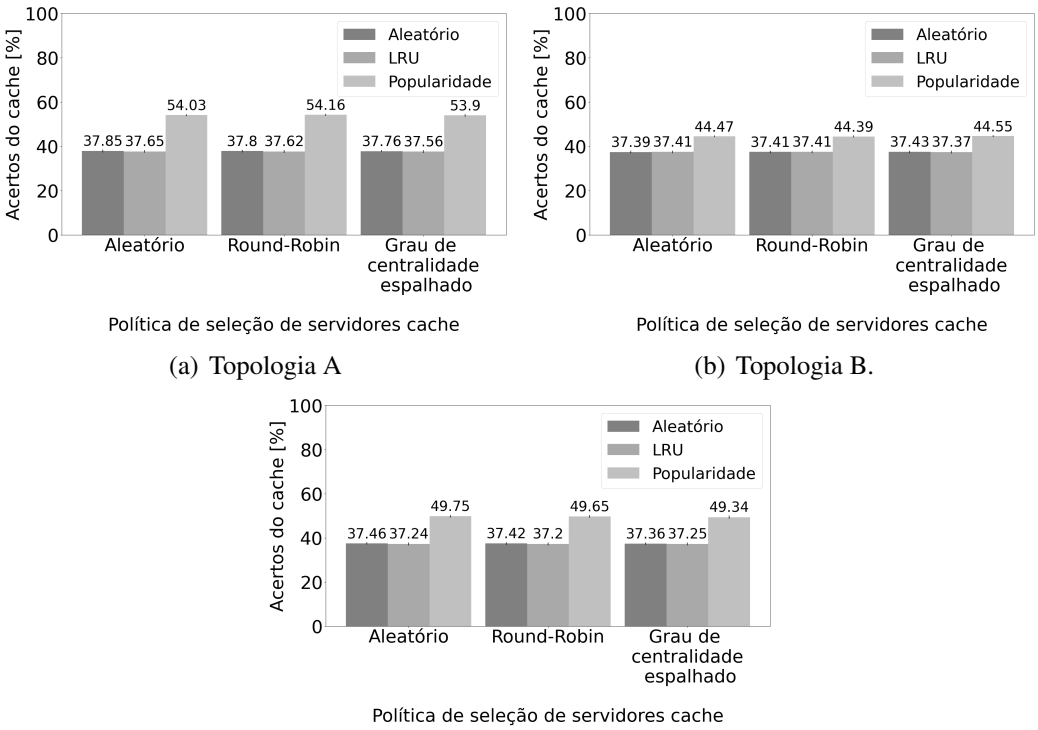
A Tabela 2 mostra os parâmetros considerados no experimento. O atraso considerado para obtenção dos segmentos da Internet é 500 ms, valor escolhido por ser bem maior do que os atrasos de obtenção de segmentos do *cache*. São executadas 10 rodadas de cada experimento variando o instante de tempo em que as requisições acontecem. É determinado empiricamente 5000 instantes de tempo para os experimentos. Cada conteúdo é dividido em segmentos com 1 MB de tamanho e cada servidor MEC tem a capacidade de armazenar 100 MB em *cache*. O trace de mobilidade é composto por 99 usuários.

As Figuras 2(a), 2(b) e 2(c) mostram o acerto médio do cache para as Topologias A, B e C, respectivamente. É possível observar que as políticas de seleção de servidores aleatório, round-robin e grau de centralidade espalhado obtém acertos médios muito próximos quando comparadas usando as mesmas políticas de substituição de conteúdos em *cache*. Esse resultado é obtido principalmente pois a capacidade total de armazenamento utilizada em *cache* é a mesma. A política de substituição de conteúdos baseada em popularidade obtém o acerto de *cache* maior, independente da política de seleção de servidores. Isso acontece pois os segmentos mais solicitados permanecem no servidores e aumentam a probabilidade das requisições serem atendidas pelo *cache*.

