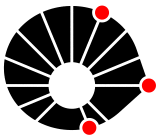
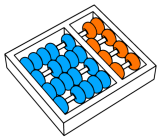




Gerenciamento de Recursos em sistemas distribuídos

A.C. Mestre *D. Charântola* *R. Zane* *L.F. Bittencourt*

Relatório Técnico - IC-PFG-19-10
Projeto Final de Graduação
2019 - Julho

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Gerenciamento de Recursos em sistemas distribuídos

Alexandre Conti Mestre[1], Danilo Charântola[2], Rafael Zane[3]

[1, 2, 3] Instituto de Computação Universidade Estadual de Campinas (UNICAMP), Caixa Postal 6176

13083-970 Campinas-SP, Brasil

ra154502@students.ic.unicamp.br

ra155124@students.ic.unicamp.br

ra157079@students.ic.unicamp.br

Resumo. Este trabalho tem como objetivo realizar estudos em sistemas distribuídos, mais especificamente, computação *fog*, caracterizada pela alocação de fontes de processamento descentralizadas perto da borda da rede. Para isso, foram realizadas simulações utilizando o *iFogSim*, ferramenta criada para modelar e simular o uso de recursos em computação nas nuvens e *fog*, que no decorrer do projeto teve sua implementação modificada e estendida para suportar simulações mais genéricas e complexas, listadas pelo criador da ferramenta como trabalhos a serem desenvolvidos no futuro.

As simulações realizadas contemplaram diversas políticas de alocação de aplicações em cloudlets (dispositivos *fog*) e nuvem, incluindo as suportadas por padrão no simulador e outras implementadas no decorrer do projeto. Essas últimas consideravam o tempo de comunicação entre os módulos da aplicação ou realizavam a alocação de cada módulo individualmente no iFogSim. Além disso, as diferentes políticas de alocação foram combinadas com arquiteturas distintas de disposição de cloudlets.

Ao final das simulações foi possível analisar as vantagens e desvantagens de cada configuração, levando em consideração o delay na execução de aplicações e o uso da rede, provendo uma melhor visão sobre o custo benefício da computação *fog* e quais as melhores políticas para determinado cenário de uso.

Palavras-Chave: *Fog computing, cloud computing, iFogSim, cloud, cloudlet, multi tier cloudlet hierarchy, edge computing,* políticas de alocação de recursos em computação distribuída

1. Introdução

Com o aumento de dispositivos mobile e a evolução do conceito de Internet of Things (IoT), cada vez mais dispositivos estão conectados à internet e esse número tende a aumentar de maneira expressiva. A maioria desses dispositivos executa aplicações que demandam que parte do processamento seja executado em datacenters com grande poder de processamento, conhecidos como cloud. Pelo fato da cloud estar fisicamente distante dos dispositivos de usuários finais, a divisão da execução de tarefas de uma aplicação entre a cloud e esses dispositivos gera um delay na comunicação. Esse delay na transmissão de informações é extremamente prejudicial para aplicações sensíveis a tempo de resposta.

Tendo em vista esse problema, o conceito de Fog Computing surge para tentar aproximar os dispositivos dos usuários finais ao dispositivo na rede que irá realizar parte do processamento. A proposta do Fog Computing se baseia na introdução de fog nodes ou cloudlets. Cloudlets são dispositivos que estão fisicamente próximos dos dispositivos dos usuários finais e vão atuar como a cloud, realizando parte do processamento das aplicações dos usuários. Dessa forma, um aparelho mobile que delegava parte do processamento de uma aplicação para a cloud passa a delegar esse processamento para uma cloudlet próxima, que será capaz de responder com muito menos delay.

Para atender a esse requisito de curta distância, o número de cloudlets se torna bem expressivo. Assim, para que seja viável a implantação desses dispositivos, eles possuem um poder de processamento muito menor em comparação com a cloud. Levando em conta essa limitação de processamento, se muitos dispositivos de usuários começarem a mandar tarefas para a cloudlet, eventualmente ela não será capaz de processar todas elas e terá que então direcionar essas tarefas para a cloud. Uma questão importante que surge é com relação a quais tarefas devem continuar rodando na cloudlet e quais devem ser mandadas para a cloud. Essa decisão depende da política de alocação seguida pela cloudlet. Outra questão importante é que, ao invés de delegar as tarefas para a cloud, a cloudlet poderia delegar para outra cloudlet presente um nível acima na hierarquia da rede, estando um pouco mais distante dos dispositivos dos usuários. A escolha de políticas de alocação e topologias da rede que sejam adequadas se tornam questões pertinentes para que os recursos sejam utilizados com eficácia e as aplicações tenham suas necessidades atendidas.

O objetivo desse trabalho é realizar o estudo e avaliação de políticas de alocação nas cloudlets e de diferentes topologias de dispositivos para múltiplos cenários com aplicações de diferentes características e em diversas quantidades. O foco da avaliação é no delay das aplicações gerado pela comunicação de seus módulos que estão sendo executados por diversos dispositivos da rede.

Para as políticas de alocação foram propostas duas novas abordagens. A primeira se baseia em tratar cada instância de aplicação de maneira isolada para que seja possível tomar uma decisão de alocação de módulos de uma única instância, não afetando módulos das demais instâncias daquela aplicação. A segunda abordagem se baseia na análise de comunicação entre os módulos de uma aplicação para escolher o conjunto de módulos que terá menor impacto ao ser realocado para um dispositivo de nível acima na hierarquia da rede. Para o estudo de arquiteturas de dispositivos, foi utilizado um modelo com um nível de cloudlets entre os dispositivos dos usuários finais e a cloud, e foi proposto um novo modelo com dois níveis de cloudlets.

Para realizar os estudos foi utilizado o toolkit iFogSim, que possibilita realizar simulações para diferentes topologias de rede, com diferentes políticas de alocação e possibilitando escolher as características das aplicações sendo executadas.

2. Trabalhos Relacionados

Em fog computing, os dispositivos são heterogêneos e apresentam recursos bastante variados. Assim, a definição da maneira como eles são organizados em rede, bem como a forma como aplicações serão distribuídas entre eles, impacta diretamente um grande número de variáveis. Isso, somado à grande diversidade de aplicações existentes e seus requisitos distintos, se torna um problema bastante complexo a ser resolvido. Por conta disso, um grande número de trabalhos têm sido apresentados, levando em consideração diferentes aspectos.

Lin et al. propõem, em [5], um sistema chamado Cloud Fog para utilizar os recursos de fog computing visando melhorar a qualidade da experiência (QoE) de usuários de jogos multiplayer online, levando em consideração o tempo de resposta, a cobertura do serviço, o consumo e o congestionamento da rede.

Mahmud et al. apresentam, em [6], uma política de escalonamento levando em consideração o nível de bateria dos dispositivos, o sinal de internet e a qualidade de serviço da aplicação (QoS). Dessa forma, obtêm um aumento do número de requisições com sucesso, ao fazer com que mais usuários sejam atendidos antes que a comunicação seja interrompida pela má conexão ou falta de bateria. Para simulação e coleta dos resultados, o toolkit CloudSim foi utilizado.

Gupta et al. sugerem, em [3], um simulador chamado iFogSim desenvolvido em cima do toolkit CloudSim com o intuito de facilitar a medição de diferentes organizações hierárquicas de rede e políticas de alocação, considerando diversos fatores. Este simulador é, portanto, uma ferramenta bastante útil para estudos de fog computing, sendo utilizado em diversos trabalhos, incluindo este. Por conta disso, é descrito em mais detalhes posteriormente. Além disso, Mahmud et al. realizam um estudo mais aprofundado dessa ferramenta em [2].

Utilizando o iFogSim, Taneja et al. propõem e analisam, em [7], uma política de alocação na qual os módulos de uma aplicação são alocados para o dispositivo com menor capacidade computacional capaz de processá-lo. Para isso, os recursos considerados são: CPU, RAM e largura de banda disponíveis.

Outras políticas utilizando o iFogSim são sugeridas por Mahmud et al. em [8], [4] e [9], considerando a latência, o consumo de energia e a qualidade de experiência, respectivamente. Enquanto isso, Skarlat et al. propõem em [10] uma solução com base em métricas de qualidade de serviço (QoS).

E, ainda, Bittencourt et al. estudam, em [1], as estratégias de escalonamento: concorrente, FCFS (first come, first served) e delay-priority, também utilizando o iFogSim para as medições, e considerando cenários nos quais aplicações começam a migrar para um mesmo cloudlet.

3. Aplicações de estudo

As aplicações escolhidas para serem estudadas e executadas na avaliação das políticas e hierarquias propostas, foram retiradas de trabalhos anteriores de Bittencourt et. al[1] e Gupta et. al [3].

3.1. Video Surveillance Object Tracking

Esta aplicação consiste em um conjunto de câmeras inteligentes operando de maneira distribuída que atuam como sensores enviando as informações visuais para os módulos da aplicação para que seja possível detectar e rastrear objetos. Uma vez rastreados os objetos, é feito o cálculo de qual a melhor posição horizontal e vertical além da intensidade de zoom nas câmeras para uma cobertura mais efetiva dos objetos rastreados.

Esses dados calculados são enviados periodicamente de volta às câmeras para que possam ser ajustadas. Essa aplicação é de interesse pois é sensível à latência, uma vez que as câmeras precisam ser ajustadas em tempo real, além de que uma grande quantidade de quadros de vídeo precisam ser transferidos.

A aplicação conta com 5 módulos, que são descritos abaixo:

1) Detector de movimentos: esse módulo está presente dentro da câmera e recebe os dados de vídeo capturados. O módulo detecta e filtra, dentre esses dados, aqueles em que alguma movimentação está presente. Em seguida, passa os dados filtrados de movimentação para o módulo de detecção de objeto.
2) Detector de objetos: esse módulo tem a responsabilidade de tentar detectar objetos nos dados de vídeo em que foram detectados sinais de movimentação. Os dados da identificação e as informações de posição dos objetos são passados para o módulo de rastreamento de objetos.
3) Rastreador de objetos: nesse módulo são utilizados os dados das posições dos objetos detectados para que a melhor posição (horizontal e vertical) e zoom da câmera possam ser calculados. Essa informações são passadas para o controlador de PTZ.
4) Controlador PTZ (Pan, Tilt e Zoom): esse módulo está presente em cada uma das câmeras e é responsável por movimentá-las de acordo com as informações passadas pelo módulo de rastreamento para que a câmera esteja posicionada de maneira efetiva no rastreamento dos objetos.
5) Interface do usuário: apresenta ao usuário uma parcela dos dados de vídeo filtrados pelos módulo de detecção de movimento e de objetos.

Na figura 1 é mostrado o diagrama dos módulos e arestas que representam a comunicação entre eles. Abaixo do número de cada aresta está representada sua direção (Up

ou Down) e a quantidade de bytes transmitidos ao longo da aresta. A aresta 3 possui seletividade 0.05 indicando que essa é a proporção com que são geradas essas arestas em comparação com a quantidade de arestas 2 que chegam no módulo Detector de objetos. A aresta 5 é a única periódica dessa aplicação, sendo gerada a cada 100 ms.

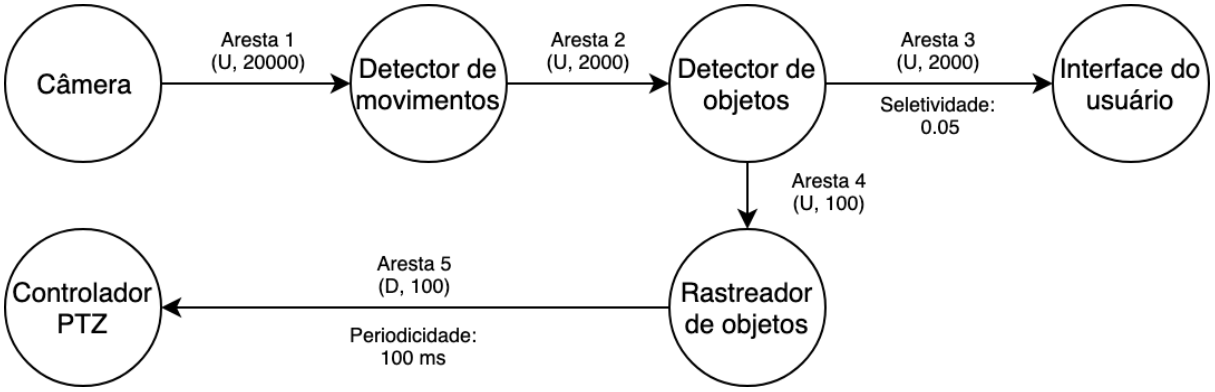


Figura 1: diagrama dos módulos e arestas que representam a comunicação entre eles para a aplicação VSOT.

3.2. VR game

Essa aplicação consiste em um jogo de celular. Nesse jogo, os usuários usam um headset que se comunica com o celular enviando medidas do nível de concentração do jogador em tempo real. O desafio do jogo consiste nos jogadores tentarem atrair um alvo para perto de si, e a força com o qual um jogador atrai o alvo está relacionada com o seu nível de concentração.

