NEMESIS: Mecanismo para Formação de Nuvens Veiculares Baseado em Previsão de Mobilidade

Joahannes B. D. da Costa[1], Wellington V. Lobato Junior[1], Allan M. de Souza[1], Eduardo Cerqueira[2], Denis Rosário[2], Leandro A. Villas[1]

[1]Instituto de Computação (UNICAMP), Campinas – São Paulo – Brasil
[2]Universidade Federal do Pará (UFPA), Belém – Pará – Brasil
{joahannes, wellington, allanms}@lrc.ic.unicamp.br
{cerqueira, denis}@ufpa.br, leandro@ic.unicamp.br

Abstract. *Considering the rapid modernization of vehicles, which directs them to increasingly intelligent and connected entities, the paradigm of Vehicle Edge Computing emerges. This paradigm provides cloud computing services closer to vehicular users through vehicular resources aggregation, called vehicular clouds formation. However, due to the high dynamicity of the vehicle environment, aggregating and using these resources poses some challenges. The main challenge is efficiently selecting which vehicles will take management roles in computational power, the so-called leading vehicles. Thus, this work presents the NEMESIS, a mechanism based on mobility prediction for vehicular clouds formation. NEMESIS selects vehicles with the longest dwell-time in the clouds to manage the resource utilization process. Simulation results have shown that NEMESIS can increase vehicular cloud lifetime, minimize leader changes, send fewer messages on the network, which contributes to reducing packet collision, and ultimately enables efficient use of aggregate vehicular resources.*

Resumo. *Levando em consideração a rápida modernização dos veículos, que os direciona para entidades cada vez mais inteligentes e conectadas, surge o paradigma de Computação de Borda Veicular. Esse paradigma fornece serviços de computação em nuvem mais próximos dos usuários veiculares através da agregação dos recursos de veículos, a qual é denominada formação de nuvens veiculares. No entanto, devido à alta dinamicidade do ambiente veicular, agregar e utilizar esses recursos impõe alguns desafios. O principal desafio é selecionar de forma eficiente quais veículos irão assumir papéis de gerenciamento na utilização do poder computacional, os chamados veículos líderes. Sendo assim, este trabalho apresenta o NEMESIS, um mecanismo baseado em previsão de mobilidade para formação de nuvens veiculares. O NEMESIS seleciona os veículos com maior tempo de permanência nas nuvens para gerenciar o processo de utilização dos recursos. Os resultados de simulação mostraram que o NEMESIS consegue aumentar o tempo de vida das nuvens, minimizar as mudanças de líderes, enviar menos mensagens na rede, o que colabora na diminuição da colisão de pacotes, e por fim, possibilita uma utilização eficiente dos recursos veiculares agregados.*

1. Introdução

A fim de tornar possível a disponibilização de serviços de nuvem nas Redes Veiculares Ad-hoc (do inglês, *Vehicular Ad-hoc Networks - VANETs*), surge o paradigma de Computação de Borda Veicular (do inglês, *Vehicular Edge Computing - VEC*) que tem como objetivo utilizar os recursos computacionais dos veículos como serviço para execução de tarefas que exigem certo poder de processamento na borda da rede

[Boukerche and Soto 2020]. Em resumo, a VEC é a união dos conceitos de Computação de Borda (do inglês, *Mobile Edge Computing - MEC*) e Computação em Nuvem. Dessa forma, os veículos passam a ser atuantes tanto no processo de consumo de recursos quanto no de fornecimento de poder computacional para a VANET.

Para que tais recursos sejam gerenciados e utilizados, se tem a necessidade de utilizar estratégias para o agrupamento de veículos e, consequentemente, seus recursos computacionais [Olariu 2019]. Em resumo, veículos próximos ou em uma região geográfica sob cobertura de uma infraestrutura de comunicação formam grupos que a literatura denomina de Nuvens Veiculares (do inglês, *Vehicular Clouds - VCs*) e que possibilita o processamento cooperativo de dados [Meneguette et al. 2021]. Esse processo de agregação de recursos computacionais é denominado de Formação de VCs. Em uma visão macroscópica, essas VCs podem ser intercomunicáveis por meio de infraestruturas dispostas nas cidades, permitindo a cooperação entre elas [Boukerche and Soto 2020].