As Figuras 3(a), 3(b) e 3(c) mostram a redução do atraso médio para obtenção dos segmentos. As figuras mostram que a política de substituição Popularidade oferece melhor desempenho. É possível observar também que as políticas de seleção de servidores diferentes resultam em uma variação menor que 1% na redução do atraso médio. Isso ocorre pois os atrasos obtidos em cada salto na topologia são muito pequenos e tornam o impacto de salvar em diferentes servidores também pequeno. Portanto, conclui-se que a política de substituição dos segmentos tem maior impacto no desempenho do *cache* do que a política de seleção de servidores *cache*.

4.3. O impacto da capacidade de armazenamento no desempenho do *cache*.

O principal objetivo desta avaliação é mostrar o impacto da capacidade de armazenamento no desempenho do *cache*. A avaliação realizada nesta seção é feita executando a política



(a) Topologia A

(b) Topologia B.

(c) Topologia C.

Figura 2. Acerto médio do *cache* para cada topologia.

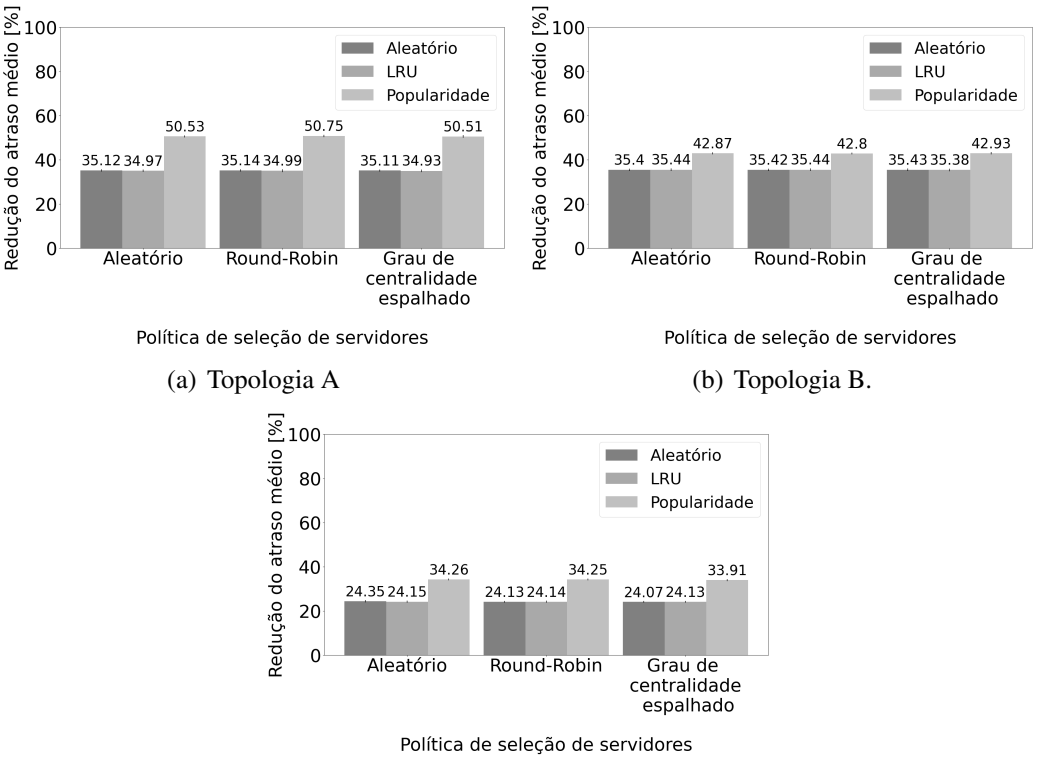
de seleção de servidor round-robin e variando o número de servidores. As mesmas Topologias A, B e C são usadas. O mínimo de servidores considerados neste experimento são 2 e o máximo são 8 e a capacidade de armazenamento total em cache está em função deste número, isto é, quanto maior o número de servidores, maior é a capacidade de armazenamento total do *cache*. Os parâmetros da Tabela 2 também são usados neste experimento.

