Alocação de Recursos na Nuvem de Dispositivos

Tiago C. S. Xavier[1], Flavia C. Delicato[2], Paulo F. Pires[2], Evandro Macedo[1], Igor L. Santos[3], Claudio L. Amorim[1]

[1]Programa de Engenharia de Sistemas e Computação (PESC), Universidade Federal do Rio de Janeiro, RJ, 21941-594, Brasil

[2] Programa de Pós-Graduação em Computação (PGC), Universidade Federal Fluminense, 24210-346, Brasil

[3] Programa de Pós-graduação em Engenharia de Produção e Sistemas (PPPRO), CEFET-RJ, Rio de Janeiro, RJ, 20271-110, 229, Brasil

{tiago.cariolano,paulo.f.pires}@gmail.com, fdelicato@dcc.ufrj.br, evandro@ravel.ufrj.br, igor.santos@cefet-rj.br

Resumo. *Em uma Nuvem de Dispositivos ou Nuvem das Coisas, os recursos disponíveis na Nuvem nem sempre são capazes de atender às restrições de latência de aplicações críticas devido à grande distância entre a Nuvem e os dispositivos que originam os dados. A adoção de computação na Borda pode auxiliar a Nuvem a fornecer serviços que atendam a tais requisitos temporais. Contudo, dois fatores afetam o desempenho no nível de Borda: a heterogeneidade das aplicações e a incerteza da taxa de chegada das requisições. Neste contexto, propomos dois mecanismos: o Gerenciador de Fila, que redistribui as requisições das filas dos Nós de Borda, permitindo reduzir o tempo de espera de uma requisição; e o Alocador de Recursos, que possibilita a colaboração entre os Nós de Borda por meio de encaminhamento de requisições.*

Abstract. *In a Cloud of Things, the computing resources available in the Cloud are often unable to meet latency constraints of time critical applications because of the large distance between the Cloud and data sources (IoT devices). Adoption of Edge Computing can help the Cloud deliver services that meet time critical application requirements. However, two factors affect the performance at the Edge tier: application heterogeneity and uncertainty of the request arrival rate. In this context, we propose two mechanisms: the Queue Manager, which redistributes requests from edge nodes' queues, thus allowing reducing the request waiting time; and the Resource Allocator, which enables collaboration between edge nodes through forwarding request.*

1. Introdução

As propriedades elásticas, a enorme disponibilidade de recursos e a razão custo-benefício da computação na nuvem tornaram-na uma plataforma de hospedagem atraente para uma variedade de aplicações da Internet das Coisas (*Internet of Things* - IoT). A integração da nuvem com a IoT deu origem ao paradigma conhecido como Nuvem das Coisas (*Cloud of Things* – CoT). No entanto, os rigorosos requisitos de qualidade de serviços (*Quality of Service* - QoS) de algumas aplicações da IoT exigem desempenho previsível da nuvem e baixas latências fim-a-fim entre o usuário e a nuvem

(Shekhar e Gokhale 2017). Nesse contexto, os paradigmas recentes de *Fog* e *Edge Computing* prometem fornecer os benefícios da computação em nuvem sem incorrer em seus problemas como a alta latência e o consumo de banda do núcleo da rede, trazendo infraestrutura virtualizada e provisionamento de recursos elásticos sob demanda para mais perto dos dispositivos finais, na borda da rede.

Considerando a sinergia entre a IoT, nuvem e dispositivos de borda, um ecossistema da CoT passa a incluir três níveis: o Nível das Coisas, o de Borda e o da Nuvem. O primeiro é composto de Entidades Físicas (EFs), que correspondem a qualquer objeto físico de interesse para fins de sensoriamento ou atuação. EFs integram recursos de processamento e comunicação (sem fio) e são instrumentadas com sensores ou atuadores, tornando-se um Dispositivo da IoT. O Nível de Borda é composto por vários Nós de Borda (*edge nodes*), os quais podem variar de *gateways*, roteadores inteligentes, *micro-clusters*, até estações rádio base (ERBs) de última geração. A proximidade desses nós com os Dispositivos da IoT permite acesso mais rápido à fonte de dados. O Nível da Nuvem oferece enormes capacidades de processamento e armazenamento para os dados gerados, e um acesso onipresente e conveniente para as aplicações, que podem compartilhar os recursos seguindo um modelo sob demanda.

Os níveis de Borda e de Nuvem são altamente virtualizados. Nós de Borda mantêm e gerenciam Nós Virtuais, criados de acordo com um modelo de virtualização (I. L. Santos, et al. 2015) (Alves, et al. 2019). A existência de uma variedade de recursos físicos e virtuais disponíveis para atender a ampla gama de aplicações clientes de sistemas da CoT gera um problema de alocação de recursos. A alocação de recursos na CoT tem um objetivo similar ao da alocação de recursos na computação em nuvem, ao provisionar os recursos e selecionar os nós computacionais que irão executar a carga de trabalho gerada pelas aplicações. Entretanto, a CoT incorpora requisitos adicionais introduzidos pelos novos *players* e pela natureza específica dos dados produzidos pelos Dispositivos IoT. Portanto, apesar de amplamente investigado na área de computação em nuvem, a alocação de recursos na CoT apresenta novos desafios que exigem novas soluções, sob a forma de algoritmos e estratégias talhados para esse cenário emergente.

Neste artigo, propomos uma estratégia de alocação de recursos composta por dois algoritmos distribuídos para lidar com o problema de alocação de recursos nos Nós de Borda, os quais levam em consideração os desafios de heterogeneidade das requisições das aplicações e a incerteza da taxa de chegada de tais requisições (Nan, et al. 2018), que são inerentes ao Nível de Borda. Esses algoritmos trabalham em conjunto para promover a colaboração entre nós do Nível de Borda. O objetivo é cobrir as lacunas identificadas na literatura e contribuir com o atendimento dos requisitos de QoS de aplicações heterogêneas em cenários de CoT com alta demanda e alto grau de incerteza quanto à chegada de requisições. O primeiro algoritmo proposto é um novo Gerenciador da Fila de requisições heterogêneas baseado no conceito de roubo de tarefa e o segundo algoritmo é um Alocador Colaborativo de Recursos. Para avaliar a efetividade do mecanismo de gerenciamento de filas em conjunto com o alocador de recursos proposto, conduzimos uma avaliação experimental do tempo de resposta da aplicação e do balanceamento de carga no Nível de Borda da CoT.