A aplicação precisa saber as informações de outros jogadores em nível global, por isso, é interessante que ela seja executada na cloud. No entanto, a aplicação também precisa receber em tempo real os dados do headset e, para evitar queda no desempenho, seria melhor se ela fosse executada no celular do usuário. Devido a esses requisitos conflitantes, essa aplicação se torna interessante para ser estudada em um modelo de fog computing com módulos distribuídos em múltiplos dispositivos.

Os módulos da aplicação são:

1) Sensor de concentração: esse módulo é responsável por mandar os dados coletados de concentração do usuário para o módulo cliente sendo executado no celular.
2) Cliente: este módulo recebe os dados de concentração e checa por inconsistências. Em seguida, envia os dados para o módulo de cálculo de concentração e espera sua resposta com o nível de concentração do jogador. Além disso, esse módulo recebe os níveis de outros jogadores do módulo conector. Tanto o nível do jogador local, quanto os níveis dos outros usuários globais são transmitidos para o módulo de exibição.
3) Calculador de concentração: este módulo é responsável por calcular o nível de concentração do usuário de acordo com os sinais captados pelo headset e enviados pelo cliente. Retorna para o cliente o nível de concentração calculado.
4) Conector: este módulo coordena os múltiplos jogadores recebendo os níveis de concentração de todos os usuários conectados e informa, para os clientes sendo executados por cada usuário, os níveis de concentração globais. Dessa forma, cada cliente consegue atualizar o módulo de exibição.
5) Exibição: este módulo recebe os níveis de concentração do usuário local e dos usuários globais para exibir corretamente para o jogador o estado atual em que o jogo se encontra.

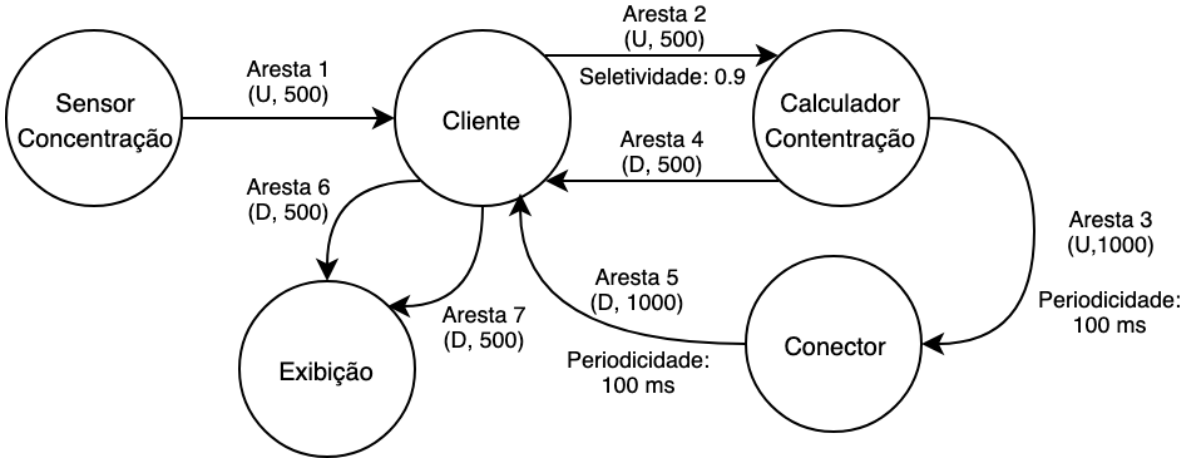


Figura 2: diagrama dos módulos e arestas que representam a comunicação entre eles para a aplicação VRGame.

Na figura 2, seguindo o mesmo modelo do diagrama da figura 1, estão representados os módulos do VRGame, assim como as arestas com suas respectivas direções e quantidade

de bytes transmitidas. A aresta 2 possui seletividade 0.9 indicando que essa é a proporção com que são geradas essas arestas em comparação com a quantidade de arestas 1 que chegam no módulo Cliente. As arestas 3 e 5 são geradas periodicamente com intervalo de 100 ms.

4. Topologias de estudo

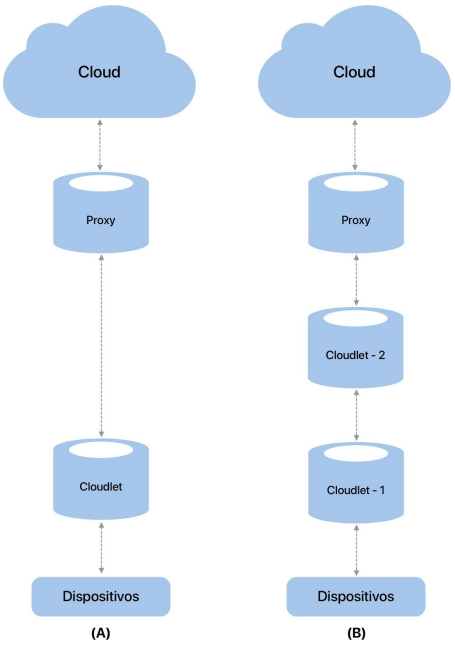


Figura 3: Topologias de redes *fog*

4.1. Topologia com um nível

A topologia de um único nível é composta por somente uma camada de cloudlets servindo os dispositivos móveis, conforme ilustrado na Figura 3 - A. Assim, quando a capacidade de uma cloudlet é totalmente preenchida, alguns módulos são enviados para a nuvem, que possui maior capacidade de processamento.

Mesmo no contexto de um único nível de cloudlets, ainda é possível construir arquiteturas bastantes distintas, a depender da necessidade que a infraestrutura deve suprir. É possível, por exemplo, estruturar a rede com muitas cloudlets de baixa capacidade, mas que estejam relativamente perto dos usuário, resultando em baixa latência na comunicação ou escolher uma arquitetura com um menor número de cloudlets, mas que tenham maior poder de processamento.

4.2. Topologia multinível

Em uma topologia multinível a rede é construída com várias camadas de cloudlets entre o dispositivo e a nuvem (Figura 3 - B). Assim, caso a capacidade máxima de processamento de uma cloudlet seja atingida, os módulos são movidos para a cloudlet da camada imediatamente acima.

Em geral, é esperado que a comunicação entre as camadas de cloudlets seja mais rápida do que entre a cloudlet e a nuvem, garantindo menor latência na transferência de dados. Além disso, a capacidade de processamento das cloudlets mais distantes do usuário deve ser maior quando comparada com as cloudlets em camadas mais baixas.

O desafio no uso de topologias multinível está em balancear seu custo benefício. É necessário uma ampla análise do número e das características das aplicações executadas para decidir o número de camadas da estrutura e a capacidade de processamento de cada dispositivo, pois o risco de alguma cloudlet (ou até mesmo a nuvem) ficar ociosa é maior quando comparado com a topologia de nível único.

5. iFogSim

iFogSim é um toolkit que permite a modelagem e simulação de diferentes topologias de fog computing, coletando informações sobre latência, utilização de memória, capacidade computacional, consumo energético e congestionamento da rede. Por conta disso, é uma ferramenta muito útil para análise de diferentes níveis hierárquicos de dispositivos em uma rede e políticas de alocação de recursos, sendo vastamente utilizado na literatura para estudo de *fog computing*, conforme mencionado na seção 2.

O simulador pode ser dividido em três componentes distintos, sendo eles: físico, lógico e de gerenciamento. O primeiro consiste nos dispositivos da rede (sensores, atuadores, dispositivos móveis, cloudlets, cloud...) e sua organização hierárquica, bem como seus recursos computacionais e latência de comunicação entre si. O componente lógico, por sua vez, consiste na representação das aplicações que serão executadas. Tais aplicações são divididas em módulos que demandam certa capacidade computacional para serem executados e podem ser alocados em dispositivos distintos.

Entretanto, deve-se considerar as relações de interdependência entre os módulos, as quais são determinadas por *AppEdges* (arestas) que ligam o resultado de um módulo à entrada

de outro. Existem duas orientações possíveis para uma aresta, Up e Down. Se ela for do tipo Up então ela está enviando dados para que o próximo módulo processe. Se ela for do tipo Down então ela está enviando dados que já foram processados e o módulo de destino está aguardando a resposta. As arestas possuem, dentre outras informações, a quantidade de bytes sendo transmitidos de um módulo para outro. A transmissão pode ser periódica ou baseada em eventos, isto é, assim que os dados de uma aresta incidente em um módulo chegam, o módulo gera dados para serem enviados para a próxima aresta. Um ponto importante é que o simulador permite definir a seletividade com o qual um módulo recebe dados de uma aresta e gera dados para enviar ao longo da próxima aresta.

As medidas de tempo de execução das aplicações são feitas com base no delay observado em ciclos de comunicação entre os módulos. Ou seja, o tempo médio entre o início da execução do primeiro módulo do ciclo até o término do último módulo, passando pelos demais.

Por fim, o componente de gerenciamento é responsável por iniciar a execução dos módulos, coletar dados da simulação e alocar os módulos das aplicações entre os dispositivos da rede. Para isto, leva em consideração os recursos requeridos e disponíveis, a prioridade de cada aplicação e a latência de comunicação entre os dispositivos. Atualmente, o iFogSim conta com três políticas distintas para fazer essa distribuição dos módulos na rede: *Mapping*, *Cloud only* e *Edgewards*.

5.1. Políticas de alocação de módulos do iFogSim

5.1.1. Module Placement Mapping

Esta política de alocação dos módulos segue estritamente o mapeamento fornecido ao *Controller* do iFogSim no momento em que este é criado, independente dos recursos computacionais. Consequentemente, se determinado módulo de uma aplicação for mapeado para uma cloudlet específica, mas esta já estiver ocupada executando outras aplicações ou outras instâncias da mesma aplicação, e não houver recursos suficientes, este módulo permanecerá alocado para ela. Assim, terá que aguardar o término da execução dos demais para então ser executado.

5.1.2. Module Placement Cloud Only

A política de alocação anterior apresenta problemas caso muitos módulos sejam alocados para uma mesma cloudlet superando a capacidade por ela suportada, então esta política apresenta uma solução bastante simples para isso.

Como o nome sugere, ela aloca todos os módulos para a cloud, uma vez que é o dispositivo mais alto na hierarquia e possui um poder computacional muito maior que os demais, podendo ser considerada, para efeitos práticos, "ilimitada". Entretanto, ainda é possível que haja muitas aplicações e ela seja sobrecarregada. Tal cenário simula, portanto, o caso em que não há dispositivos fogs na rede e os usuários necessitam enviar os dados para a cloud.

Assim sendo, essa abordagem apresenta desvantagens uma vez que a cloud está muito mais distante dos dispositivos móveis do que os dispositivos intermediários da hierarquia. Consequentemente, a latência de comunicação é consideravelmente maior.

5.1.3. Module Placement Edgewards

A política de alocação edgewards combina as duas políticas anteriores, buscando resolver as desvantagens apresentadas por elas. Esta abordagem também tenta alocar os módulos nos devices de acordo com o mapeamento fornecido ao *Controller* do iFogSim, entretanto, antes de alocar um módulo, verifica se o dispositivo possui recursos suficientes para atender seus requisitos.

Caso o dispositivo não tenha capacidade suficiente, o módulo é movido para o dispositivo logo acima na hierarquia (dispositivo pai). Além disso, se o dispositivo filho já possuir outras instâncias do módulo alocadas para ele, estas também são movidas. E, ainda, se houver instâncias de módulos, no dispositivo filho, que dependem do resultado de um dos módulos que estão sendo movidos (tem como entrada o resultado de um módulo movido), essas instâncias também são movidas. E o processo se repete até não haver mais módulos a serem movidos.

Além disso, se o dispositivo pai não for capaz de atender todos os módulos que foram movidos para ele, o algoritmo se repete enviando os módulos ainda mais pra cima na

hierarquia até que eles possam ser alocados. Assim sendo, no pior caso possível, todos os módulos serão movidos para cima até chegar na cloud, e esta política terá o mesmo resultado que a alocação *Cloud only*.

6. Políticas de alocação de módulos propostas

Além das políticas de alocação de módulos já existentes no iFogSim, é possível imaginar outras abordagens. Neste trabalho, propomos novas soluções e realizamos uma análise comparativa do seu desempenho em termos do delay médio observado em cada aplicação e os diversos fatores que causam impactos sobre ele.

6.1. Individual Module Placement Edgewards

A primeira política proposta consiste em uma modificação da política *edgewards* já existente no simulador para considerar cada instância dos módulos de uma aplicação de forma isolada. Da mesma forma que a política *edgewards*, essa abordagem inicia tentando alocar os módulos de acordo com o mapeamento informado ao *Controller* e, antes de alocar uma instância de um módulo a um dispositivo, verifica se este é capaz de processá-lo. A diferença consiste no fato de que, caso o dispositivo não possua recursos suficientes, somente a nova instância do módulo será movida para um dispositivo mais alto na hierarquia. Qualquer instância do módulo que já tivesse sido previamente alocada para o dispositivo não será alterada. Além disso, se houver, no dispositivo, outro módulo cuja entrada depende do resultado do módulo movido, ele só será movido também se pertencer à mesma instância.