As Figuras 4(a), 4(b) e 4(c) mostram os acertos médios em *cache* usando a política round-robin com diferentes números de servidores para as Topologias A, B e C, respectivamente. É possível observar que quanto maior o número de servidores usados nas topologias, maior a porcentagem de acerto para número maior de servidores usados. Assim como as demais avaliações, a política de substituição dos segmentos popularidade apresenta maior de acerto médio do que as políticas aleatório e LRU. Já as Figuras 5(a), 5(b) e 5(c) mostram as porcentagens de redução do atraso médio. Conforme o esperado, a redução do atraso médio aumenta à medida que mais servidores MEC são usados pois, quanto maior o número de servidores, mais segmentos podem ser armazenados em cache.

4.4. O impacto da distribuição da capacidade de armazenamento entre os servidores *cache*.

Este experimento é feito comparando três cenários. O Cenário A de uso considera o uso de todos os servidores da topologia com a mesma capacidade de armazenamento e efetua a política de cópia dos segmentos. O Cenário B considera todos os servidores da topologia com a mesma capacidade de armazenamento, porém não efetua a cópia dos segmentos. O Cenário C considera apenas um servidor da topologia com a capacidade total do *cache*.

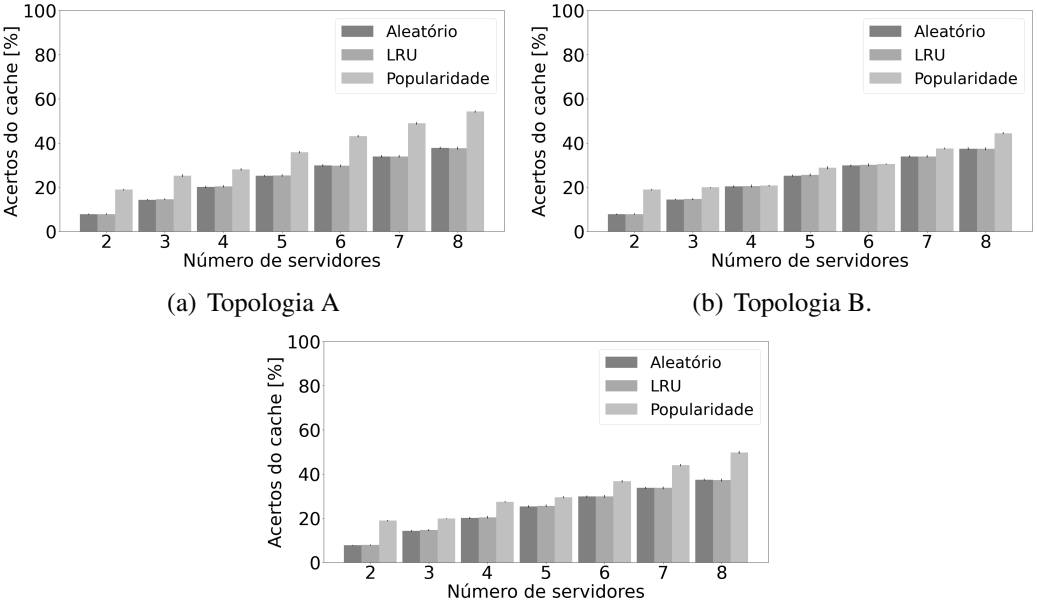
A Tabela 3 mostra a diferença dos custos entre os cenários A, B e C. O custo dos



(a) Topologia A (b) Topologia B.

(c) Topologia C.

Figura 3. Redução do atraso médio do *cache* para cada topologia.