Nossa principal contribuição consiste em aplicar o conceito de roubo de tarefas no compartilhamento de requisições na camada de borda da CoT, por meio do Gerenciador de Filas. Os trabalhos mais recentes relacionados com a nossa proposta e

que consideram o escalonamento de requisições na camada de borda são as descritos em (Nan, Li, Bao, Delicato, Pires, & Zomaya, 2018) e (Li, Guan, Jin, & Guo, 2019). A principal diferença desses trabalhos para nossa proposta é que os autores tratam do escalonamento de requisições internamente ao Nó de Borda, enquanto a nossa é uma solução distribuída que escalona (compartilha) requisições entre Nós de Borda vizinhos.

O restante do artigo está organizado da seguinte forma. Na Seção 2 nós apresentamos os trabalhos relacionados. Na Seção 3 nós descrevemos a arquitetura do sistema. Na Seção 4 apresentamos nossa solução de alocação de recursos e gerenciamento de requisições. Na Seção 5 apresentamos a avaliação experimental da proposta e análise dos resultados. Na Seção 6 apresentamos as considerações finais.

2. Trabalhos Relacionados

Em (Nan, Li, Bao, Delicato, Pires, & Zomaya, 2018), os autores propõem um algoritmo para selecionar as requisições que são encaminhadas para a fila de chegada de requisições do Nó de Borda ou para a fila de chegada de requisições da Nuvem. Já em (Li, Guan, Jin, & Guo, 2019), os autores propõem um gerenciador de fila que separa as requisições de acordo com o tipo de dados a fim de balancear as filas internas dos Nós Virtuais do Nó de Borda. O trabalho de (Nan, Li, Bao, Delicato, Pires, & Zomaya, 2018) não aborda o problema da heterogeneidade das requisições das aplicações e os autores em (Li, Guan, Jin, & Guo, 2019) não consideram a colaboração entre os nós da Camada de Borda Além disso, o trabalho dos autores (Li, Guan, Jin, & Guo, 2019) e (Li, Guan, Jin, & Guo, 2019) realiza o escalonamento de requisições internamente ao Nó de Borda, enquanto nosso Gerenciador de Requisições baseado em roubo de tarefas escalona as requisições entre diferentes Nós de Borda.

Em (Noreikis, Xiao e Jiang 2019), os autores propõem uma nova solução de planejamento de capacidade de borda baseada em teoria de filas, que maximiza a utilização de recursos por meio de uma precisa estimativa da capacidade de CPU e GPU. Diferentemente do trabalho de Noreikis et al., nosso trabalho é baseado em um modelo de virtualização não sendo restrito a otimização do uso de CPU e GPU. O trabalho de (Beraldi e Alnuweiri 2019) propõe um algoritmo de balanceamento de carga ponto a ponto distribuído que executa sobre um conjunto de nós de borda cooperativos e autônomos. O algoritmo de Beraldi e Alnuweiri utiliza virtualização orientada a containers, enquanto nossa proposta considera a virtualização orientada a dados. Além disso, o trabalho de Beraldi e Alnuweiri não considera a incerteza na taxa de chegada e a heterogeneidade das requisições, em relação aos tipos de dados requisitados.

Em (Zhao, Barijough e Gerstlauer 2018), os autores propõem um arcabouço para execução adaptada e distribuída de aplicações de inferência baseadas em redes neurais convolucionais para Nós de Borda com restrições de recursos. A proposta realiza uma abordagem de roubo de tarefas distribuído para permitir a distribuição dinâmica da carga de trabalho e o equilíbrio no tempo de execução. Contudo, Zhao et al. não tratam a heterogeneidade das requisições em relação aos dados requisitados. Em (Fernando, Loke e Rahayu 2019), os autores apresentam um modelo de compartilhamento de carga de trabalho, usando uma adaptação do método de roubo de tarefas para equilibrar tarefas independentes entre nós móveis heterogêneos. Enquanto Fernando et al. consideram a heterogeneidade dos dispositivos e das cargas de trabalho das aplicações, o nosso trabalho considera a heterogeneidade do tipo de dados requisitado pelas aplicações, em um modelo de virtualização baseado em dados.

3. Arquitetura do Sistema

O Alocador Colaborativo de Recursos é o algoritmo responsável por alocar requisições para nós virtuais com o objetivo de reduzir o custo de alocação e utilizar de maneira mais eficiente os recursos dos nós no Nível de Borda. O modelo de virtualização empregado pelo Alocador Colaborativo de Recursos é baseado no trabalho de (Santos, et al. 2019), o qual aloca a requisição para o Nó Virtual que minimiza o custo envolvido no processo de virtualização. Tal modelo fornece uma estimativa do custo do Nó Virtual para atender uma requisição, o que possibilita uma alocação de recursos mais eficiente. No entanto, nosso Alocador de Recursos é colaborativo, pois permite que os Nós de Borda realizem colaboração entre si, encaminhando requisições para serem alocadas em Nós Virtuais de Nós de Borda vizinhos que sejam capazes de atendê-las.

O novo Gerenciador de Fila é inspirado na abordagem de roubo de tarefas (*work stealing*), um paradigma empregado em sistemas distribuídos ou de programação paralela, desenvolvido inicialmente no trabalho de (Blumofe e Leiserson 1999). Em um sistema de roubo de tarefas, um processo chamado ladrão (*thief*) pode "roubar" uma tarefa (requisição) de outro processo chamado vítima (*victim*) e executar o respectivo processamento daquela tarefa. Ao aplicar o paradigma de roubo de tarefas para gerenciamento de filas na CoT, um Nó de Borda ocioso pode "sondar" as filas dos nós vizinhos e, em alguns casos, obter tais requisições e imediatamente começar a realizar trabalho útil, o que aumenta a distribuição de carga de trabalho nos Nós de Borda e reduz o tempo de resposta para a aplicação. Esses dois mecanismos permitem que um Nó de Borda atenda aos requisitos de QoS, mesmo em um cenário de aplicações heterogêneas e de alta demanda dos usuários.