Em resumo, se houver dois usuários executando a mesma aplicação - por exemplo, dois usuários executando um VRGame - esta abordagem diferencia os módulos da aplicação de ambos, movendo-os de forma individual. Respeitando, porém, a dependência entre os módulos que pertencem a um mesmo usuário. Consequentemente, espera-se que essa abordagem mantenha mais módulos sendo executados em dispositivos mais baixos na hierarquia e, portanto, mais próximos do usuário. Logo, espera-se um delay menor nas aplicações.

Para implementar essa abordagem, além de desenvolver uma nova política de alocação, foi necessário modificar o simulador para que fosse possível diferenciar os módulos que pertencem a instâncias diferentes da aplicação.

Porque, conforme descrito por Gupta et al.,

> "iFogSim assume que um módulo *m* alocado em um dispositivo *d* irá lidar com todas as tuplas advindas de dispositivos ao sul de *d* na hierarquia [...]. Considere dispositivos *dp* e *dc*, em que *dp* é pai de *dc* na topologia e ambos possuem instâncias do mesmo módulo *m*. Nesse caso, qualquer tráfego incidente para o módulo *m* em *dc* seria processado aí e não seria enviado para cima." [3]

Primeiramente, os módulos da aplicação foram modificados para conter um identificador da instância a qual pertencem. Da mesma forma, esse identificador foi adicionado aos sensores, supondo-se assim que cada instância da aplicação está associada a um sensor diferente. Por exemplo, cada jogo de realidade virtual estaria associado a um headset (sensor) distinto.

Feito isso, foi modificado o processo de criação de tuplas - tarefas que são passadas de um módulo da aplicação para outro e executadas em um dispositivo - pelo sensor, para que a tupla também possuísse o identificador da instância a qual pertence.

E, por fim, a classe *FogDevice*, que representa um dispositivo, foi modificada para que, ao receber uma tupla, verificasse não somente se foi alocado para ele o módulo de destino da tupla (e portanto se é capaz de executá-la), mas também se a instância do módulo alocado para ele é a mesma que a instância da tupla. Em caso afirmativo, o dispositivo executa a tupla e, em caso negativo, a envia para um dispositivo mais alto na hierarquia, que deve ser capaz de executá-la.

Para exemplificar, é possível que exista dois usuários distintos da aplicação VRGame e que o módulo concentration-calculator do primeiro tenha sido alocado para a cloudlet, enquanto o do segundo não coube e foi alocado para a cloud. Assim, a cloudlet, ao receber uma tupla cujo destino é o módulo concentration-calculator, irá executá-la se ela pertencer ao primeiro usuário, mas enviá-la para a cloud caso contrário.

6.2. Communication Based Module Placement Edgewards

A segunda política criada também consiste em uma modificação da política *edgewards* já existente no simulador. A ideia dessa política é que a decisão de escolha de quais módulos devem ser movidos para um dispositivo superior na hierarquia seja tomada levando em conta a comunicação entre os módulos da aplicação.

A política segue o procedimento inicial da *edgewards* e a principal diferença introduzida pela política proposta está presente na situação em que um dispositivo não é capaz de processar um módulo que foi mapeado para ele. Nessa situação a *edgewards* move todas as instâncias do módulo que não pode ser alocado e outros módulos presentes no dispositivo cuja entrada dependa dos resultados de um módulo movido. Já a política proposta irá analisar qual o conjunto de módulos irá gerar menor impacto de comunicação ao ser movido para dispositivos superiores na hierarquia e então irá mover esse conjunto.

Para que seja possível calcular o acréscimo de comunicação num dispositivo causado pela movimentação de um conjunto de módulos, primeiramente deve ser esclarecido um aspecto importante no funcionamento do simulador. Quando uma aresta com módulo de origem *Mo* e destino *Md* vai ser executada por um dispositivo, primeiramente tenta-se executá-la no dispositivo no qual *Mo* está presente, no entanto, caso *Md* não esteja presente neste dispositivo, a aresta é transmitida para dispositivos superiores ou inferiores na hierarquia até que seja encontrado o dispositivo em que *Md* está. Essa transmissão da aresta gera um delay na simulação que é calculado como sendo a razão entre a quantidade de bytes da aresta e largura de banda entre os dispositivos em que a aresta está sendo movida e esse delay é somado ao delay base de comunicação entre esses dispositivos. Portanto, para evitar o aumento do delay na execução de uma aplicação, é importante que menos arestas transmitam dados entre dispositivos distantes e que as arestas que realizarem mais transmissões entre esses dispositivos não sejam as que transmitem a maior quantidade de bytes. Satisfazer estes requisitos está atrelado com a escolha dos dispositivos em que cada módulo será alocado.

Para que essa escolha seja a de menor impacto, a política baseada em comunicação primeiramente gera todos os conjuntos possíveis de módulos que podem ser movidos para um dispositivo superior. Inicialmente é gerado um conjunto para cada módulo presente no dispositivo que não seja origem de arestas do tipo Up em que o módulo destino esteja nesse mesmo dispositivo. Em seguida, para cada conjunto, são buscados módulos que são origem

de uma aresta cujo destino é o módulo presente no conjunto, então para cada módulo que satisfaça essa condição, é criado um novo conjunto com esse módulo e o módulo inicial do conjunto. Esse processo segue de forma iterativa até que nenhum novo conjunto seja criado.

Uma vez criado os conjuntos, é necessário calcular o impacto de comunicação de cada um deles, que se resume a soma do impacto de cada módulo presente. A métrica de impacto é somente uma forma proposta de se comparar os conjuntos de módulos candidatos a serem movidos utilizando algumas das informações de comunicação da aplicação e de alocação dos dispositivos. O impacto de comunicação de uma aresta é definido pela fórmula (1).

$$I = b * s * \frac{Ta}{p} \quad (1)$$

Onde b representa a quantidade de bytes da aresta, s sua seletividade, Ta a periodicidade padrão com o qual as arestas da aplicação são transmitidas e p a periodicidade da aresta. Caso uma aresta não seja periódica, tem-se que p tem o mesmo valor que Ta. O impacto de comunicação para um módulo é calculado da seguinte forma:

Para cada aresta da aplicação:

1) Se o módulo for origem da aresta:
    a) Se nenhum dispositivo superior na hierarquia possui o módulo destino da aresta:
        i) Se o módulo destino da aresta não está no conjunto então o impacto de comunicação da aresta é somado ao impacto total do módulo.
    b) Se um dispositivo superior na hierarquia possui o módulo destino da aresta então impacto de comunicação da aresta é subtraído do impacto total do módulo.
2) Se o módulo for destino da aresta:
    a) Se nenhum dispositivo superior na hierarquia possui o módulo origem da aresta:
        i) Se o módulo origem da aresta não está no conjunto então o impacto de comunicação da aresta é somado ao impacto total do módulo.
    b) Se um dispositivo superior na hierarquia possui o módulo origem da aresta então o impacto de comunicação da aresta é subtraído do impacto total do módulo.

A seguir é apresentado um exemplo de criação de conjuntos e cálculo de impacto de comunicação para uma configuração de módulos apresentada na figura 4.

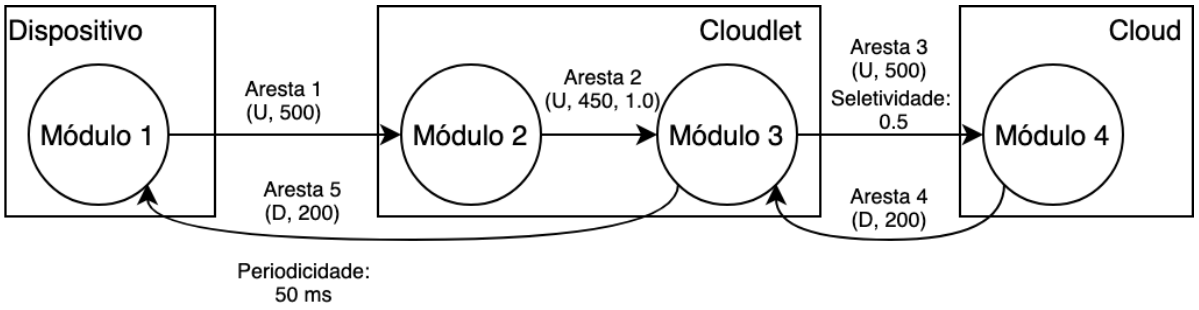


Figura 4: diagrama dos módulos e arestas que representam a comunicação entre eles para uma aplicação de exemplo.

Para este cenário é considerando que a política de alocação baseada em comunicação está tentando realocar os módulos da cloudlet. As informações presentes nas arestas indicam respectivamente orientação (Up ou Down) e quantidade de bytes. Será considerado como periodicidade padrão das arestas para aplicação o valor de 10ms. A política baseada em comunicação deve movimentar o conjunto de módulos que irá gerar menos impacto no tempo de ciclo da aplicação e no uso de rede. Nesse cenário a cloud está muito distante da cloudlet fisicamente, portanto a política deve evitar o aumento na quantidade de dados transmitidos com origem ou destino sendo a cloud .

Inicialmente é criado um conjunto com o módulo 3. O módulo 2 possui aresta do tipo Up para o módulo 3 que está no mesmo dispositivo e os módulos 1 e 4 estão em outros dispositivos. Portanto esses módulos não podem ser considerados como conjuntos iniciais. Então, na primeira iteração é adicionado um novo conjunto: {módulo 3, módulo 2}. Esse conjunto é criado pois o módulo 2 é origem de uma aresta com destino sendo um módulo do conjunto, o 3. E por fim, na terceira iteração nenhum conjunto é adicionado. Assim os conjuntos que terão análise de impacto de comunicação com a alocação dos módulos na cloud são: {módulo 3}, {módulo 3, módulo 2}.

Para o conjunto {módulo 3}:

1) A aresta 3 não precisará ser transmitida para a cloud, portanto o impacto é - 500 * 0.5 = -250 bytes.
2) A aresta 4 não precisará ser transmitida a partir da cloud, portanto o impacto é - 200 * 1 = -200 bytes.
3) A aresta 2 passará a ser transmitida para a cloud, portanto o impacto é 450 * 1 = 450 bytes.
4) A aresta 5 passará a ser transmitida da cloud, portanto o impacto é 200 * 1 * (10/50) = 40 bytes.

Logo, o impacto de comunicação para esse conjunto é 40 bytes.

Para o conjunto {módulo 3, módulo 2}:

1) A aresta 3 não precisará ser transmitida para a cloud, portanto o impacto é - 500 * 0.5 = -250 bytes.
2) A aresta 4 não precisará ser transmitida a partir da cloud, portanto o impacto é - 200 * 1 = -200 bytes.
3) A aresta 1 passará a ser transmitida para a cloud, portanto o impacto é 500 * 1 = 500 bytes.
4) A aresta 5 passará a ser transmitida da cloud, portanto o impacto é 200 * 1 * (10/50) = 40 bytes.

Logo, o impacto de comunicação para esse conjunto é 90 bytes.

Portanto, neste exemplo apresentado, o conjunto que menos aumenta a quantidade de dados sendo transmitidos que tem a cloud como destino ou origem é o {módulo 3}.

7. Estratégias de escalonamento

Neste trabalho, analisamos estratégias distintas de escalonamento das aplicações desenvolvidas com base nas políticas de alocação de módulos presente no iFogSim e nas propostas sugeridas. Tais estratégias são: concorrente, FCFS (*first come, first served)* e *delay priority*, tanto na forma apresentada por Bittencourt et al.[1], quanto com algumas variações propostas.

7.1. Estratégias *first come, first served* (FCFS)

A primeira estratégia estudada leva em consideração a quantidade de recursos disponível em cada dispositivo. Para tal, aloca as aplicações conforme a ordem de chegada e, caso o dispositivo não seja capaz de atender uma aplicação, ela é movida para outro dispositivo, utilizando-se o algoritmo Edgewards padrão do iFogSim ou um dos algoritmos Edgewards propostos, o que considera as instâncias individualmente ou o que faz análise de comunicação. Consequentemente, serão analisadas três estratégias FCFS distintas. Por questão de simplicidade, as nomearemos FCFS padrão (com algoritmo Edgewards padrão), FCFS individual (com algoritmo Edgewards que considera as instâncias individualmente) e FCFS baseado em comunicação (com o algoritmo Edgewards baseado em comunicação).