No entanto, a dinamicidade topológica das VANETs se apresenta como um dos principais desafios para a proposição de abordagens eficientes no contexto veicular e, principalmente, na formação de VCs [Duarte et al. 2018, da Costa et al. 2020]. Por exemplo, pode ocorrer que a conexão entre o veículo cliente e os veículos que estão realizando o processamento na VC não dure até que as tarefas escalonadas finalizem suas execuções, o que gera processos de reescalonamento e aumento do custo pela utilização dos recursos. Dessa forma, para mitigar o impacto da mobilidade, muitos trabalhos passam a considerar informações de mobilidade veicular e, assim, utilizam algoritmos robustos para previsão do posicionamento futuro desses veículos [Long et al. 2022]. No entanto, embora a mobilidade seja levada em consideração em alguns trabalhos, eles não a envolvem em seus problemas de otimização, ou seja, o recurso de mobilidade é como uma condição de gatilho para outros processos [Wu et al. 2020]. O agrupamento de veículos é definido e tratado no restante deste trabalho como Formação de VCs.

A literatura apresenta duas formas para formação desse tipo de nuvem, a primeira com veículos em estacionamentos e a segunda com os veículos em movimento [Hagenauer et al. 2019, da Costa et al. 2020]. Dessa forma, considera-se neste trabalho os veículos em movimento e que estão em regiões geográficas específicas na cidade. Além disso, tais regiões contam com a cobertura de infraestruturas de comunicação, como Estações Base (do inglês, *Base Stations - BS*) [Abdel-Halim et al. 2019]. Diferentemente de abordagens distribuídas para formação de VCs, os veículos não possuem autonomia no processo de definição dos papéis de Veículo Líder da VC (LVC) e Veículo Membro da VC (MVC). Nesse sentido, a BS que cobre a região em que os veículos estão se movimentando que realiza os cálculos e inferências necessárias para essas definições e informa os respectivos veículos.

Este artigo apresenta um meca**N**ismo bas**E**ado em previsão de **M**obilidad**E** para aprimorar a formação de nuven**S** ve**I**culare**S**, chamado **NEMESIS**. O NEMESIS é um mecanismo para formação de VCs com auxílio de infraestrutura de comunicação e que considera a predição da mobilidade para escolha dos veículos mais estáveis na rede e, assim, aumentar o tempo de vida das VCs. Especificamente, a BS recebe informações contextuais dos veículos por meio da troca de *beacons* que naturalmente ocorre nas VANETs. Após determinados intervalos, acontece a agregação das informações na região e se tem a formação da VC coberta por essa BS. De forma geral, o número de VCs é proporcional à disposição das BSs no cenário. Considera-se também que um Controlador em um nível acima na rede que consegue ter uma visão geral dessas VCs e gerenciá-las para utilização dos recursos agregados por parte de entidades interessadas.

Em resumo, as principais contribuições deste trabalho podem ser destacadas da

seguinte forma: *(i)* introdução de um mecanismo eficiente para formação de VCs com auxílio de infraestrutura; e *(ii)* utilização de informações de predição de mobilidade para mitigar impactos da alta mobilidade veicular em ambientes urbanos. O NEMESIS foi comparado à outras abordagens para formação de VCs por meio de simulações computacionais e os resultados mostram os benefícios do NEMESIS, que oferece 37.20% maior tempo de vida das VCs, 68.87% menor número de mudanças de LVC, 65.31% menor número de mensagens trocadas na rede e 70.51% menor número de colisões de pacotes. Além disso, a eficiência e estabilidade das VCs formadas foi avaliada com um serviço importante no cenário de VEC, que é o escalonamento de tarefas em tempo real. Nessa segunda avaliação, pode-se observar que o NEMESIS emprega uma melhoria de 34.44% no escalonamento de tarefas, quando comparado com outras abordagens do estado da arte.