3.1. Arquitetura de Três Níveis da CoT

Os elementos que compõem o ecossistema da CoT considerado neste trabalho são organizados segundo a arquitetura de três níveis ilustrada na Figura 1. O Nível das Coisas contempla qualquer dispositivo que seja instrumentado com conectividade com a Internet e que possa prover dados (por meio de sensores) ou atuar sobre o ambiente em que esteja inserido (por meio de atuadores). No mais alto nível da arquitetura está o Nível da Nuvem, o qual possui o máximo de recursos computacionais possíveis para atendimento às requisições das aplicações. De maneira intermediária entre os Níveis de Nuvem e das Coisas está o Nível de Borda, no qual estão os equipamentos (Nós de Borda) que conseguem prover recursos computacionais mais próximos fisicamente da origem dos dados para as aplicações da IoT.

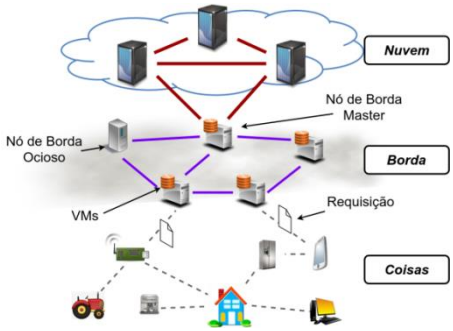

Figura 1 – Arquitetura de Três Níveis

Embora existam diversas arquiteturas em ambientes da CoT, nós propomos uma arquitetura de três camadas, em que o Nível de Borda é organizado de forma hierárquico, com dois tipos de nós: Nó de Borda Mestre (*Master*) e Nó de Borda Escravo (*Slave*). O Nó *Master* é o único que possui conexões com o Nível da Nuvem e é responsável por gerenciar o processo de alocação de recursos e atender às requisições das aplicações. O Nó *Slave* pode apenas atender às requisições e encaminhá-las para outros nós. O Nó *Master* é o nó que minimiza as distâncias geográficas para todos os outros nós do Nível de Borda e é escolhido por meio de um algoritmo de caminho mínimo. O objetivo da organização hierárquica do Nível de Borda é reduzir o atraso de propagação devido à distância física entre os Nós de Borda. Cada Nó de Borda é um ponto de entrada de aplicações. Cada Dispositivo da IoT do Nível das Coisas se conecta a um único Nó de Borda que estiver mais próximo geograficamente.

Apesar de nossa proposta considere apenas um Nó de Borda Mestre, ela pode ser estendida para cenários com mais de um Nó de Borda Mestre, sem necessidade de grande modificação do Alocador de Recursos ou do Gerenciador de Fila, bastando apenas a identificação de nó mestre ou escravo em cada Nó de Borda. O Alocador de Recursos apenas exige que pelo menos um nó mestre seja conectado à nuvem. Já o Gerenciador de Filas é indiferente quanto à topologia da rede ou a categorização do nó de borda como mestre/escravo, pois ele tenta distribuir as requisições de um nó de borda para seus vizinhos, independentemente de seu papel. No cenário de falha de algum Nó de Borda Mestre, mesmo para o cenário com apenas um nó desta categoria, consideramos que algum vizinho escravo assume seu lugar. O nó vizinho infere a falha do Nó de Borda Mestre por meio do envio de mensagens de broadcast periódicas.

3.2. Arquitetura do Nó de Borda

A Figura 2 apresenta os componentes de um Nó de Borda. As requisições chegam ao ponto de acesso por meio dos Dispositivos da IoT conectados ao Nó de Borda, ou através de outros Nós de Borda, quando ocorre colaboração no Nível de Borda. A taxa de chegada de requisições (λ) não é conhecida *a priori*, bem como a taxa com que outros nós enviam requisições para o Nó de Borda em questão. Quando uma requisição chega ao ponto de acesso, o Gerenciador de Fila a encaminha para a Fila de Requisições, onde ela aguarda para ser processada pelo Alocador de Recursos. Cada requisição possui um tipo de dados requisitado e o Nó de Borda possui um conjunto de tipos de dados determinados de acordo com os Dispositivos da IoT conectados ao Nó de Borda. Uma requisição é atendida por um Nó de Borda apenas se seu tipo de dado corresponde a algum tipo de dado pertencente ao Nó de Borda. Devido aos diferentes tipos de dado e à desconhecida taxa de chegada das requisições, uma quantidade excessiva de requisições pode permanecer na fila do Nó de Borda esperando até que o Alocador de Recursos possa processá-las. O Alocador de Recursos pode: (i) encaminhar a requisição para a Nuvem, (ii) para um Nó de Borda vizinho, ou (iii) pode alocá-la em um Nó Virtual local.

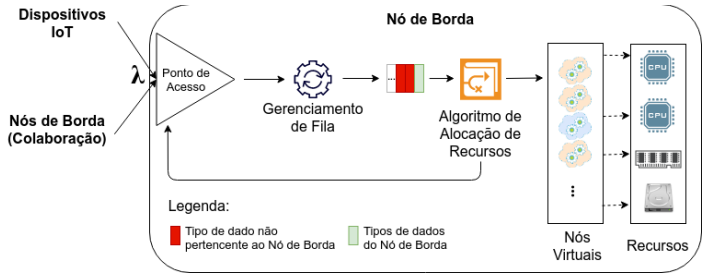

Figura 2 – Arquitetura do Nó de Borda na CoT

Em alguns cenários de alta demanda, requisições (em Vermelho na Figura 2) com tipo de dados não pertencentes ao Nó de Borda permanecem por um longo período na Fila de Requisições. Para resolver este problema, o Gerenciador de Fila proposto neste trabalho atua de forma proativa, roubando requisições das filas dos nós vizinhos que sejam compatíveis com o conjunto de tipos de dados do Nó de Borda que está com a fila vazia, passando a processá-las. Dessa forma, há uma redução no número de requisições não pertencentes ao Nó de Borda que estão na Fila de Requisições. O Alocador de Recursos obtém cada requisição da fila e a aloca para um novo Nó Virtual, ou para algum Nó Virtual já instanciado, ou encaminha essa requisição de volta para o Ponto de Acesso para ser encaminhada para outro Nó de Borda ou para a Nuvem.

3.3. Modelo de Rede

Em nossa proposta, o Nível de Borda é formalmente representado como um grafo $G = (E, L, d)$, em que $E = \{e_i \forall i \in 0..n\}$ é o conjunto de n Nós de Borda, $L = \{(e_i, e_j) \forall i, j \in 1..n \wedge e_i, e_j \in E\}$ é o conjunto de enlaces de comunicação entre os Nós de Borda e d é a função de distância euclidiana entre dois Nós de Borda $d: E \times E \rightarrow \mathbb{R}$. A seguir são definidos o Nó de Borda, o Nó Virtual, a requisição e carga de trabalho. A Seção 4.1 apresenta o modelo de Nó Virtual.