7.2. Estratégias *delay priority*

Uma segunda abordagem possível é escalonar as aplicações de acordo com a sensibilidade que elas apresentam em relação a atrasos. Por exemplo, entre as aplicações estudadas, o jogo de realidade virtual é menos tolerante a atrasos do que o sistema de

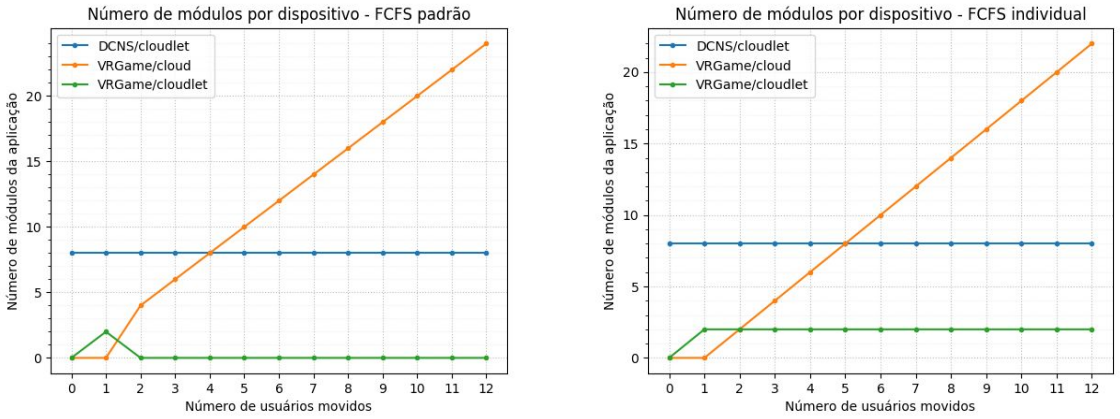
vigilância. Portanto, esta estratégia aloca primeiro no dispositivo as aplicações que possuem uma maior prioridade em termos de delay.

Caso o dispositivo não tenha capacidade suficiente para atender uma aplicação, será empregado o mesmo procedimento que a estratégia anterior. Portanto, essa estratégia também apresenta três variantes. Sendo elas *delay priority* padrão, na qual todas as instâncias do módulo são movidas juntas conforme o algoritmo Edgewards padrão, *delay priority* individual, em que cada instância do módulo é movida independentemente conforme algoritmo Edgewards individual proposto e *delay priority* baseado em comunicação, em que as instâncias dos módulos são movidas de acordo com a análise de comunicação, conforme o algoritmo Edgewards baseado em comunicação.

8. Análise de desempenho do algoritmo Edgewards individual em relação ao algoritmo Edgewards padrão

Uma vez definidas as estratégias de escalonamento a serem estudadas, foi simulado, para cada uma delas, um cenário onde a rede apresentava a primeira topologia descrita na seção 4, ou seja, apenas um nível de cloudlets entre o dispositivo final e a nuvem. Além disso, havia inicialmente quatro aplicações VSOT conectadas à cloudlet 1, e, então, usuários executando a aplicação VRGame começavam a migrar para esta cloudlet. Portanto, os cenários nos quais as aplicações foram analisadas são os mesmos de Bittencourt et al.[1], visando a uma análise comparativa entre os resultados.

Primeiramente, podemos observar, no gráfico de módulos por dispositivo da figura 5, a distribuição de módulos entre a cloudlet e a cloud pelas estratégias FCFS que utilizam o algoritmo edgewards padrão e o algoritmo individual. Nota-se que ambos algoritmos mantiveram todos os módulos da aplicação VSOT alocados na cloudlet, uma vez que são os primeiros módulos a serem alocados. Entretanto, é possível verificar a diferença entre os algoritmos a partir do segundo VRGame movido, uma vez que nesses casos não é possível alocar todas as aplicações VSOT e VRGame na cloudlet simultaneamente. Enquanto o algoritmo padrão move todos os módulos do VRGame para a cloud, o algoritmo individual mantém parte dos módulos na cloudlet e aloca o restante para a nuvem.



(a) (b)

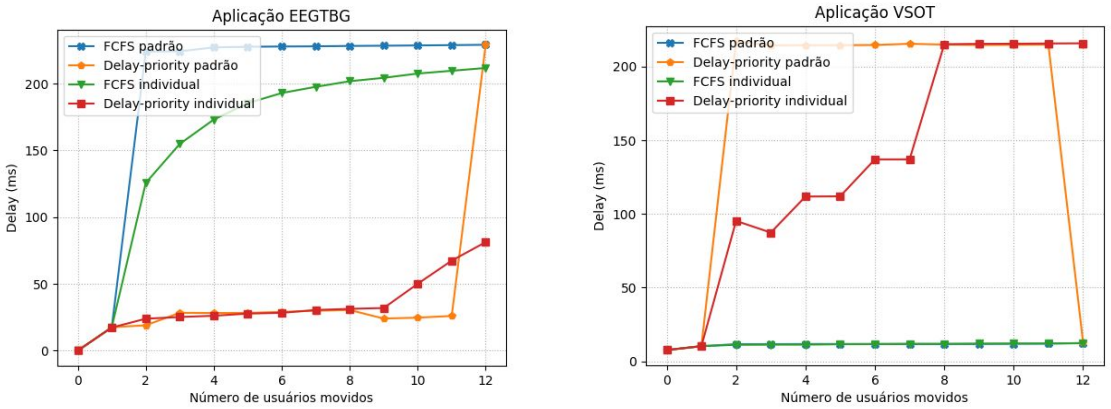
Figura 5: Gráfico do número de módulos de cada aplicação alocados para cada dispositivo de acordo com o número de aplicações VRGame movidas para a cloudlet. (a) Estratégia FCFS padrão (b) Estratégia FCFS individual.

Consequentemente, o algoritmo proposto explora ao máximo a capacidade da cloudlet e mantém mais módulos próximos ao usuário. Dessa forma, o tempo médio de comunicação diminui e o delay médio das aplicações VRGame é menor, conforme pode ser observado no gráfico de delay médio para cada estratégia na figura 6.

A partir da segunda aplicação VRGame, o algoritmo padrão observa uma mudança abrupta do delay, aumentando quase 200ms em relação ao observado quando todas as instâncias podem ser alocadas na cloudlet. Enquanto isso, o delay médio para o algoritmo individual aumenta de forma mais gradual, conforme o número de módulos na cloud aumentam. Entretanto, é necessário ressalvar que nem todas as instâncias da aplicação VRGame observarão esse delay menor. De fato, se for analisado o delay observado por cada instância da aplicação de forma individual, em vez de se analisar o delay médio, será observado que a primeira instância, por ser alocada para a cloudlet, praticamente não sofrerá variações em seu delay, independente da quantidade de instâncias movidas. Por outro lado, as demais instâncias serão alocadas para a cloud e terão um delay muito próximo ao observado pelas aplicações no algoritmo padrão.

Além disso, um impacto muito pequeno pode ser observado nas aplicações VSOT, que apresentam um delay ligeiramente maior com o algoritmo individual. Isso ocorre porque, no algoritmo padrão, todas as instâncias do VRGame são movidas para a cloud a partir da chegada da segunda aplicação, deixando somente as aplicações VSOT na cloudlet. Já no

algoritmo individual, uma instância é mantida na cloudlet, acarretando uma maior concorrência por recursos.



(a) (b)

Figura 6: Gráfico do delay médio das instâncias da aplicação para diferentes políticas de alocação de acordo com o número de aplicações VRGame. (a) Aplicação VRGame (b) Aplicação VSOT.

Analisando-se agora o impacto dos diferentes algoritmos edgewards para a estratégia delay priority, observa-se no gráfico de alocação dos módulos por dispositivo da figura 7 que, até a chegada da primeira instância do VRGame, os algoritmos apresentam o mesmo resultado, visto que todos os módulos podem ser alocados na cloudlet. E, com a chegada da segunda até a oitava instância, ambos os algoritmos alocam todos os módulos do VRGame na cloudlet, pois estes possuem uma prioridade maior que a aplicação VSOT em termos de delay.

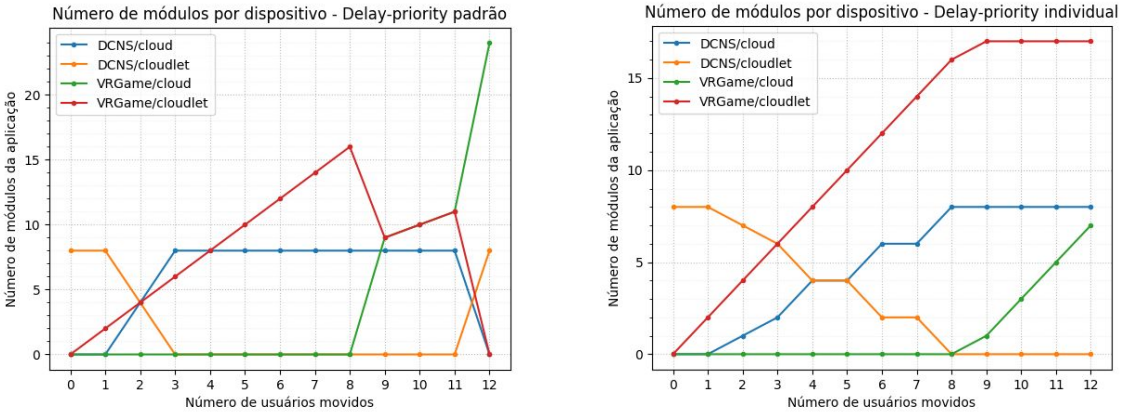
A diferença em o algoritmo edgewards padrão e individual consiste em como os módulos das aplicações VSOT são alocados. Com a chegada da segunda instância do VRGame, o algoritmo padrão move para a cloud os módulos rastreador de objetos das 4 instâncias VSOT anteriormente alocadas na cloudlet. E, a partir da terceira instância, também move os 4 módulos detectores de objetos, de forma que todas as aplicações VSOT sejam inteiramente alocadas na cloud. Por outro lado, o algoritmo individual move, com a chegada de novas aplicações, somente a quantidade necessária de módulos do VSOT para conseguir alocar todas as instâncias do VRGame. Logo, os módulos são movidos de forma mais gradual, como pode ser observado na figura 7.

Com a chegada da segunda instância do VRGame, o rastreador de objetos de uma instância do VSOT é movido para a cloud. Com a terceira instância, o detector de objetos associado ao rastreador de objetos movido anteriormente também é alocado para a cloud. E, então, na chegada do quarto, do sexto e do oitavo VRGame, os módulos detectores de objetos e rastreador de objetos de mais uma instância da aplicação VSOT são movidos para a cloud. Assim, com a chegada do oitavo VRGame, os dois algoritmos voltam a apresentar o mesmo resultado, com todos os módulos do VRGame alocados na cloudlet e todos os módulos do VSOT na cloud.

Entretanto, com a chegada da nona instância, os algoritmos alocam os módulos de forma distinta novamente. Como a cloudlet não tem capacidade suficiente para processar todos os módulos das 9 instâncias de VRGame, o algoritmo padrão decide mover os módulos conector de todos os VRGames para a cloud, e mantém esse comportamento com a chegada da décima e da décima primeira instância. Enquanto isso, o algoritmo individual mantém os módulos conector das 8 primeiras instâncias na cloudlet, alocando, na cloud, somente o conector da nona instância, e o calculador de concentração e conector da décima e décima primeira instância, além dos módulos do VSOT.

Por fim, com a chegada da décima segunda instância, há uma grande diferença entre o resultado dos dois algoritmos. Uma vez que a cloudlet não é capaz de processar 12 módulos calculadores de concentração, o algoritmo padrão opta por mover todos eles para a cloud, juntamente com os conectores. Assim, aloca todos os módulos dos VRGames na cloud, liberando capacidade computacional suficiente na cloudlet para processar todos os módulos do VSOT. Logo, o algoritmo padrão passa a alocar os VRGames completamente na cloud e os VSOT completamente na cloudlet, apesar do VRGame possuir uma prioridade maior em termos de delay.

O algoritmo individual, por sua vez, mantém o mesmo comportamento que apresentou na chegada da décima e da décima primeira instância. Assim, aloca os módulos calculadores de concentração e conector do décimo segundo VRGame na cloud, e mantém os demais módulos da mesma forma que estavam antes da chegada dessa instância.



(d) (d)

Figura 7: Gráfico do número de módulos de cada aplicação alocados para cada dispositivo de acordo com o número de aplicações VRGame movidas para a cloudlet. (a) Estratégia *delay priority* padrão (b) Estratégia *delay priority* individual.

Confirma-se, portanto, que a utilização do algoritmo individual não só apresenta uma maior utilização da capacidade computacional das cloudlets e mantém os módulos mais próximos do usuário final, como garante que a política delay priority irá sempre favorecer os dispositivos que tenham maior prioridade. Isso ocorre porque este algoritmo não possui cenários onde módulos com alta prioridade e previamente alocados para a cloudlet tenham que ser movidos para a cloud, o que acontece no algoritmo padrão, como foi possível observar com a chegada do décimo segundo VRGame.

Este comportamento, por sua vez, reflete diretamente no delay observado pelas aplicações, conforme observado na figura 6, uma vez que, com a chegada da décima segunda instância, o delay médio observado pelo VRGame foi de 211ms ao utilizar o algoritmo edgewards padrão, e apenas 80ms ao utilizar o algoritmo edgewards individual. Por outro lado, o delay observado pelo VSOT foi de 14ms no algoritmo padrão e 216ms no algoritmo individual, uma vez que todos os módulos foram alocados na cloud. Como a estratégia delay priority tem por objetivo garantir um menor delay para aplicações mais sensíveis a atrasos, o algoritmo individual apresenta um desempenho muito melhor.