O restante deste artigo está organizado como se segue. A Seção 2 apresenta os trabalhos relacionados à esta pesquisa. A Seção 3 descreve o NEMESIS. A Seção 4 apresenta a metodologia de avaliação e análise dos resultados. Por fim, a Seção 5 apresenta a conclusão do trabalho.

2. Trabalhos Relacionados

Zhao *et al.* propuseram um algoritmo para formação de grupos de dispositivos móveis que considera fatores sociais, mais especificamente o coeficiente de grau [Zhao et al. 2017]. O objetivo do trabalho é maximizar a taxa de transferência da rede entre diferentes grupos de dispositivos. No processo de formação, líderes e membros de cada grupo são selecionados levando em consideração tanto atributos sociais quanto fatores físicos, como comunidade, conexões e proximidade geográfica. É uma abordagem genérica que pode ser considerada para o ambiente veicular. No entanto, os autores consideram apenas o fatores relacionados à sociabilidade dos nós, como selecionar o nó com maior grau. Ou seja, não consideram a dinâmica dessa sociabilidade em relação à mobilidade do nó na rede e o impacto que isso pode inserir na utilização dos recursos agregados.

Hagenauer *et al.* apresentaram uma abordagem baseada em mapas para formação de VCs que seleciona o veículo mais ao centro da região coberta por uma BS [Hagenauer et al. 2019]. As nuvens formadas podem fornecer serviços em suas proximidades e juntas formam VCs maiores permitindo serviços mais complexos e abrangendo cidades inteiras. O objetivo do trabalho é mostrar que a formação eficiente de VCs possibilita a utilização de recursos computacionais agregados de maneira também eficiente por parte das aplicações veiculares. Nessa abordagem, o processo de formação de VCs está limitado a intersecções presentes nos ambientes urbanos, além de que o tamanho das VCs está limitado ao raio de comunicação do LVC selecionado. Essa definição pode limitar a quantidade de recursos disponíveis para processamento das tarefas.

Pannu *et al.* propuseram uma solução para formação de VCs que considera o tempo de permanência dos veículos nas VCs [Pannu et al. 2021]. O objetivo do trabalho é garantir que os dados em processamento nas VCs sejam mantidos intactos até a conclusão desse processamento. Para isso, os tempos de permanência dos veículos em uma VC precisam ser conhecidos ou previstos com precisão. A operação do algoritmo está limitada apenas a cruzamentos presentes em ambientes urbanos. Os autores baseiam o funcionamento do algoritmo na coleta de dados de tempo de permanência de um traço de mobilidade realístico. Ou seja, o tempo de permanência dos carros é obtido a partir de uma distribuição com parâmetros já conhecidos.

Da Costa *et al.* introduziram um mecanismo que considera a formação de VCs baseada no algoritmo de agrupamento DBSCAN (do inglês, *Density Based Spatial Clustering of Application with Noise*) [da Costa et al. 2020]. Em suma, o DBSCAN identifica grupos baseando-se na densidade espacial dos indivíduos. O mecanismo obtém de forma

centralizada o posicionamento dos veículos e executa o DBSCAN para identificação das VCs. No entanto, pelo fato de não considerar aspectos temporais nessa seleção, a VC formada perde a validade no próximo instante de tempo. Além disso, a seleção do LVC nessa abordagem se dá pelo cálculo do centroide na distribuição espacial da VC e identificação do veículo que está mais próximo desse centroide. Nessa abordagem, aplicações que exigem um maior tempo de processamento podem ser prejudicas pela falta de informação temporal das VCs.

Com base na análise dos trabalhos relacionados, é possível observar que considerar informações de mobilidade veicular é fundamental para contornar os problemas impostos pela dinamicidade da VANET. No entanto, algumas abordagens não consideram a validade temporal que as informações de mobilidade dos veículos possuem. Outras já desconsideram possíveis erros de estimativa no posicionamento futuro dos veículos, o que distanciar de uma implantação em ambiente realístico.