Definição do Nó de Borda: Um Nó de Borda e_i é definido como uma tupla $e_i = (C_{e_i}, M_{e_i}, \tau_{e_i}, D_{e_i}, B_{e_i}, Q_{e_i})$, para $i > 0$, onde C_{e_i} é a taxa de processamento, M_{e_i} é a quantidade de memória, $\tau_{e_i} = \{MASTER, SLAVE\}$ é o tipo de Nó de Borda, D_{e_i} é o conjunto de tipo de dados que pode ser fornecido pelo Nó de Borda, e Q_{e_i} é a fila de requisição do Nó de Borda. A definição de cada tipo de dado $\delta_{e_{ik}} \in D_{e_i}, k > 0$, é dada pelo conjunto de Dispositivos da IoT conectados a e_i.

Definição do Nó Virtual: Cada Nó de Borda e_i possui um conjunto de μ nós virtuais $V_{e_i} = \{v_{ij} \mid i \in 0..n \ \wedge j \in 0..\mu \ \wedge e_i \in E\}$ e cada Nó Virtual é definido como uma tupla $v_{ij} = (m_{v_{ij}}, \delta_{ij})$, em que $m_{e_{ij}}$ é a memória do Nó de Borda e_i que é destinada ao Nó Virtual v_{ij}, e $\delta_{ij} \in D_{e_i}$ é o tipo de dado de v_{ij}.

Definição da requisição: Uma requisição r_k submetida por uma aplicação a partir de um Dispositivo da IoT chega em algum Nó de Borda e é definida como uma tupla $r_k = (I_{r_k}, m_{r_k}, \delta_{r_k})$, para $k > 0$, em que I_{r_k} é o número de instruções para serem executadas , m_{r_k} é a quantidade de memória requerida, δ_{r_k} é o tipo de dado da requisição. O momento t_{0r_k} é o instante de tempo em que a requisição é gerada pelo dispositivo. O momento t_{fr_k} é o instante em que o dispositivo recebe a resposta com o dado da requisição. n_r é a quantidade total de requisições geradas.

Definição de carga de trabalho: A carga de trabalho (*workload*) de um Nó de Borda e_i é definida como:

$$W_{e_i} = \sum_{k=1}^{n_r} \frac{I_{r_k}}{C_{e_i}} \alpha_{e_i r_k}, \quad (1)$$

em que $\alpha_{e_i r_k} = \begin{cases} 1, se\ r_k\ foi\ atendida\ por\ e_i \\ 0, se\ \ r_k\ não\ foi\ atendida\ por\ e_i \end{cases}$ é uma variável de decisão, a qual indica se a requisição r_k foi atendida por e_i. A média da carga de trabalho da rede é $\overline{W} = \frac{1}{n}\sum_{i=1}^{n} W_{e_i}$.

4. Alocação de Recursos Colaborativa e Gerenciamento de Filas na Borda

4.1.Modelo de Virtualização

Quando uma requisição r_k chega ao Nó de Borda e_i, o Nó de Borda decide se r_k será atendida localmente, verificando as seguintes restrições:

$$m_{r_i} < M_{e_i} - \sum_{j=0}^{\mu} m_{v_{ij}} \quad (2) \qquad \delta_{r_k} \in D_{e_i} \quad (3)$$

As Equações (2) e (3) garantem que os recursos disponíveis de memória são suficientes e o tipo de dado da requisição pode ser atendido por e_i. Se pelo menos uma dessas restrições não for válida, então o Nó de Borda encaminha essa requisição para algum vizinho. Se as restrições são válidas, então o Nó de Borda irá escolher Nó Virtual que apresenta o menor custo para atender r_k.

O Nó Virtual v_{ij} escolhido para atender r_k é aquele que minimiza a função de custo $C(v_{ij}, r_k)\ \forall v_{ij} \in V_{e_i}$. A função de custo corresponde à soma do valor estimado de atender à requisição r_k pelo nó v_{ij} e é definida como:

$$C(v_{ij}, r_k) = C_r(v_{ij}) + C_i(v_{ij}) + C_p(v_{ij}, r_k) + C_u(v_{ij,} r_k) + C_q(v_{ij}, r_k) \quad (4)$$

onde $C_r(v_{ij})$ é o custo de reconfiguração, $C_i(v_{ij})$ é o custo de instanciação, $C_p(v_{ij})$ é o custo de processamento, $C_u(v_{ij}, r_k)$ é o custo de atualização, correspondente ao tempo médio para o Nó Virtual obter os dados necessários dos Dispositivos da IoT, e $C_q(v_{ij}, r_k)$ é o custo de fila.

Quando o Nó de Borda e_i recebe r_k, e_i necessita escolher o Nó Virtual para atender r_k. Reconfigurar o Nó Virtual pode ser mais vantajoso do que instanciá-lo. Então, o custo de reconfiguração é definido como:

$$C_r(v_{ij}) = \begin{cases} 0 \quad , \quad if\ v_{ij}\ \text{é instanciado.} \\ 0 \quad , \quad if\ v_{ij}\ \text{é instanciado} \wedge \text{reconf. NÃO é necessária.} \\ \Delta_r(v_{ij}), \quad if\ v_{ij}\ \text{é instanciado} \wedge \text{reconf. é necessária.} \end{cases} \quad (5)$$

onde $\Delta_r(v_{ij})$ é o tempo médio de reconfiguração do Nó Virtual v_{ij}. O custo de instanciação de um Nó Virtual é dado por:

$$C_i(v_{ij}) = \begin{cases} 0 \quad \text{, se instancia novo Nó Virtual.} \\ \Delta_i(v_{ij}), \text{ se reutiliza o Nó Virtual.} \end{cases} \quad (6)$$

onde $\Delta_i(v_{ij})$ é o tempo médio de instanciação de um Nó Virtual em e_i.

$C_p(v_{ij})$ é uma estimativa do custo de processamento da requisição no Nó Virtual v_{ij} e é definido como:

$$C_p(v_{ij}, r_k) = \frac{I_{r_k}}{C_{e_i}} \quad (7)$$

$C_u(v_{ij}, r_k)$ é uma estimativa do tempo médio de atendimento do Nó Virtual v_{ij} atender uma requisição.