Além disso, apesar desse algoritmo ter apresentado um delay muito maior para o VSOT no caso específico em que há 12 instâncias de VRGame, é possível notar que, para os casos em que havia entre 2 e 7 instâncias, o delay foi consideravelmente menor. Isso aconteceu porque, conforme já analisado, o algoritmo individual manteve parte dos módulos

VSOT sendo executados na cloudlet, enquanto o algoritmo padrão enviou todos para a cloud. E, ainda, é possível notar que o delay observado pelo VSOT no algoritmo individual cresceu com a chegada da segunda, quarta, sexta e oitava instâncias do VRGame, que foram exatamente os momentos em que o algoritmo individual enviou mais módulos do VSOT para a cloud.

Com a chegada do terceiro VRGame, entretanto, observa-se uma diminuição do delay, apesar do número de módulos enviados para a cloud ter aumentado. Isso acontece porque, com a chegada dessa instância, a alocação é feita de forma que a quantidade total de mips dos módulos da cloudlet seja 3900, enquanto, no momento da chegada da segunda instância, era de 4000 MIPS, exatamente a capacidade da cloudlet. Assim, apesar do número de módulos na cloud ter aumentado com a terceira instância, a concorrência por recursos na cloudlet diminuiu, resultando em um menor tempo de processamento e compensando o aumento no atraso de comunicação.

Um último ponto a ser analisado a partir do gráfico de delay médio da figura 6 é o fato do delay observado pelo VRGame com o algoritmo individual ter aumentado com a chegada das instâncias 10 e 11, enquanto o delay do algoritmo padrão diminuiu com as instâncias 9, 10 e 11.

Para isso, é necessário lembrar primeiramente que o delay medido para a aplicação VRGame e apresentado no gráfico considera o ciclo da aplicação formado pelos módulos:

$$eeg \rightarrow cliente \rightarrow calculador\ de\ concentração \rightarrow cliente \rightarrow display$$

Tal ciclo representa a coleta de dados através do sensor, o processamento desses dados e a apresentação do estado do jogador, sendo portanto o ciclo de principal interesse para o usuário. Um outro ciclo possível, poderia levar em consideração o tempo necessário para enviar os dados do jogo ao conector, ele ser executado, atualizar o estado de todos os jogadores, e então enviar esse estado para ser exibido para o usuário. Tal ciclo seria formado pelos módulos:

$$eeg \rightarrow cliente \rightarrow calculador\ de\ concentração \rightarrow conector \rightarrow cliente \rightarrow display$$

Entretanto, tal ciclo não foi considerado.

Dito isso, podemos analisar o atraso observado na estratégia delay priority para os dois algoritmos edgewards, com a chegada das instâncias 9 a 11 do VRGame. Conforme mencionado anteriormente, com a chegada da nona instância, o algoritmo edgewards padrão decide mover todos os módulos conector para a cloud e mantém esse comportamento após a chegada das instâncias 10 e 11. Como o ciclo de interesse da aplicação não contém esse módulo, movê-lo para a cloud não trará prejuízo para o delay observado, uma vez que o tempo de enviar os dados até ele, executá-lo e enviar as informações de volta ao dispositivo não será levado em consideração.

Na verdade, movê-lo para a cloud ainda tem como efeito colateral diminuir a carga na cloudlet, fazendo com o que os módulos calculadores de concentração sejam executados de forma mais rápida e causem a diminuição do delay observada no gráfico. Note que, na prática, isso significa que o usuário receberá mais rapidamente as informações sobre o seu estado no jogo, entretanto, notará um delay maior para receber o estado dos outros jogadores, pois o ciclo com o conector será impactado.

Por outro lado, para a estratégia delay priority com o algoritmo edgewards individual, o escalonamento será feito de forma que o calculador de concentração das instâncias 10 e 11 fiquem na cloud, enquanto o conector das instâncias 1 a 8 sejam alocados para a cloudlet. Portanto, como parte das instâncias do calculador de concentração estarão na cloud e este módulo é considerado no ciclo analisado, o delay médio observado deve aumentar consideravelmente com a chegada das instâncias 10 e 11, o que de fato ocorre.

Um último ponto a ser analisado, a fim de comparar os algoritmos edgewards padrão e individual, é a utilização da rede por cada uma das estratégias. Através do gráfico de uso da rede da figura 8, verifica-se que tanto a estratégia delay priority, quanto FCFS, foram beneficiadas pela utilização do algoritmo individual, uma vez que esse consegue alocar uma quantidade maior de módulos na cloudlet e, portanto, necessita enviar menos dados até a cloud. E verifica-se que esse envio de dados para cloud é o principal responsável pelo aumento do uso da rede, uma vez que as principais variações ocorrem exatamente nos pontos onde migrações são realizadas. Por exemplo, com a chegada das instâncias 2, 3, 4, 6 e 8 na estratégia delay priority.

Além disso, é possível notar que a estratégia FCFS apresentou uma menor utilização da rede porque manteve os módulos VSOT alocados na cloudlet, e estes realizam uma troca de dados maior que os módulos do VRGame. Isso também pode ser verificado com a chegada da décima segunda instância na estratégia delay priority padrão, visto que esta decide mover os VRGames para a cloud e mover os VSOT de volta para a cloudlet, gerando uma grande diminuição da utilização da rede.

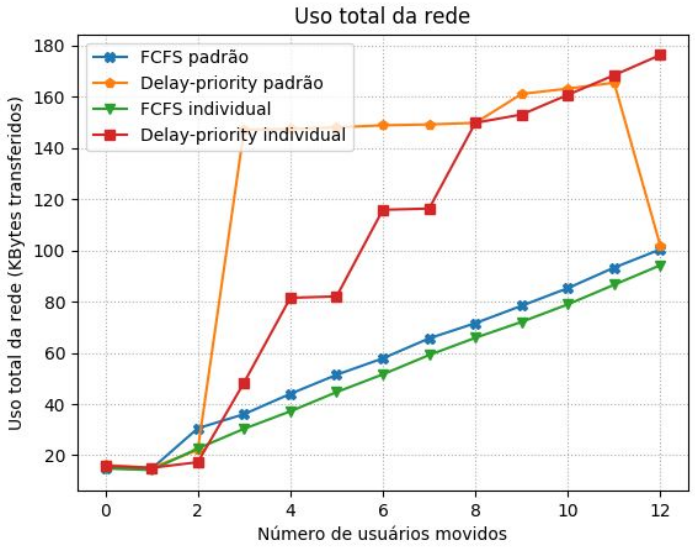


Figura 8: Gráfico do uso total da rede por cada estratégia.

9. Análise de desempenho do algoritmo Edgewards baseado em comunicação em comparação com o Edgewards padrão

Os cenários utilizados para a análise dessa política são os mesmos utilizados na simulação da política *edgewards individual* onde usuários executando a aplicação VRGame migravam gradualmente para a cloudlet que já executava quatro aplicações VSOT.

9.1. FCFS

Primeiramente foram executados experimentos utilizando a estratégia de escalonamento FCFS, com as variantes padrão e baseado em comunicação, que nas figuras está indicado por CB (communication based). Os resultados da alocação de módulos e delay médio são apresentados na Figura 9.

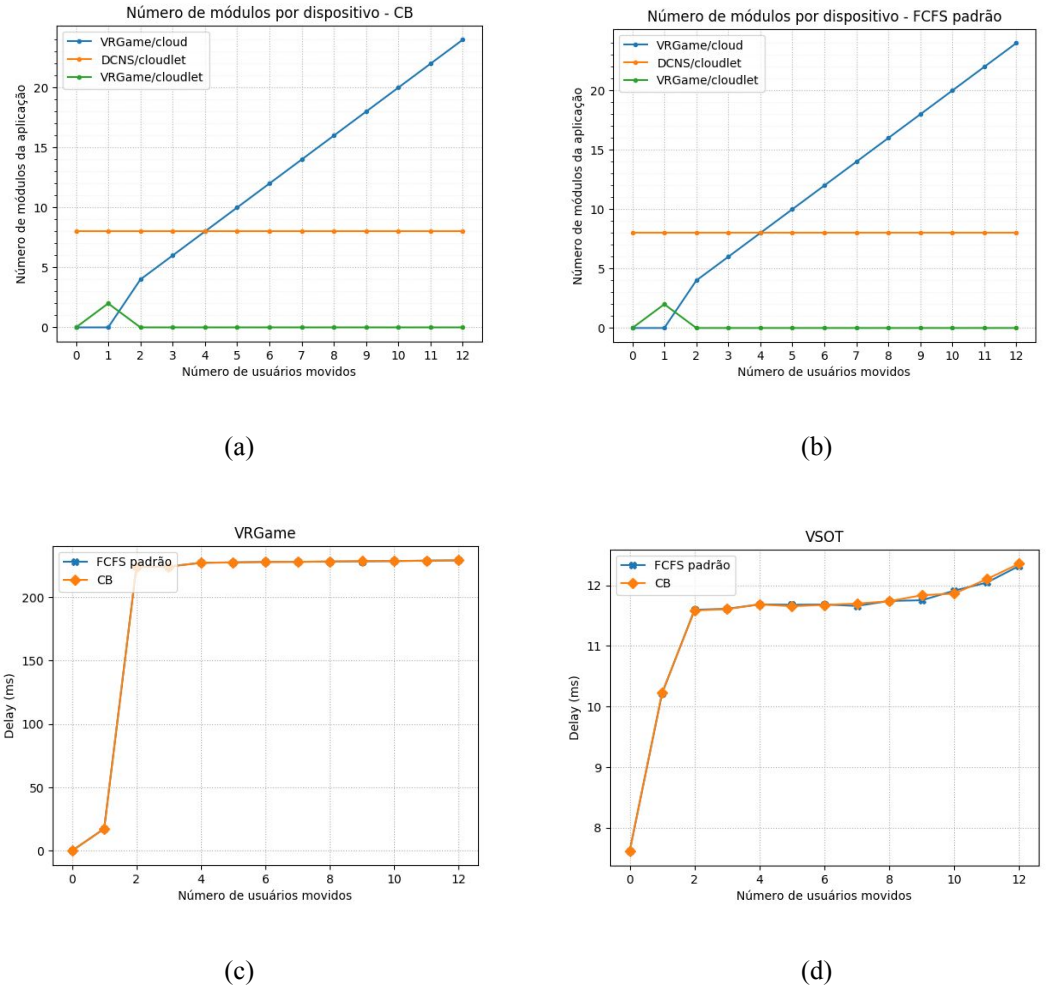


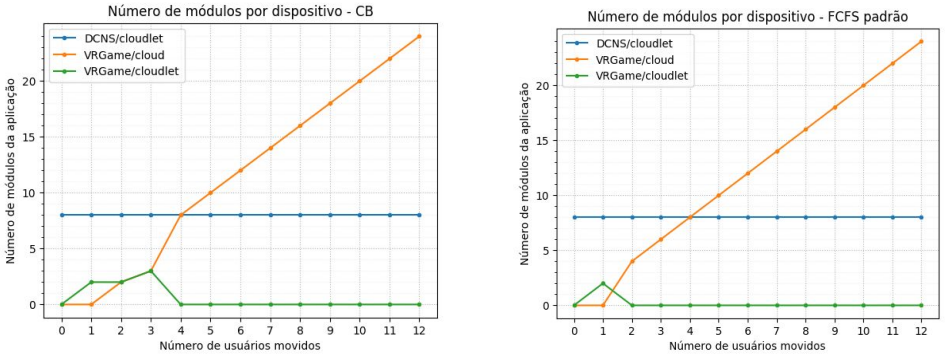
Figura 9: Resultados para a política *edgewards padrão* e *edgewards communication based* utilizando FCFS. O número de módulos por dispositivo para a política baseada em comunicação é mostrada em (a) e para a padrão em (b). O delay da aplicação VRGame para as duas políticas é mostrado em (c) e o delay da aplicação VSOT é mostrado em (d).

Os delays para o VRGame e para a VSOT mostram que não teve diferenças significativas entre entre as duas abordagens. Essa semelhança se deve ao fato de que a alocação de módulos foi exatamente a mesma para as duas políticas.