3. Formação de Nuvens Veiculares

Esta seção apresenta o NEMESIS, que considera previsão de mobilidade para seleção dos veículos mais estáveis na rede. Tais veículos atuam como LVCs nas VCs formadas no cenário veicular e podem gerenciar a utilização de recursos computacionais.

3.1. Modelo de Sistema

A Figura 1 apresenta a arquitetura de sistema considerada. Nesse cenário, a cidade é coberta por BSs, e cada BS é responsável por cobrir uma determinada área na cidade, o que é definido como regiões (Regiões A e B, na Figura 1). Desta forma, os veículos que estão na região consequentemente estão cobertos pela BS. Oportunamente, as BS tem a capacidade de agregar e manter o conhecimento de quais e quantos veículos estão sob sua cobertura. Além disso, a BS consegue aplicar algoritmos para previsão de mobilidade dos veículos e estimar por quanto tempo eles permaneceram sob sua cobertura, podendo assim escolher o veículo mais estável para liderar sua VC.

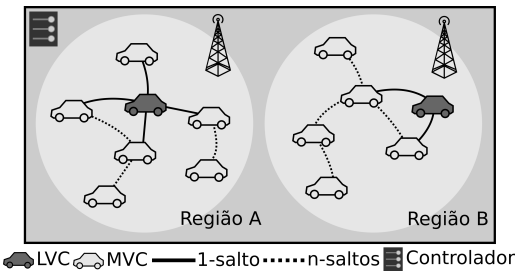


Figura 1. Arquitetura de sistema considerada.

Consideramos um cenário composto por um conjunto U de veículos, onde cada veículo u_i possui uma identificação individual ($i \in [1, x]$) e é equipado com uma Unidade de Bordo (do inglês, *On-Board Unit - OBU*) que possibilita comunicação Veículo-para-Tudo (do inglês, *Vehicle-2-Everything - V2X*). O cenário conta também com k BSs implantadas na cidade. Os veículos periodicamente enviam *beacons* na rede, de modo que o BS coleta essas informações em tempo real e constrói o conhecimento necessário para sua tomada de decisão. Informações como posição, velocidade, via atual e recursos computacionais são adicionadas ao *beacon*. Cada BS tem conhecimento das ruas que estão sob sua cobertura. Além disso, foi denotado cada VC como $v_j \in V = \{v_1, \ldots, v_m\}$, sendo $m = k$, que consiste em uma BS k com um subconjunto de veículos $N_k \subset U$ associados à ela, que são capazes de compartilhar até 2 de seus recursos computacionais ω_i, como i)

processamento e ii) armazenamento. Considerando que uma BS atende diretamente uma VC, o número de VCs é igual ao número de BSs.

Nesse cenário, os veículos podem assumir dois papéis principais. O primeiro é o veículo eleito como líder da VC (LVC) pela BS e que irá gerenciar o processo de escalonamento de tarefas. O segundo papel é o de veículo membro da VC (MVC), os quais irão compartilhar seus recursos computacionais e realizar o processamento cooperativo dentro da VC. Portanto, quando um veículo se associa à uma BS, ele automaticamente passa a fazer parte da VC coordenada por essa BS, a qual deve selecionar o veículo que gerenciará de forma microscópica esta VC. Considerando que o raio de comunicação da BS é maior que o raio de comunicação dos veículos, o LVC pode estar a n saltos de distância de alguns MVCs. Após a BS selecionar e informar o LVC, este veículo notifica os demais MVCs através de mensagens *broadcast*.

Por fim, ressalta-se que o Controlador não tem atuação direta no processo de formação de VCs. Ele é responsável apenas pela agregação das informações repassadas pelas BSs após o término do processo de formação de VCs. Com isso, aplicações de VANETs que necessitem de recursos computacionais podem solicitar tais recursos na rede e o Controlador se encarrega de decidir, baseado em uma estratégia qualquer, qual VC executará as tarefas referentes a essa aplicação [Luo et al. 2021].