$C_q(v_{ij}, r_k)$ é o custo decorrente da espera de atendimento da requisição r_k aguarda até chegar em e_i. Este custo é calculado como:

$$C_q(v_{ij}) = \frac{1}{m}\sum_{k=0}^{m} \mathrm{T}_q(v_{ij}, t_{0r_k}) \quad (8)$$

onde $T_q(v_{ij}, t_{0r_k})$ corresponde à diferença de tempo desde o momento em que a requisição foi gerada t_{0r_k}, até o momento em que a requisição chega ao Nó de Borda. m é o número de requisições que chegaram no Nó de Borda até o momento de chegada da requisição r_k em e_i.

4.2. Alocador Colaborativo de Recursos

O Alocador de Recursos faz uso de um modelo de virtualização que abstrai a infraestrutura física para fornecer os dados solicitados para a aplicação. Nós Virtuais (NVs) possuem um valor de custo, usado pelo Alocador para selecionar o melhor NV para atender uma dada requisição. O Alocador determina de forma distribuída onde cada requisição de aplicação enviada para o sistema CoT será atendida. Quando uma requisição r_k chega ao Nó de Borda e_i, o Alocador faz uma estimativa do custo de alocar r_k para cada NV já instanciado e o custo da possibilidade de instanciar um novo NV. A partir destes custos, o Alocador escolhe o NV com menor custo para alocar a requisição r_k. Se nenhum Nó Virtual local pertencente à e_i pode atender r_k, então a requisição é encaminhada para algum Nó de Borda vizinho.

A Figura 3(a) apresenta o algoritmo do Alocador Colaborativo de Recursos. Inicialmente o Nó de Borda chama a função "*search_best_vn_locally*" a qual retorna uma lista de nós virtuais capazes de atender a requisição r_k. A seguir, a função "*compute_best_vn_locally*" é chamada para retornar o Nó Virtual com menor custo para atender r_k. A função "*compute_best_vn_locally*" é responsável por calcular o custo $C(v_{ij}, r_k)$ da Equação (4). Se algum Nó Virtual v_{ij} é retornado então r_k é alocada para v_{ij}, caso contrário o Alocador chama a função "*search_possible_vn_slave*", que estabelece uma comunicação com cada Nó de Borda vizinho procurando algum candidato para atender a requisição. Caso algum vizinho possa atender r_k, então o Nó de Borda lhe encaminha a requisição. Se nenhum nó vizinho pode atender r_k, o Nó de Borda encaminha a requisição ou para a Nuvem, caso seja um Nó de Borda *Master*, ou para o seu nó pai (*Master*), caso seja um Nó de Borda *Slave*. O Nó de Borda também é interrompido quando um Nó Virtual termina sua execução. Nesse caso, o Nó de Borda atualiza as informações dos recursos que estavam sendo ocupados pelo Nó Virtual.

Algorithm 1: Alocador de Recursos Colaborativo

```
1  while true do
2      event ← receiveEvent();
3      if event =< REQUEST, e_j, r_k > then
4          vnList ← search_best_vn_locally (r_k);
5          v_ij ←compute_best_vn (vnList);
6          if v_ij is found then
7              allocate (v_ij, r_k);
8          else
9              bestSlave = search_possible_vn_slave (r_k);
10             if bestSlave is found then
11                 foward_request (bestSlave, v_ij, r_k);
12             else
13                 if τ_ei = MASTER then
14                     forward_request (cloud, r_k);
15                 if τ_ei = SLAVE then
16                     foward_request (parent, r_k);
17     if event =< VNFINISH, v_ij > then
18         v_ij ← (VN) event.message;
19         update_resources (v_ij);
```

(a)

Algorithm 2: Roubo de Requisições Heterogêneas

```
1  function Gerenciador_de_Fila ()
2      while True do
3          event ← receiveEvent();
4          if event =< REQUEST, e_j, r_k > then
5              push_queue(r_k);
6          if event =< VNFINISH, v_ij > then
7              c_ei ← busy_cores() ;
8              if size_queue() = 0 ∧ c_ei ≤ c_max then
9                  foreach e_j ∈ N_ei do
10                     send ( < PROBE, e_j, D_ei, c_max − c_ei >);
11         if event =< PROBE, e_j, D_ej, c_ej > then
12             i ← 0;
13             foreach r ∈ Q_ei do
14                 if i < c_ej ∧ δ_r ∈ D_ej ∧ c_ei < c_ej then
15                     remove_queue(r);
16                     send (< REQUEST, e_i, r >);
17                     i ← i + 1;
```

(b)

Figura 3 – (a) Algoritmo do Alocador Colaborativo de Recursos e Distribuído; (b) Gerenciador de Filas

4.3. Gerenciamento de Filas baseado em Roubo de Requisições

A Figura 4 ilustra um exemplo do mecanismo de roubo de requisições heterogêneas entre quatro Nós de Borda. Divide-se o funcionamento do mecanismo em duas etapas: Fase de Sondagem e Fase de Roubo. Na primeira, o Gerenciador de Fila do Nó de Borda 1 ocioso (fila de requisições vazia) envia uma mensagem de sondagem ($PROBE$) para todos os seus vizinhos, contendo o número de identificação dos tipos de dados do nó emissor da mensagem e a informação da quantidade de núcleos de processamento disponíveis. Na Fase de Roubo, o Gerenciador de Fila dos vizinhos verifica a fila de requisições e envia para o Nó de Borda ocioso apenas requisições que possuem o mesmo tipo de dado dele (requisições da cor azul). O Nó de Borda recebe essas requisições e imediatamente passa a processá-las com o Algoritmo de Alocação de Recursos, cujo objetivo é fazer com que a carga do sistema seja mais bem distribuída.