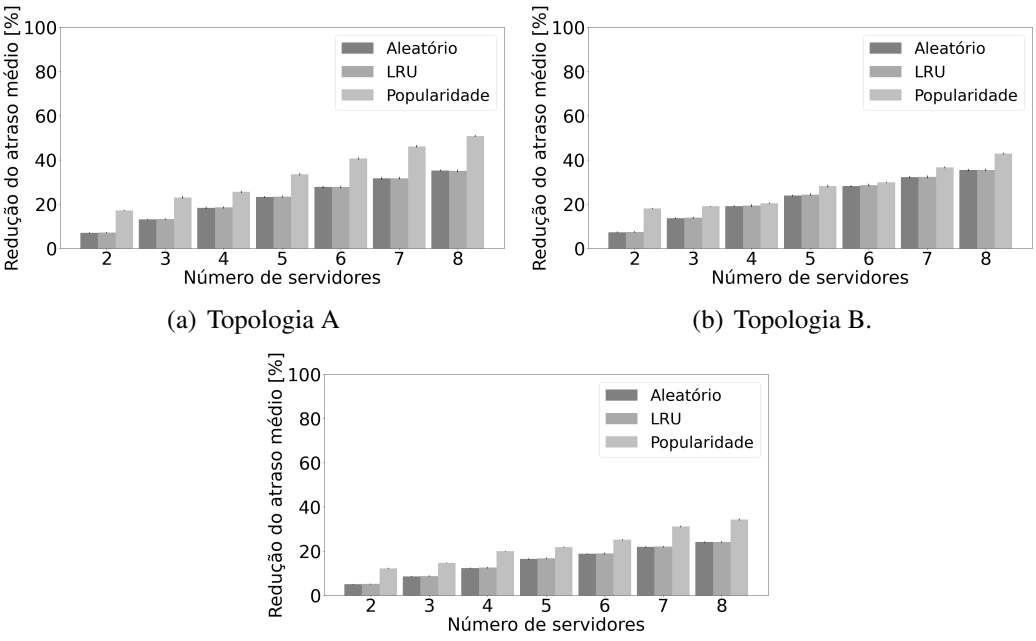


(a) Topologia A (b) Topologia B.

(c) Topologia C.

Figura 4. Acerto médio do *cache* para cada topologia.

Cenários A e B os servidores possuem as mesmas configurações da Tabela 2 e, por esse motivo, os custos são os mesmos do mostrado para o uso dos 8 servidores na Tabela 1. Já para o cenário C, é considerada a mesma instância da Amazon EC2, com a mesma



(a) Topologia A

(b) Topologia B.

(c) Topologia C.

Figura 5. Redução do atraso médio do *cache* para cada topologia.

capacidade de CPU, Memória e Rede dos Cenários A e B, porém com a capacidade de armazenamento 7 vezes maior, ou seja 700 MB. Isso é considerado pois, dos 8 servidores totais da topologia, 7 atuam como *cache* e 1 como nó de controle.

Cenário	Armazenamento	USD/mês
A	8 servidores com 100MB de *cache* e com cópia habilitada	399,52
B	8 servidores com 100MB de *cache* e com cópia desabilitada	399,52
C	1 servidor aleatório com 700 MB de *cache* + 1 nó de controle	117,88