3.2. Definição do Problema

As abordagens de previsão de mobilidade, em essência, estimam a posição de um determinado veículo a partir de informações atuais ou passadas do mesmo. Ou seja, utilizam como dados de entrada velocidade e posição geográfica e, assim, pode-se aplicar equações cinemáticas para prever posições futuras [Long et al. 2022, Abdel-Halim et al. 2019]. Com as informações de previsão, é possível ter conhecimento se em um instante futuro o veículo fará parte ou não de uma determinada VC. É importante tratar a mobilidade como uma série temporal, em que cada medição constitui uma entrada fornecida ao mecanismo preditor para ajustar o modelo de previsão [Long et al. 2022]. Além disso, a granularidade de previsão, no contexto espaço-temporal, pode ser definida com base nos intervalos de formação das VCs pré-definidos pelo administrador da rede.

O padrão de mobilidade em VANETs possibilita a modelagem de um sistema matemático para prever as posições geográficas futuras dos nós [Sun et al. 2018]. Sendo assim, considera-se $L_u = (X_u, Y_u)$ sendo a localização do veículo no tempo atual t e $L_{u(t+1)} = (X_{u(t+1)}, Y_{u(t+1)})$ a localização do veículo no tempo $t+1$, e assim sucessivamente até o limite de tempo considerado. Cada estrada na qual o veículo trafega possui informações de localização associadas a ela. Por exemplo, seja uma via r de largura l com início na posição geográfica $x_i, y_i = (200, 700)$ e final em $x_f, y_f = (500, 700)$, é possível definir se uma posição p está contida nessa via ou não. Dados de localização geográfica são obtidos por meio de mapas digitais e do Sistema Global de Navegação por Satélite.

Portanto, quando os veículos se associam à uma BS e fornecem suas informações, a BS pode agregá-las e criar a infraestrutura de nuvem para processamento cooperativo de dados, ou seja, a VC. No entanto, para garantir o atendimento das solicitações em tempo real, é necessário levar o gerenciamento desses recursos para o interior da VC. Para isso, a BS seleciona o veículo mais estável (com base em uma métrica pré-definida) para operar como o LVC. No entanto, essa definição LVC é um dos pontos críticos no processo de formação de VCs. Esse LVC receberá as regras do controlador e gerenciará a atribuição de tarefas entre os MVCs. A eficiência geral do sistema pode ser degradada dependendo dos critérios de seleção utilizados. Resumidamente, o alto número de mudanças de LVC leva a um aumento do número de mensagens de aviso na rede e, consequentemente, a um alto número de colisões de pacotes. Além de degradar a eficiência do sistema, a rede

ficará congestionada com transmissões desnecessárias.

3.3. NEMESIS

Com base na posição predita do veículo, o NEMESIS verifica de qual rua essa posição geográfica faz parte. Ou seja, como a BS sabe quais ruas estão sob sua cobertura, se a previsão apontar para uma rua que não consta na sua base de dados, a BS sabe que esse veículo não estará mais em sua cobertura e, consequentemente, não fará mais parte de sua VC. É importante ressaltar que o NEMESIS pode funcionar com qualquer modelo para previsão de mobilidade existente na literatura.

A BS mantém uma estrutura de vizinhança para guardar as informações dos veículos em sua cobertura. A cada recebimento de *beacon*, a BS verifica se este veículo já consta nessa estrutura. Caso negativo, o veículo é adicionado, junto das informações presentes no *beacon*. Ao adicionar um veículo em sua lista de vizinhos, é iniciado um temporizador exclusivo para esse nó. A função do temporizador é guardar o instante em que o veículo foi adicionado na lista e se esse valor não for renovado na próxima rodada de recebimento de *beacons*, significa que o veículo com temporizador zerado não faz mais parte da vizinhança da BS correspondente.