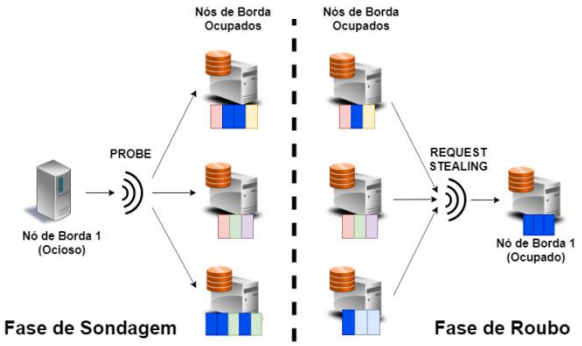

Figura 4 – Roubo de requisições heterogêneas entre Nós de Borda na CoT

O Gerenciador de Fila é responsável por gerenciar as requisições ou mensagens que chegam ao Ponto de Acesso do Nó de Borda e seu algoritmo é descrito na Figura 3(b). Quando uma requisição r_{e_j} (mensagem de tipo $REQUEST$) chega ao Nó de Borda e_i, o Gerenciador de Fila insere r_{e_j} na última posição de Q_{e_i}. Essa requisição permanece na fila até que, ou o Alocador de Recursos a processe, ou o Gerenciador de Fila a encaminhe para outro Nó de Borda. Quando um dos núcleos do Nó de Borda fica desocupado, ou seja, na ocorrência do evento $VNFINISH$, o Gerenciador de Fila é

acionado. Ele verifica quantos núcleos estão ocupados do Nó de Borda (função "*busy_cores"*) e, caso exista algum núcleo desocupado e a fila de requisições esteja vazia, envia uma mensagem do tipo "PROBE" para os vizinhos.

Quando o Gerenciador de Fila do nó e_i recebe uma mensagem de sondagem do tipo PROBE do nó emissor e_j, o nó e_i verifica todas as requisições presentes em sua fila. Cada requisição r que possui o tipo de dado δ_r pertencente a D_{e_j} é encaminhada para o Nó de Borda emissor. No entanto, o Gerenciador de Fila envia uma quantidade de requisições limitada à quantidade de núcleos de processamento disponíveis pelo Nó de Borda emissor c_{e_i}. O algoritmo do Gerenciador de Fila pode ser visto na Figura 3(b).

5. Avaliação Experimental

A metodologia GQM (*Goal Question Metric*) (Basili 1992) foi utilizada para planejar a avaliação experimental. O objetivo G1 é analisar o desempenho do algoritmo de alocação de recursos em colaboração com o mecanismo de roubo de requisições, com o propósito de comparar a redução obtida no tempo de resposta das requisições em relação ao algoritmo de alocação de recursos tradicional (sem roubo). O G2 consiste em analisar o algoritmo de alocação de recursos em colaboração com o mecanismo de roubo de requisições, comparando o balanceamento da carga de trabalho obtido no Nível de Borda em relação ao algoritmo de alocação de recursos tradicional. O G3 é analisar o algoritmo de alocação de recursos em colaboração com o mecanismo de roubo de requisições, comparando o número de requisições não atendidas obtido no Nível de Borda em relação ao algoritmo de alocação de recursos tradicional. Para cada um dos objetivos (G1, G2), nós definimos questões correspondentes (Q1, Q2 e Q3), as quais estão descritas na Tabela 1.

Tabela 1- Questões e Objetivos

Questão	Descrição da Questão	Objetivo
Q1	**O mecanismo de roubo de tarefas para o algoritmo de alocação de recursos ajuda a reduzir o tempo de resposta do atendimento de requisição quando comparado com um algoritmo de alocação de recursos tradicional?**	**G1**
Q2	**O mecanismo de roubo de tarefas para o algoritmo de alocação de recursos ajuda a balancear a carga de trabalho quando comparado com um algoritmo de alocação de recursos tradicional?**	**G2**
Q3	**O mecanismo de roubo de tarefas para o algoritmo de alocação de recursos baseado em roubo de tarefas ajuda reduzir a quantidade de requisições não atendidas?**	**G3**

Tabela 2 - Resumo das métricas utilizadas para responder às questões

Métrica	Descrição da métrica	Questão
MKR	**Makespan da Requisição –Diferença absoluta entre o momento quando a requisição é gerada pela aplicação até o momento em que a aplicação recebe a resposta.**	**Q1**
BCR	**Balanceamento de Carga da Rede– Desvio padrão da carga de trabalho total dos Nós de Borda.**	**Q2**
NRN	**Número de Requisições Não-atendidas – Requisições que não foram atendidas no nível de Borda**	**Q3**

A fim de responder às questões acima, nós definimos um conjunto de métricas apresentado na Tabela 2. A primeira métrica, MKR é definida como:

$$MKR = \frac{1}{n_r}\sum_{k=1}^{n_r}\left[t_{fr_k} - t_{0r_k}\right] \quad (9)$$

A segunda métrica é a BCR, utilizada em (Wang, et al. 2019), a qual mede a dispersão da carga de. A partir da definição de carga de trabalho W_{e_i} de um Nó de Borda e_i dada pela Equação (1), o BCR é definido conforme a equação (9). Quanto menor é o valor do BCR, mais balanceada é a carga de trabalho de cada Nó de Borda.

$$BCR = \sqrt{\frac{\sum_{i=1}^{n}(W_{e_i} - \overline{W})^2}{n}} \quad (10)$$

A métrica NRN é definida como

$$NRN = n_r - \sum_{k=1}^{n_r}\sum_{i=1}^{n}\left[\alpha_{e_i r_k}\right] \quad (11)$$

5.1.Configuração Experimental

Os resultados foram obtidos a partir de simulações por meio do simulador YAFS (Lera, Guerrero e Juiz 2019). O YAFS é um simulador de código-fonte aberto e foi utilizado pois disponibiliza bibliotecas de grafos e de simulação de eventos discretos que permitem a comunicação entre Nós de Borda para qualquer topologia, o que é essencial para nossa arquitetura. Os experimentos foram repetidos 30 vezes e os resultados correspondem à média das 30 repetições. No Nível de Borda, os *links* de comunicação entre os nós foram aleatoriamente e apenas o Nó de Borda *Master* possui conexão com a Nuvem. Adotamos a mesma configuração de largura de banda dos *links* descrita em (Yousefpour, et al. 2018) nos experimentos. Assumimos que um Dispositivo IoT do Nível das Coisas se comunica com um Nó de Borda com largura de banda de 54 Mbps e que as conexões entre nós do Nível de Borda são *links* de 100 Mbps e as conexões entre o Nó de Borda *Master* e a Nuvem são*links* de 1Gbps. Assumimos que um Dispositivo da IoT possui 16 MHz de taxa de CPU e 32 KB de memória. No Nível de Borda, os nós foram configurados com 4 núcleos de 1 GHz de taxa de CPU e 2 GB de memória total. Consideramos que a capacidade de processamento e de memória do Nível da Nuvem são suficientes para atender às requisições que chegam até lá. Cada Dispositivo da IoT possui um único tipo de dado, e os tipos foram distribuídos aleatoriamente entre todos os Dispositivos da IoT. Cada tipo de dado possui um tempo médio de atualização do dado, correspondente ao tempo para o Dispositivo IoT obtê-lo do sensor e enviá-lo, configurado com os valores [0.025 ms, 0.125 ms, 0.5 ms] (I. L. Santos, et al. 2019)