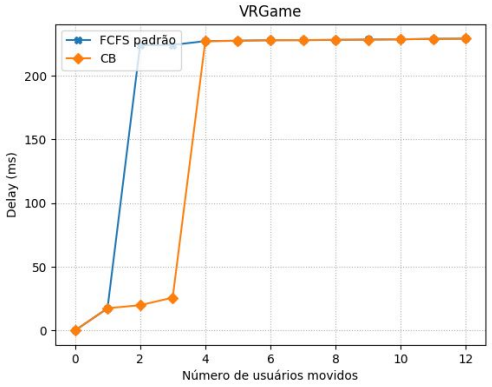
Para o cenário proposto, como a estratégia é a FCFS, as aplicações VSOT são alocadas primeiro e portanto são alocadas para a cloudlet e permanecem nelas mesmo com novos usuários da aplicação VRGame migrando para essa cloudlet. Nesse ponto era esperado que as duas políticas resultassem na mesma alocação de módulos para a VSOT. Para a aplicação VRGame, têm-se que a cloudlet suporta a migração dos dois primeiros usuários, no entanto, para o terceiro é necessário que ocorra uma realocação dos módulos dessa aplicação entre os dispositivos, visto que a cloudlet não tem os recursos para processar todos os

módulos. A política *edgewards* padrão tenta alocar o módulo calculador de concentração para a terceira instância do VRGame, e por não conseguir, acaba realocando todas as instâncias dos módulos calculador de concentração e conector para serem executadas na cloud. Já a política *edgewards communication based*, ao tentar alocar o módulo calculador de concentração, faz a comparação do impacto de comunicação adicionado na cloud caso seja realocado somente as instâncias do módulo conector e caso sejam realocados tanto as instâncias do módulo calculador de concentração quanto as do conector. As instâncias do módulo conector são escolhidas para serem enviadas para a cloud pois este módulo possui impacto igual a 300 bytes enquanto o conjunto com os módulos conector e calculador de concentração resultaria em um impacto de 1100 bytes. No entanto, ao serem realocados as duas instâncias de conector presentes na cloudlet para a cloud, a quantidade de capacidade de processamento liberada na cloudlet ainda não é suficiente para que o terceiro módulo do calculador de concentração seja alocado na cloudlet. Portanto, as duas instâncias anteriores e a nova que precisa ser alocada são enviadas para a cloud. Com isso, conforme novos usuários vão migrando, todos os próximos módulos do tipo calculador de concentração e do tipo conector são alocados na cloud uma vez que já existem instâncias desses módulos sendo executados neste dispositivo.

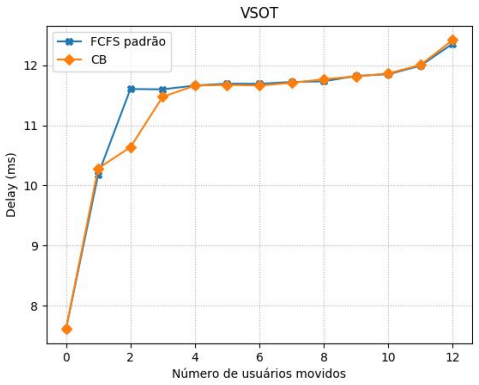
Para que seja notada diferença entre as duas políticas para esse cenário, é preciso que o módulo conector tenha uma quantidade requerida de processamento (MIPS) maior e o módulo calculador de concentração tenha uma quantidade menor. Assim, quando as instâncias do conector forem realocadas, o módulo calculador de concentração poderá ser alocado na cloudlet. Os resultados da execução do cenário anterior com essas alterações propostas é mostrado na figura 10.



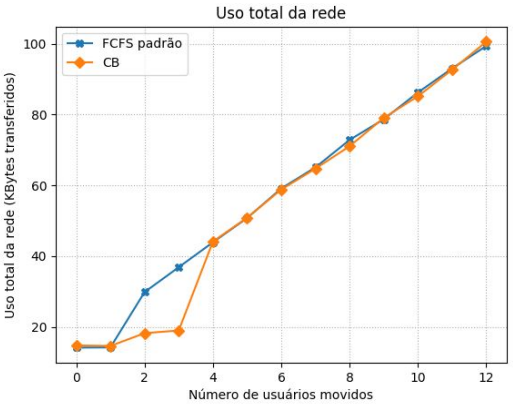
(a) (b)



(c)



(d)



(e)

Figura 10: Resultados para a política *edgewards padrão* e *edgewards communication based* utilizando FCFS com alteração na quantidade de MIPS requerida para execução dos módulos. O número de módulos por dispositivo para a política baseada em comunicação é mostrada em (a) e para a padrão em (b). O delay da aplicação VRGame para as duas políticas é mostrado em (c) e o delay da aplicação VSOT é mostrado em (d). O uso total da rede é apresentado em (e).

Nesse novo cenário de execução é necessário realocar os módulos já no segundo usuário VRGame que migra para a cloudlet, pois ela fica sem recursos de processamento suficientes para processar todos os módulos. Da mesma forma que no cenário anterior, a política *edgewards* realoca na cloud tanto as instâncias dos módulos conector quanto as do calculador de concentração. No entanto, para a política baseada em comunicação, agora é possível realocar as instâncias do módulo conector na cloud e com a diminuição no processamento requerido, a cloudlet agora é capaz de suportar o novo módulo do calculador de concentração. A diferença entre as políticas de alocação é visível nas figuras 10 (a) e (b). Na figura 10 (a), quando dois usuários do VRGame migram, dois módulos dessa aplicação

são mantidos na cloudlet, que são referentes às instâncias do calculador de concentração, e dois são enviados para serem executados na cloud, que são referentes às instâncias do conector. Na figura 10 (b), para este mesmo caso, quatro módulos são executados na cloud e, portanto, nenhum módulo dos usuários VRGame está sendo executado na cloudlet.

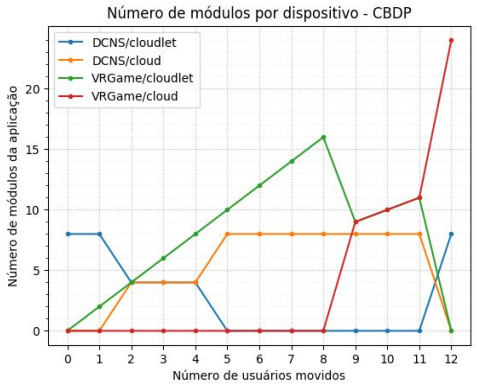
Para o caso em que três usuários VRGame realizam a migração, as duas políticas ainda se mantém distintas, de forma que a padrão aloca os seis módulos na cloud e a baseada em comunicação aloca três módulos do conector na cloud e três módulos do calculador de concentração na cloudlet. A partir do quarto usuário que migra, não é possível para a política baseada em comunicação colocar mais um módulo de calculador de concentração na cloudlet, e portanto, os 8 módulos são alocados na cloud. Dessa forma, a partir desse ponto, as duas políticas têm o comportamento igual.

Analisando os casos com migração de dois e três usuários VRGame, que são os casos onde as diferenças entre as políticas são evidenciadas, pode-se perceber que o delay no ciclo do VRGame é muito menor para política baseada em comunicação. No cálculo do delay ao longo do ciclo do VRGame, o módulo conector não está sendo levado em conta, portanto, a diferença nos resultados das duas políticas se deve ao fato de que o módulo calculador de concentração se mantém na cloudlet com o uso de uma política e na cloud com o uso da outra. Apesar do delay da aplicação VRGame não evidenciar nesses casos que de fato menos transmissão de dados ocorreu, através da figura 10 (e), têm se a confirmação de que a política baseada em comunicação de fato alocou os módulos de modo que ocorreu menor uso da rede do que em comparação com a alocação feita pela política padrão. Para o caso em que 3 usuários migraram, o uso da rede foi de cerca de 37 KB para a política padrão, contra 19 KB para a política baseada em comunicação.

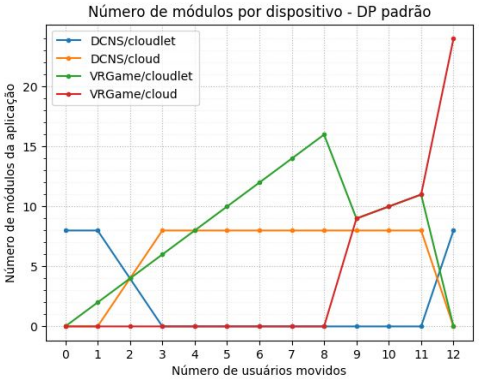
9.2. Delay Priority

Uma vez executadas as simulações com a estratégia FCFS, a próxima estratégia executada foi a Delay Priority, nas variantes com edgewards padrão e baseado em comunicação, que nas figuras é apresentado como CBDP (communication based delay priority). Para esses experimentos considerou-se a quantidade original de MIPS exigida para

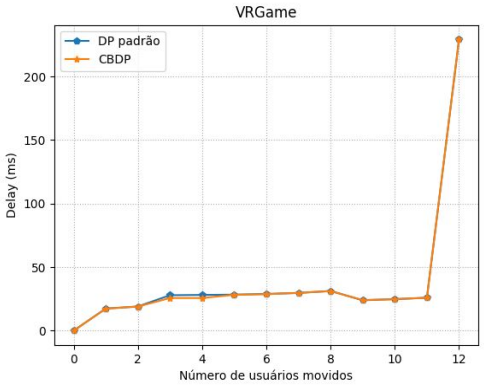
processamento dos módulos do VRGame. Os resultados da alocação de módulos, delay e uso da rede são mostrados na figura 11.



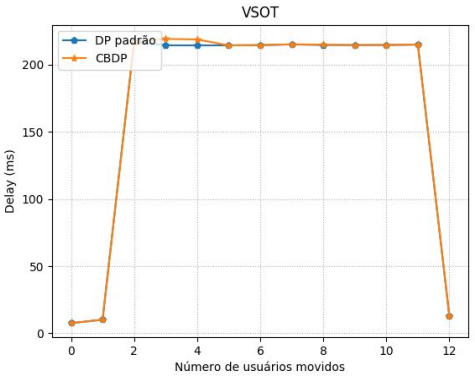
(a)



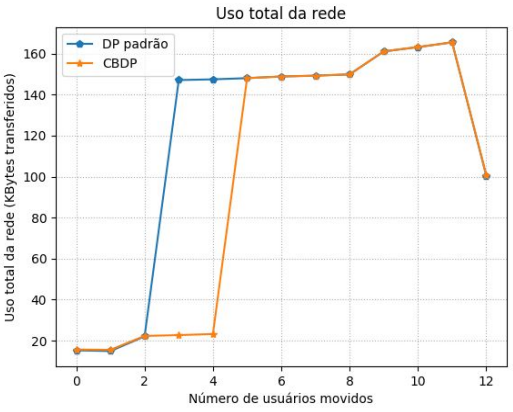
(b)



(c)



(d)



(e)

Figura 11: Resultados para a política *edgewards* e *edgewards communication based* utilizando Delay Priority. O número de módulos por dispositivo para a política baseada em comunicação é mostrada em (a) e para a

padrão em (b). O delay da aplicação VRGame para as duas políticas é mostrado em (c) e o delay da aplicação VSOT é mostrado em (d). O uso total da rede é apresentando em (e).

Como se trata da estratégia Delay Priority, as instâncias da aplicação VRGame serão alocadas com maior prioridade que a aplicação VSOT, visto que a primeira aplicação é mais sensível a delay do que a segunda.

Ao analisar os gráficos de módulos por dispositivo, observa-se que a política *edgewards* baseada em comunicação tomou decisões diferentes de alocação em comparação a política *edgewards* padrão para os casos em que três e quatro usuários do VRGame haviam migrado. Para esses casos, quando a cloudlet fica sem recursos, o primeiro módulo que já não pode mais ser alocado é o detector de objetos. Como o módulo rastreador de objetos tem entrada que depende dos resultados do módulo detector de objetos, a política *edgewards* padrão move todas as instâncias desses dois módulos para a cloud e passa alocar as próximas instâncias diretamente na cloud. A política baseada em comunicação, faz uma análise do impacto de comunicação gerado pela movimentação de cada conjunto de módulos e conclui que movimentar somente o módulo rastreador de objetos é melhor pois causa um impacto de somente 106 bytes em comparação com o impacto de 1906 bytes atrelado a realocar os dois módulos para a cloud. Para os casos em que cinco ou mais usuários VRGame migraram, as duas políticas se comportaram da mesma maneira visto que não era mais possível manter todas as instâncias do módulo detector de objetos na cloudlet.

Essa diferença de alocação fez com que o uso da rede com a política baseada em comunicação fosse significativamente inferior em comparação com a política padrão, sendo de 22KB para a primeira política e 147KB para a segunda, indicando que de fato houve menor transmissão de dados. É importante notar que o delay da aplicação VSOT quando utilizada a política baseada em comunicação teve um pequeno aumento em comparação com o uso da política padrão. Uma possível razão para isso ter ocorrido é que a cloudlet fica com mais módulos para processar com a primeira política, e por isso, pode demorar mais para gerar as respostas que serão utilizadas como entrada para os próximos módulos, sendo portanto um acréscimo de tempo devido ao processamento e não a transmissão.