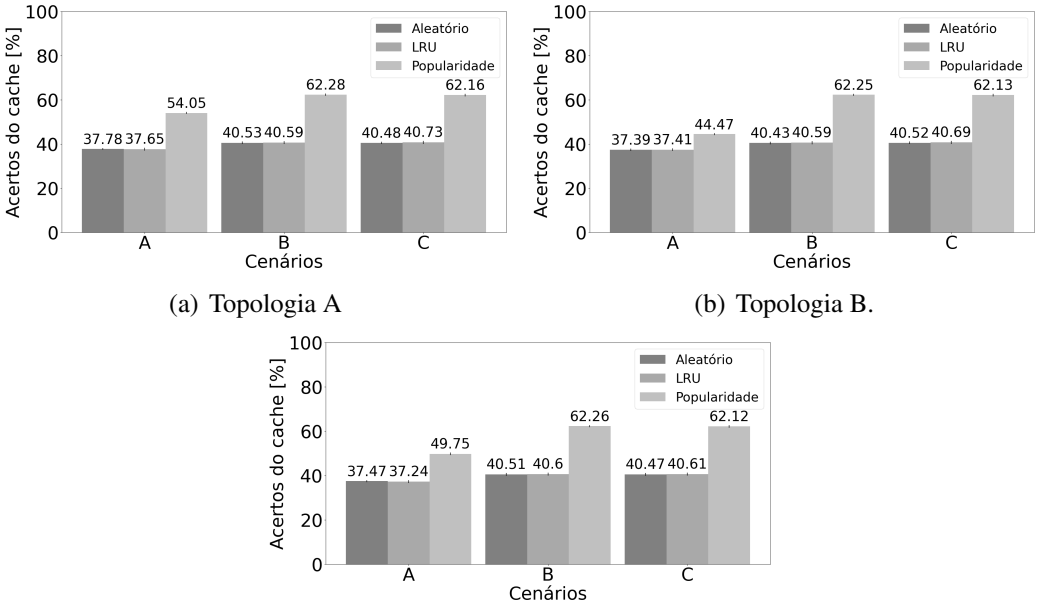
Tabela 3. Custo dos servidores por cenário.

As Figuras 6(a), 6(b) e 6(c) mostram os acertos médios do *cache* para as Topologias A, B e C, respectivamente. A diferença do acerto médio entre os cenários nas três topologias se mantém menor que 3% na maioria dos resultados. Já as Figuras 7(a), 7(b), 7(c) mostram a redução do atraso médio para a obtenção dos segmentos nas Topologias A, B e C. Os gráficos de redução do atraso médio para a obtenção dos segmentos também mostram diferença menor que 3% entre os Cenários B e C.

Esta avaliação mostra novamente que selecionar diferentes servidores para salvar os segmentos resulta em diferenças muito pequenas no atraso médio devido aos pequenos atrasos resultantes cada salto nos enlaces. Sendo assim, é possível reduzir muito os custos de manutenção de servidores *cache* ao concentrar toda a capacidade de armazenamento *cache* em um único servidor sem que haja perda de desempenho do *cache*.

5. Conclusão

Este artigo avalia por meio de simulações o impacto das políticas de seleção de servidores *cache*. As métricas utilizadas são a taxa de acerto médio, o atraso médio para obtenção

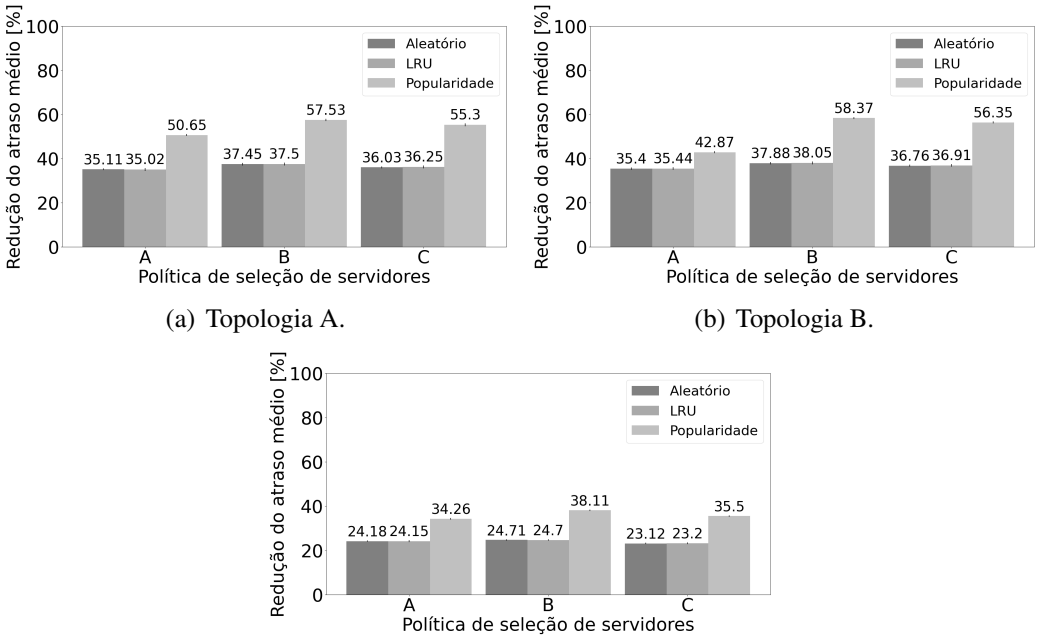


(a) Topologia A

(b) Topologia B.