Todos os Nós de Borda são pontos de entrada de requisições de aplicações. O número total de requisições geradas é distribuído entre os Nós de Borda uniformemente. A taxa de geração de requisições é exponencialmente distribuída durante todo o tempo de simulação. Os tipos de dados foram uniformemente distribuídos entre todas as requisições. O tamanho de cada requisição foi de 80KB (Yousefpour, et al. 2018). Com relação às características das requisições, consideramos um típico cenário Borda-Nuvem, onde diferentes usuários enviam requisições pertencendo a domínios de aplicações heterogêneos. Isto significa que as aplicações são heterogêneas, demandam diferentes tipos de dados de sensoriamento e possuem diferentes requisitos de QoS. O comportamento do sistema é agnóstico ao domínio da aplicação.

5.2.Resultados

Foram executados dois experimentos para responder as questões de pesquisa planejadas no GQM. O primeiro experimento (E1) consiste em um cenário onde o número de tipos

de dados é fixado em 16, pois o foco é analisar o impacto da variação do número de Nós de Borda e do número de requisições sobre as métricas MKR, NRN e BCR. No segundo experimento (E2), mantém-se o número de Nós de Borda (8 nós) e varia-se o número de tipos de dados fornecidos pelos dispositivos. O objetivo deste experimento é analisar como as métricas NRN e BCR se comportam perante a existência de heterogeneidade do tipo de dado. Cabe ressaltar que devido ao fato da distribuição dos tipos de dados pelos Nós de Borda ser aleatória, os resultados dessa métrica são enviesados pela aleatoriedade, logo o E2 não apresenta valores para MKR.

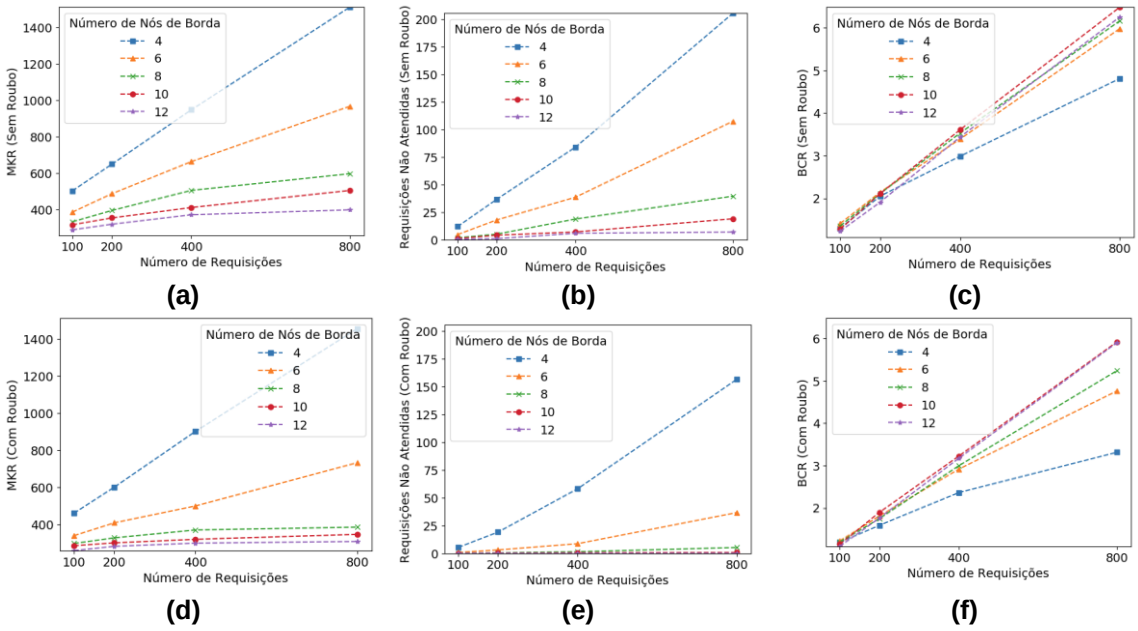

Figura 5 – Variação do Número de Nós de Borda para MKR, NRN e BCR

Os resultados coletados para E1 são apresentados na Figura 5. Nota-se que nas duas abordagens (com e sem roubo) o valor de MKR e NRN aumenta inversamente proporcional com relação ao número de Nós de Borda. Esse comportamento ocorre, pois quanto menor o número de Nós de Borda, maior é a degradação de desempenho da alocação de recursos e gerenciamento de filas. Nota-se que o desempenho para MKR e NRN para a abordagem "Com Roubo" é superior do que a abordagem "Sem Roubo" em todos os cenários, exceto no cenário onde o número de Nós de Borda é igual a 4. Este comportamento para 4 Nós de Borda ocorre, pois quando existem poucos recursos na rede (4 nós, neste caso específico), passa a não importar mais o algoritmo de alocação ou o gerenciamento de fila, pois não existem mais recursos para as requisições serem alocadas. Então, a diferença entre "Sem Roubo" e "Com Roubo" tende a diminuir significativamente. Mais detalhadamente, para um número de Nós de Borda igual a 4, alguns tipos de dados passam a não serem providos por nenhum nó, pois os tipos de dados são atribuídos aleatoriamente aos Nós de Borda. Então, muitas requisições são encaminhadas para a nuvem invariavelmente ou nem sequer são atendidas, degradando o desempenho do sistema independentemente da estratégia de alocação ou gerenciamento de filas. Com relação a métrica BCR, os gráficos da Figura 5 (c) e (f) mostram que o valor de BCR para a abordagem "Com Roubo" é menor (melhor balanceamento) em todos os cenários. Contudo, as figuras também indicam uma tendência de diminuição da vantagem da abordagem "Com Roubo" quando há aumento

do número de Nós de Borda. Contudo, em todos os cenários o valor de BCR é menor para “Com Roubo” do que para “Sem Roubo”, o que significa que a abordagem baseada em roubo de requisições apresenta um melhor balanceamento de carga com relação à abordagem tradicional. Desta forma, os resultados do experimento E1 indicam que a abordagem “Com Roubo” apresenta desempenho superior para todas as métricas, respondendo positivamente as questões de pesquisa Q1, Q2 e Q3.