Para os cenários estudados, pôde-se perceber que a política baseada em comunicação conseguiu diminuir o uso da rede, no entanto não teve efeito na diminuição do delay das aplicações. Para que se possa observar as vantagens da política baseada em comunicação, é

importante que a diferença de impacto entre o conjunto de módulos escolhidos para serem movidos pela política baseada em comunicação seja grande em comparação com o conjunto escolhido pela política padrão. Portanto foram executados experimentos em que a aresta dois da aplicação VSOT tivesse sua quantidade de bytes transmitidos aumentada. Dessa forma, a escolha da política padrão de alocar o módulo detector de objetos na cloud, geraria muito impacto de comunicação neste dispositivo. Os resultados das simulações com a aresta dois tendo sua quantidade de bytes aumentada são mostrados na figura 12.

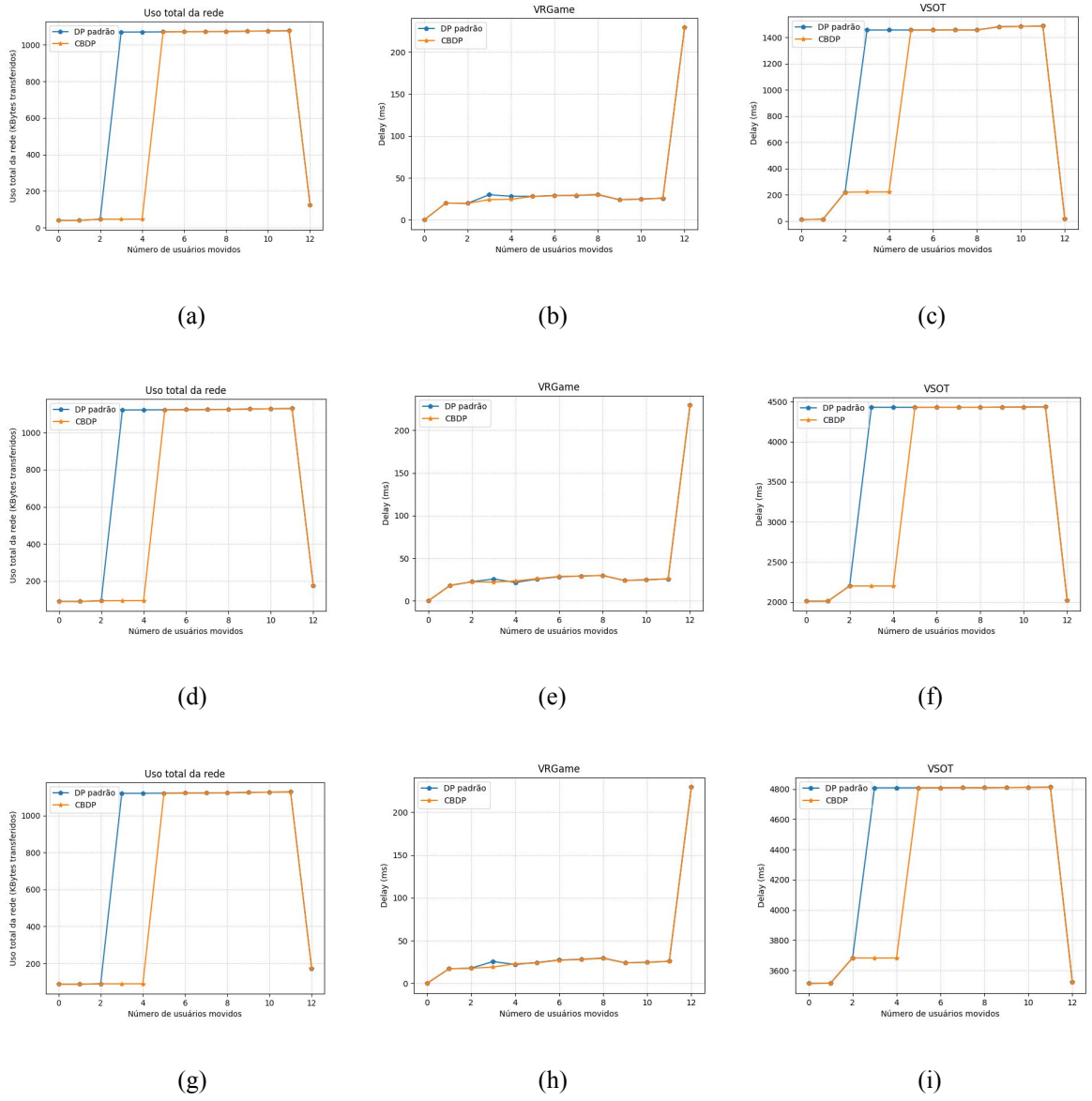


Figura 12: Resultados para a política *edgewards* e *edgewards communication based* utilizando Delay Priority com alteração na quantidade de bytes transmitidos pela aresta dois. O uso total da rede, o delay da aplicação VRGame e o delay da aplicação VSOT são apresentados em (a), (b) e (c) para a simulação com aresta

multiplicada por dez, em (d), (e) e (f) para a simlação com multiplicação por cinquenta e em (g), (h) e (i) para a simulação com multiplicação por cem.

Na figura 12, os gráficos (a), (b) e (c) representam os experimentos com a aresta tendo sua quantidade de bytes transmitidos multiplicada por dez, os gráficos (d), (e) e (f) representam os experimentos com essa quantidade multiplicada por cinquenta e por fim os gráficos (g), (h) e (i) representam os experimentos com a quantidade de bytes transmitidos multiplicada por cem.

Nestes experimentos pode-se perceber que o uso da rede chega a ser cerca de 1MB de diferença entre as duas políticas e a diferença no loop da aplicação VSOT foi cerca de 1000ms para os cenários com multiplicador de dez vezes e de cem vezes, chegando a 2000ms de diferença para o cenário com multiplicador de cinquenta vezes. Assim, esses cenários mostram a relevância da política baseada em comunicação quando existem arestas que transmitem uma quantidade de dados muito maior.

10. Análise do impacto de diferentes níveis hierárquicos na rede

Além de diferentes políticas de alocação de módulos, foram também realizados estudos considerando a execução de aplicações em diferentes topologias de computação fog. Com o intuito de analisar o impacto que a introdução de mais níveis hierárquicos de cloudlets teria no consumo de rede e no tempo de execução das aplicações propostas, modificamos o iFogSim para que fosse possível criar uma camada extra de cloudlets.

Assim, as simulações foram realizadas para 3 diferentes topologias, conforme ilustrado na figura 13.

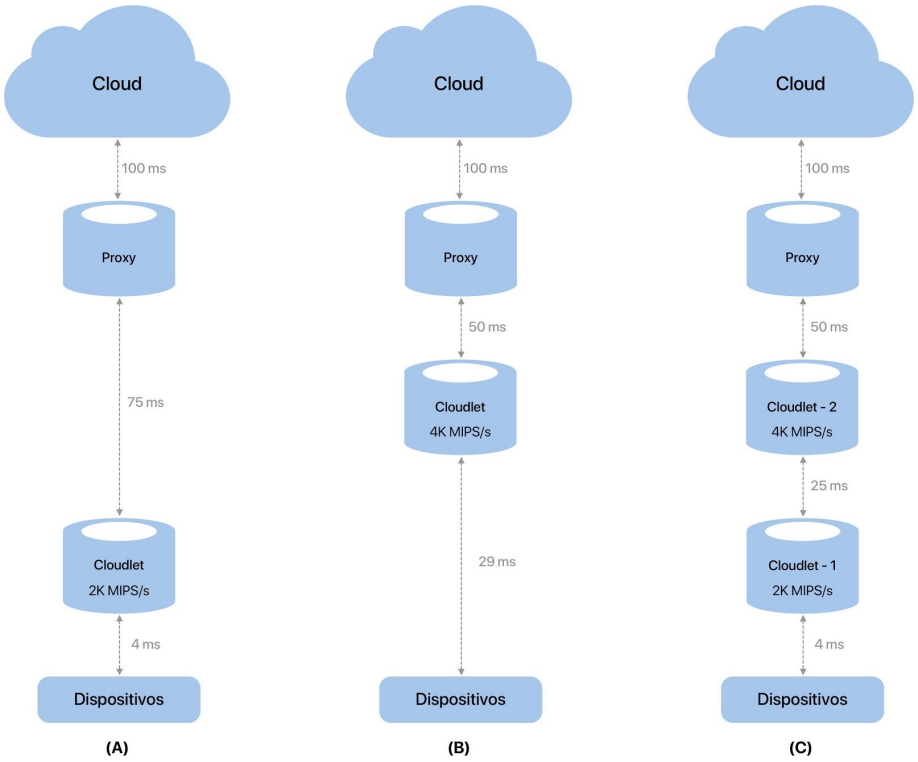


Figura 13: cenários utilizados para análise do impacto de diferentes níveis hierárquicos

A estrutura de rede **A** apresenta uma única cloudlet, de baixo processamento e próxima ao usuário, apresentando latência de apenas 4 ms na comunicação com os dispositivos. Já a estrutura **B** dispõe de uma cloudlet de maior processamento (o dobro da utilizada anteriormente) e que se encontra mais distante dos usuários, apresentando latência de 29 ms na comunicação com os dispositivos conectados. O cenário **C** realiza a união dos dois anteriores: uma cloudlet de baixo processamento e próxima dos usuários se conecta a outra cloudlet, mais distante dos dispositivos móveis porém mais perto do que a nuvem.

O mesmo cenário de simulação utilizado nas análises anteriores foi aplicado para as três topologias, de forma que usuários executando a aplicação VRGame migravam gradualmente para a cloudlet que inicialmente executava quatro aplicações VSOT.

10.1. Análise do uso de rede

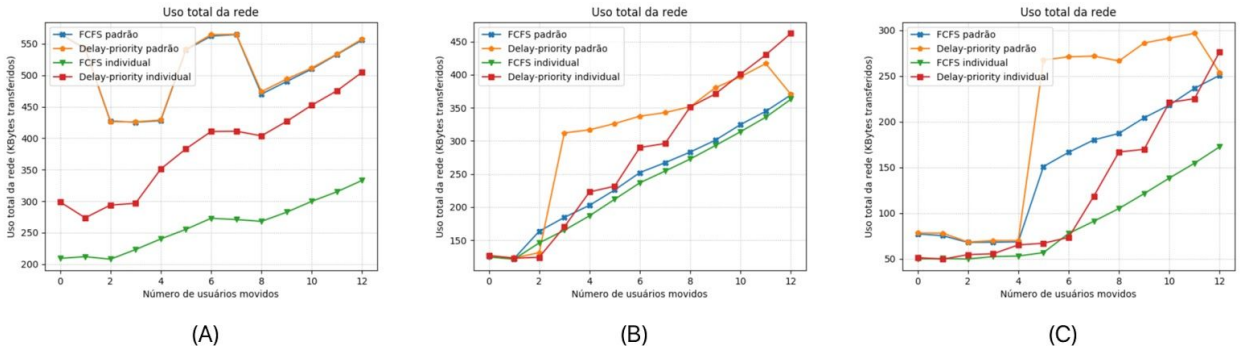


Figura 14: Resultado da simulação para as 3 diferentes topologias

Verifica-se, a partir dos gráficos de uso da rede da figura 14, que, no geral, a configuração (A) (cloudlet única de baixo processamento) faz uso mais intenso da rede, independentemente da política de alocação utilizada, uma vez que muitos módulos precisam ser movidos para a cloud. De forma oposta, poucos módulos precisam ser movidos para a cloud quando utilizada a configuração (B), devido ao maior poder de processamento da cloudlet.

Porém, a combinação das duas configurações é a que apresenta o melhor resultado, realizando uso intenso da rede somente em casos extremos, onde é necessário mover módulos para a cloud, se mostrando especialmente eficaz quando o número de usuários movidos é baixo.

10.2. Análise do delay na execução das aplicações

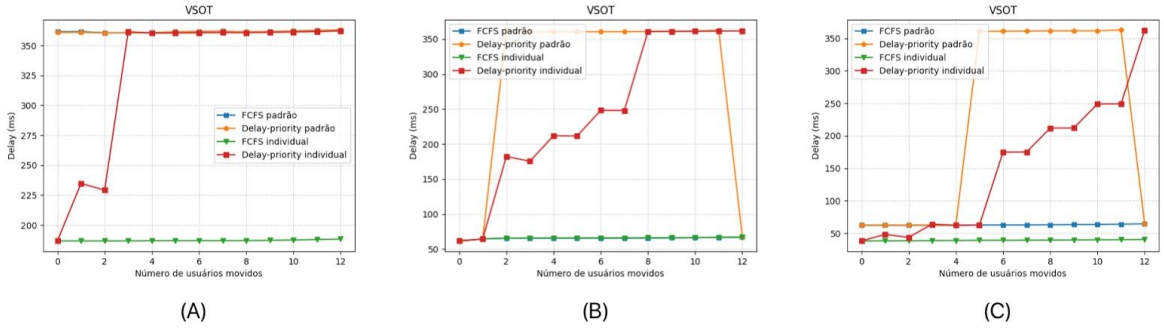


Figura 15: delay na execução da aplicação VSOT para as três topologias.