(c) Topologia C.

Figura 6. Acerto médio do *cache* para cada topologia.



(a) Topologia A.

(b) Topologia B.

(c) Topologia C.

Figura 7. Redução do atraso médio do *cache*.

dos segmentos e o custo. Os resultados mostram que as diferentes políticas de seleção de servidores *cache* apresentam menos de 1% de diferença de desempenho entre si para o atraso médio de obtenção dos segmentos. Em contrapartida, ao aumentar capacidade de armazenamento em 7 vezes pode ocorrer a redução de 36% do atraso médio para a obtenção dos segmentos. Além disso, este artigo mostra que concentrar a capacidade de armazenamento em um único servidor pode reduzir em até 70% os custos enquanto o

impacto no desempenho do *cache* é menor que 3%. Para trabalhos futuros, pretende-se modelar a limitação do raio de alcance das estações base e o congestionamento através das filas nos enlaces.

6. Agradecimentos

Os autores agradecem a FAPERJ pelo suporte financeiro.

Referências

Cisco, U. (2021). Cisco annual Internet report (2018–2023) white paper. 2020. *Acessado em 24 de fevereiro de 2022*, 10(01).

de Souza, G. and Duarte, E. (2020). Uma arquitetura de alta disponibilidade para funções virtualizadas de rede. In *Anais do XXXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 407–420, Porto Alegre, RS, Brasil. SBC.

Dolui, K. and Datta, S. K. (2017). Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing. In *2017 Global Internet of Things Summit (GIoTS)*, pages 1–6.

Garcia-Saavedra, A., Salvat, J. X., Li, X., and Costa-Perez, X. (2018). Wizhaul: On the centralization degree of cloud ran next generation fronthaul. *IEEE Transactions on Mobile Computing*, 17(10):2452–2466.

Ge, X., Zhou, R., and Li, Q. (2020). 5g nfv-based tactile internet for mission-critical iot services. *IEEE Internet of Things Journal*, 7(7):6150–6163.

Hu, Y. C., Patel, M., Sabella, D., Sprecher, N., and Young, V. (2015). Mobile edge computing—a key technology towards 5G. *ETSI white paper*, 11(11):1–16.

Huang, X., He, L., Chen, X., Liu, G., and Li, F. (2020). A more refined mobile edge cache replacement scheme for adaptive video streaming with mutual cooperation in multi-mec servers. In *2020 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6.

Khan, J. A., Westphal, C., and Ghamri-Doudane, Y. (2017). A content-based centrality metric for collaborative caching in information-centric fogs. In *2017 IFIP Networking Conference (IFIP Networking) and Workshops*, pages 1–6.

Liu, Z., Zhang, J., Li, Y., and Ji, Y. (2020). Hierarchical mec servers deployment and user-mec server association in c-rans over wdm ring networks. *Sensors*, 20(5).

Ravache, G. (2021). Globoplay abandona TVs antigas e se alia ao Google para não 'travar' no BBB. url: https://www.uol.com.br/splash/colunas/guilherme-ravache/2021/12/12/globoplay-abandona-tvs-antigas-e-se-alia-ao-google-para-nao-travar-no-bbb.htm.

Ren, D., Gui, X., Zhang, K., and Wu, J. (2020). Mobility-aware traffic offloading via cooperative coded edge caching. *IEEE Access*, 8:43427–43442.

Zhu, H. and Huang, C. (2017). Availability-aware mobile edge application placement in 5g networks. In *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, pages 1–6.