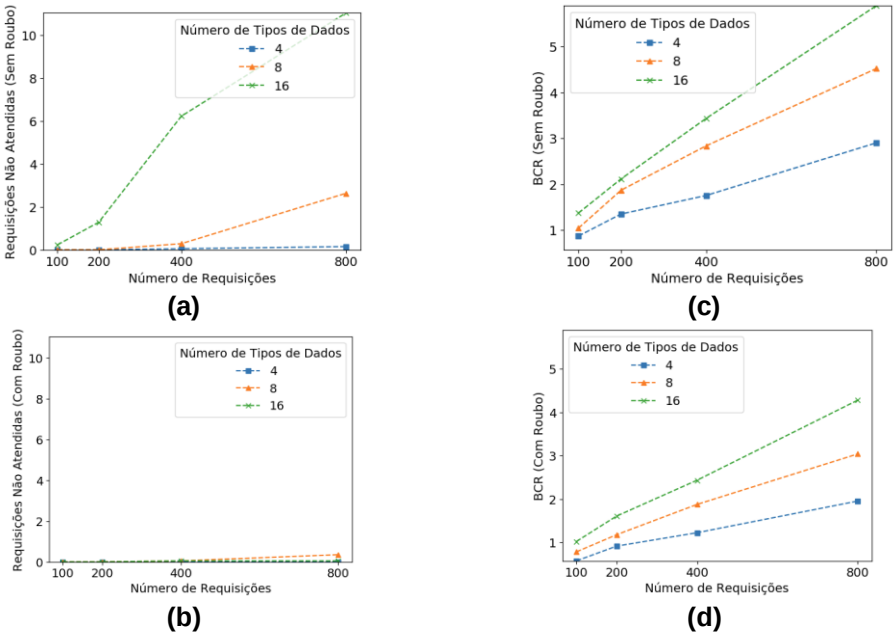

Figura 6 – Variação do Número de tipos de dados para MKR, NRN e BCR

6. Considerações Finais

Este trabalho apresentou uma nova estratégia conjunta para gerenciamento de filas baseado em roubo de requisições e alocação de recursos para a Nuvem de Coisas e avaliou sua efetividade em comparação com a abordagem tradicional (sem roubo de tarefas). Os resultados de simulações mostram que a estratégia proposta foi capaz de reduzir o tempo de resposta para a aplicação, bem como aumentar o balanceamento de carga do sistema. Como trabalho futuro, pretendemos analisar a provisão de dados de tipos não atendidos diretamente pelos Dispositivos IoT conectados a um Nó de Borda através da colaboração entre os Nós vizinhos, explorando mecanismos de *caching*.

Agradecimentos

Esse trabalho foi parcialmente financiado pela FAPESP (2015/24144-7). Flavia C. Delicato, Paulo F. Pires e Claudio Amorim são bolsistas (PQ e DT) do CNPq.

Referências

Alves, Marcelo Pitanga, Flavia C. Delicato, Igor L. Santos, e Paulo F. Pires. “LW-CoEdge: a lightweight virtualization model and collaboration process for edge computing.” *World Wide Web*, 11 2019.

Basili, Victor R. *Software Modeling and Measurement: The Goal/Question/Metric Paradigm.* Tech. rep., Univ. of Maryland at College Park, 1992.

Beraldi, Roberto, e Hussein Alnuweiri. "Exploiting Power-of-Choices for Load Balancing in Fog Computing." *2019 IEEE ICFC.* 2019. 80-86.

Blumofe, Robert D., e Charles E. Leiserson. "Scheduling Multithreaded Computations by Work Stealing." *J. ACM* (ACM) 46 (1999): 720-748.

Fernando, Niroshinie, Seng W. Loke, e Wenny Rahayu. "Computing with Nearby Mobile Devices: A Work Sharing Algorithm for Mobile Edge-Clouds." *IEEE Trans. on Cloud Computing*, 2019: 329-343.

Lera, I., C. Guerrero, e C. Juiz. "YAFS: A Simulator for IoT Scenarios in Fog Computing." *IEEE Access* 7 (2019): 91745-91758.

Nan, Yucen, Wei Li, Wei Bao, Flavia C. Delicato, Paulo F. Pires, e Albert Y. Zomaya. "A dynamic tradeoff data processing framework for delay-sensitive applications in Cloud of Things systems." *Journal of Parallel and Distributed Computing* 112 (2018): 53-66.

Noreikis, Marius, Yu Xiao, e Yuming Jiang. "Edge Capacity Planning for Real Time Compute-Intensive Applications." *2019 IEEE ICFC.* 2019. 175-184.

Santos, I. L., L. Pirmez, F. C. Delicato, S. U. Khan, e A. Y. Zomaya. "Olympus: The Cloud of Sensors." *IEEE Cloud Computing* 2 (3 2015): 48-56.

Santos, Igor L., et al. "Zeus: A resource allocation algorithm for the cloud of sensors." *Future Generation Computer Systems* 92 (2019): 564-581.

Santos, Igor, Flávia Delicato, Paulo Pires, Marcelo Alves, Ana Oliveira, e Tiago Calmon. "Data-centric resource management in edge-cloud systems for the iot." *Conference: Very Large Internet of Things (VLIOT 2019).* 2019. 29-46.

Shekhar, S., e A. Gokhale. "Dynamic Resource Management Across Cloud-Edge Resources for Performance-Sensitive Applications." *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID).* 2017. 707-710.

Wang, Shangguang, Yali Zhao, Jinlinag Xu, Jie Yuan, e Ching-Hsien Hsu. "Edge server placement in mobile edge computing." *Journal of Parallel and Distributed Computing* 127 (2019): 160-168.

Yousefpour, A., G. Ishigaki, R. Gour, e J. P. Jue. "On Reducing IoT Service Delay via Fog Offloading." *IEEE Internet of Things Journal* 5 (4 2018): 998-1010.

Zhao, Zhuoran, Kamyar Mirzazad Barijough, e Andreas Gerstlauer. "DeepThings: Distributed Adaptive Deep Learning Inference on Resource-Constrained IoT Edge Clusters." *IEEE Trans. on Computer-Aided Design of Int. Circuits and Systems*, 2018: 2348-2359.