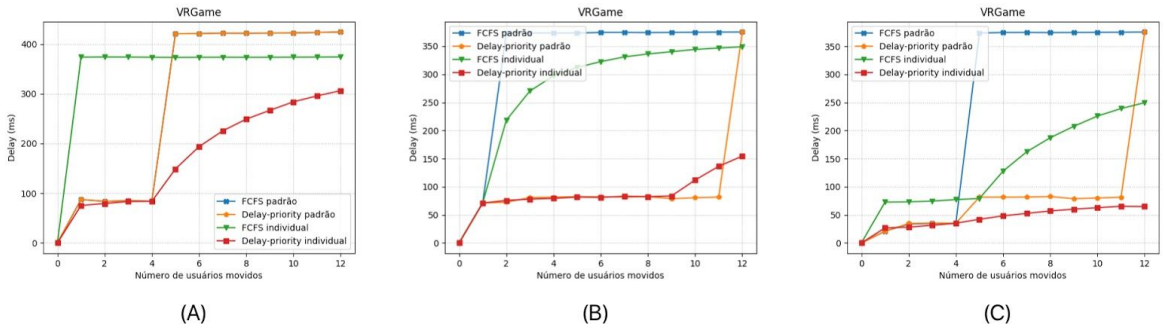


Figura 16: delay na execução da aplicação VRGame para as três topologias.

Quando analisamos o delay na execução da aplicação VSOT em cada configuração, tendo em vista os gráficos da figura 15, vemos que os resultados variam bastante com a política de alocação utilizada. No caso da política *First Come First Served* padrão, observamos que as configurações B e C levam grande vantagem, mantendo o delay sempre menor que 100 ms, enquanto a configuração A, devido ao baixo poder de processamento da cloudlet, gera delays superiores a 350 ms. Quando analisada a política *First Come First Served* individual, observamos que o delay na execução da aplicação VSOT na arquitetura multinível foi reduzido a cerca de um quinto quando comparado com a configuração **A**. Uma melhora ainda mais expressiva foi observada na aplicação VRGame, que para essa política de alocação apresentava delays de aproximadamente 370 ms quando executada na configuração **A** e atingiu o máximo de 250 ms na configuração **C**, conforme apresentado nos gráficos da figura 16.

De maneira semelhante, para as políticas de alocação *Delay Priority*, observamos que a configuração de cloudlets multinível resulta em uma série de vantagens. No caso da política *Delay Priority* padrão, a aplicação VSOT somente apresenta delays acima de 350 ms quando o quinto usuário é movido para a cloudlet, enquanto as configurações de um único nível de cloudlet já apresentam esse delay ao receber o primeiro usuário movido. Comportamento semelhante é observado na análise da aplicação VR Game. Quando o quinto usuário é movido, a configuração de um único nível de baixo processamento já utiliza a cloud para a execução dos módulos, enquanto a configuração multinível move seus usuários para a cloudlet de maior processamento, evitando o delay de acesso a cloud.

No caso da política *Delay Priority* individual as melhoras são ainda mais visíveis, uma vez que é realizado um melhor uso das cloudlets ao mover somente os módulos necessários. Observamos que conforme os usuários são movimentados, o delay da aplicação

VSOT aumenta levemente, uma vez que seus módulos são movidos da cloudlet de nível mais baixo para a de nível mais alto. É só quando os módulos são transferidos para a cloud, com a chegada do sexto usuário, que o delay apresenta um aumento significativo (cerca de 100 ms) e então passa a crescer gradativamente, chegando a cerca de 360 ms com a chegada do décimo segundo usuário. A vantagem dessa configuração de topologia é clara para essa política de alocação, uma vez que a configuração de cloudlet única de baixa capacidade de processamento apresenta delays de 360 ms para a aplicação VSOT já na chegada do terceiro usuário, quando se faz necessário o uso da cloud. Já a configuração com somente a cloudlet de maior processamento garante um resultado intermediário, mantendo a aplicação VSOT com delays abaixo de 250 ms até a chegada do oitavo usuário, quando o delay cresce para 360 ms. Além disso, quando observada a execução da aplicação VRGame, vemos que o delay atinge no máximo 60 ms para a configuração multinível, valor atingido logo na movimentação dos primeiros usuários nas estruturas de nível único.

10.3. Análise de ociosidade

Apesar dos ótimos resultados alcançados com o uso de topologias multiníveis, quando analisados os delays na execução das aplicações, é também necessário se avaliar o custo benefício dessa estrutura de cloudlets.

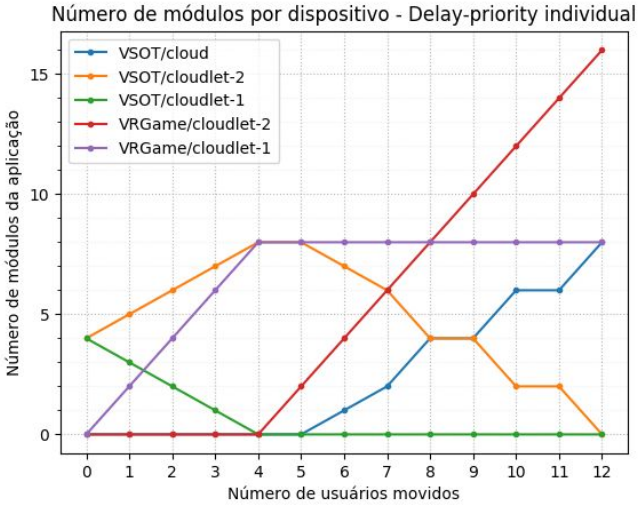


Figura 17: Número de módulos por dispositivo utilizando arquitetura multinível

Para a política de alocação *Delay Priority* individual, que apresentou os melhores resultados em arquitetura multinível de cloudlet, podemos observar, através do gráfico de módulos por dispositivo da figura 17, que a cloud passa boa parte do tempo ociosa, sendo utilizada somente quando o sexto usuário é movido. Assim, podemos concluir que, caso a estrutura fosse composta por três níveis hierárquicos de cloudlets, teríamos poder de processamento sendo desperdiçado, gerando um custo de manutenção desnecessário. É importante, portanto, estruturar a rede de forma que dispositivos de alto poder de processamento sejam compartilhados por muitos usuários, evitando ociosidade.

11. Trabalhos futuros

Além das estratégias estudadas neste trabalho, muitas outras podem ser estudadas. Ao modificarmos o simulador para permitir a análise das instâncias de cada aplicação independentemente no algoritmo edgewards, não só obtemos uma melhor utilização da rede e exploramos melhor as vantagens da computação em fog, como abrimos espaço para um novo leque de estudos possíveis. Por exemplo, com essa diferenciação dos usuários, é possível implementar agora uma estratégia com usuários premiums, na qual alguns usuários de uma aplicação possuem prioridade em relação aos demais, tendo uma maior probabilidade de serem alocados para as cloudlets.

Conforme verificamos na estratégia FCFS com o algoritmo edgewards individual, a primeira instância do VRGame foi alocada para a cloudlet, tendo um delay de aproximadamente 30ms, enquanto as demais foram alocadas para a nuvem e experimentaram um delay superior a 200ms. Com a estratégia de usuários premiums, espera-se uma vantagem semelhante para tais usuários.

Outra possível implementação seria combinar a política Communication Based Module Placement com a política Individual Module Placement. Desta forma, as instâncias da aplicação seriam consideradas de forma independente, diminuindo a quantidade de módulos a serem movidos. E, no momento em que fosse necessário mover alguns módulos, este processo seria feito de forma a minimizar a quantidade de comunicação necessária, diminuindo, portanto, o tempo de comunicação e o uso da rede. Da mesma forma, trabalhos podem ser desenvolvidos combinando a análise individual dos módulos com o estudo do

consumo energético, uso de memória, custo, entre outros. Por fim, a política baseada em comunicação pode ser estendida para utilizar mais informações dos dispositivos como larguras de banda e latência base de comunicação para que fórmulas mais complexas possam ser utilizadas no momento da escolha de quais módulos serão realocados.

12. Conclusão

O projeto tinha como objetivo o estudo de diversas soluções para o problema de alocação de recursos em fog computing, através da utilização da ferramenta iFogSim. Para isso, uma série de modificações foram realizadas no código fonte da ferramenta.

Uma das extensões realizadas no simulador adicionou suporte à alocação de módulos de instâncias distintas de forma independente, permitindo a análise das política de alocação *First Come First Served* - Individual e *Delay Priority* - Individual, que garantiram a execução de aplicações com um menor número de módulos migrados entre cloudlets e nuvem, reduzindo o delay médio e o uso da rede, além de prover maior garantia de que a política Delay Priority irá priorizar os módulos corretos, dado que, caso não seja possível alocar todas as instâncias de um módulo prioritário na cloudlet, o algoritmo ainda mantém certas instâncias no dispositivo.

Além disso, uma nova política de alocação foi criada, de forma que o tempo gasto na comunicação entre módulos fosse considerado para a alocação de tarefas nas cloudlets. Com os resultados obtidos dos experimentos executados, observou-se que para essa política ter efeito é necessário que a cloudlet tenha capacidade de processamento disponível para que de fato a escolha da política baseada em comunicação possa ser aplicada. Além disso, a proporção de processamento requerido entre os módulos é importante para que o espaço liberado na cloudlet com a movimentação de alguns módulos eleitos pela política baseada em comunicação seja o suficiente para que as próximas instâncias dos módulos que não foram transferidos ainda possam ser alocadas na cloudlet. Por fim, a política baseada em comunicação conseguiu reduzir o uso da rede nos experimentos realizados. Além disso, os cenários em que haviam arestas transmitindo uma quantidade de dados muito maior, mostraram que a política baseada em comunicação conseguiu um tempo de delay muito menor para a aplicação VSOT.

Ao final, o simulador foi modificado para permitir topologias de rede mais complexas, passando a suportar simulações com múltiplos níveis de cloudlets. Baseado nos testes realizados com três diferentes topologias, foi possível observar que as vantagens do uso de hierarquias multinível está altamente relacionado com a política de alocação utilizada. Assim, os melhores cenários para diminuir o delay na execução das aplicações simuladas foram obtidos ao combinar dois níveis hierárquicos de cloudlets com as políticas de alocação de módulos individuais, que garantem o melhor uso do poder de processamento dos dispositivos *fog*.

Com a finalização das análises concluímos que as diversas simulações realizadas foram suficientes para obtermos um melhor entendimento da computação *fog* e como diferentes estruturas de rede e políticas de alocação de recursos podem afetar a execução de aplicações móveis.

Referências

[1] Luiz F. Bittencourt, Javier Diaz-Montes, Rajkumar Buyya, Omer F. Rana and Manish Parashar, "Mobility-Aware Application Scheduling in Fog Computing", *IEEE Cloud computing (2017)*.

[2] Redowan Mahmud and Rajkumar Buyya, "Modeling and Simulation of Fog and Edge Computing Environments Using iFogSim Toolkit". *Principles and Paradigms*, chapter 17 (2019).

[3] H. Gupta, A. V. Dastjerdi, S. K. Ghosh, and R. Buyya, "iFogSim: A toolkit for modeling and simulation of resource management techniques in internet of things, edge and fog computing environments". CoRR, vol. abs/1606.02007, 2016.

[4] M. Mahmud, J. Rodrigues, K. Saleem, J. Al-Muhtadi, N. Kumar, V. Korotaev, "Towards energy-aware fog-enabled cloud of things for healthcare". *Computers & Electrical Engineering*, 67: 58-69, 2018

[5] Y. Lin, H. Shen, "Cloud fog: Towards high quality of experience in cloud gaming". *Proceedings of the 44th International Conference on Parallel Processing (ICPP '15)*, IEEE (2015), pp. 500–509.

[6] M. R. Mahmud, M.Afrin, M. A. Razzaque, M. M. Hassan, A. Alelaiwi, M. Alrubaian, "Maximizing quality of experience through context-aware mobile application scheduling in cloudlet infrastructure". *Softw. - Pract*. Exp. 46 (11) (2016) 1525–1545 spe.2392.

[7] M. Taneja, A. Davy, "Resource aware placement of IoT application modules in Fog-Cloud Computing Paradigm", *Proceedings of the IFIP/IEEE Symposium on Integrated Network and Service Management (IM'17)*, pp. 1222-1228, Lisbon, Portugal, May. 8 a 12 de 2017

[8] R. Mahmud, K. Ramamohanarao and R. Buyya, "Latency-aware Application Module Management for Fog Computing Environments". *ACM Transactions on Internet Technology (TOIT)*. DOI: 10.1145/3186592, 2018.

[9] R. Mahmud, S. N. Srirama, K. Ramamohanarao and R. Buyya, "Quality of Experience (QoE)-aware placement of applications in Fog computing environments". Journal of Parallel and Distributed Computing, DOI: 10.1016/j.jpdc.2018.03.004, 2018.

[10] O. Skarlat, M. Nardelli, S. Schulte and S. Dustdar, "Towards QoS-Aware Fog Service Placement". Proceedings of the 1st IEEE International Conference on Fog and Edge Computing (ICFEC'17), pp. 89-96, Madrid, Spain, May. 14-15, 2017.