A Figura 2 apresenta uma visão detalhada do NEMESIS. O objetivo é saber até que instante um determinado veículo fará parte da VC. No exemplo da figura, o veículo permanece 12 segundos na cobertura da BS da Região A. Como alguns métodos para previsão exigem uma base de informações passadas para realizarem as estimativas futuras, e como tais informações chegam na BS com frequência de 1 Hz através da troca de *beacons*. A BS guarda as informações de mobilidade de todos os veículos em sua cobertura para então construir uma base para o método de previsão. Ou seja, se a previsão for realizada no instante $t = 4$ com uma janela de observação de 10 segundos, as informações para construção da base de dados serão do intervalo de $t = 0$ até $t = 4$ e, assim, será identificado que esse veículo não fará mais parte da VC da Região A no instante $t = 14$.

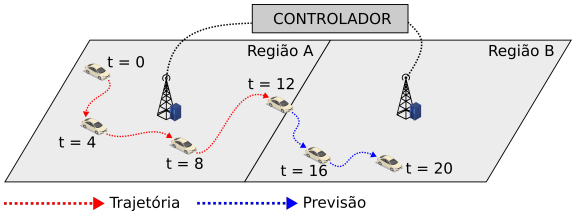


Figura 2. Visão microscópica do NEMESIS.

Dessa forma, conforme mostrado no Algoritmo 1, o conjunto de veículos N_k (onde k representa a identificação da BS), as ruas cobertas R_k e a janela de observação t para a previsão são fornecidos ao NEMESIS. Primeiramente, o algoritmo verifica se o conjunto N_k não é vazio, o que significa que existem veículos na cobertura da BS. Caso N_k esteja vazio, os recursos dessa VC são indicados como 0 e o Controlador é informado (Linha 20). Caso contrário, as posições desses veículos são previstas considerando a janela de observação t (Linha 6) e as posições estimadas são adicionadas a um conjunto temporário N'_k que contém a identificação do veículo e a lista de posições estimadas (Linha 7). Sendo assim, cada um dos veículos cobertos é verificado e seus recursos compartilhados são agregados para a VC (Linha 9). Além disso, caso as posições estimadas constarem no conjunto de ruas cobertas R_k, é contabilizado um intervalo de tempo de permanência na VC para cada posição (Linha 13 e 14). Ao final da verificação das posições estimadas, o tempo de permanência na VC é adicionado ao conjunto N_k inicial (Linha 15). Como o objetivo é selecionar o veículo que passará mais tempo na cobertura da BS, basta buscar

o tempo máximo conjunto N_k (Linha 16). Com o veículo identificado, a BS envia uma mensagem de aviso com a identificação do LVC (Linha 17). Por fim, a BS informa o Controlador acerca da VC criada e seus recursos disponíveis (Linha 18).

Algoritmo 1: Abstração do NEMESIS na BS

```
   Entrada: conjunto de veículos N_k, ruas cobertas R_k e janela de tempo t
1  início
2      meuId ← k
3      recursos ← 0
4      LVC ← NULL
5      se N_k ≠ ∅ então
6          posicoesPrevistas ← ARIMA(N_k, t)
           ▷ N'_k.posicoes É UMA LISTA COM TAMANHO t
7          N'_k.posicoes ← posicoesPrevistas
8          para cada i ∈ N_k faça
               ▷ AGREGA OS RECURSOS DOS VEICULOS
9              recursos ← recursos + i.recurso
10             para cada i' ∈ N'_k faça
11                 tempoNaVC ← 0
12                 para cada p ∈ i'.posicoes faça
                       ▷ SE p É UMA POSIÇÃO DE ALGUMA RUA EM R_k
13                     se p ∈ R_k então
14                         tempoNaVC ← tempoNaVC + 1
15             i.tempo ← tempoNaVC
16         LVC ← max(N_k.tempo)
17         enviaMensagemLVC(meuId, LVC)
18         enviaMensagemControlador(meuId, LVC, recursos)
19     senão
20         enviaMensagemControlador(meuId, LVC, recursos)
```

Por outro lado, o Algoritmo 2 apresenta uma abstração de quando o veículo recebe a mensagem de aviso da BS. No NEMESIS, os veículos podem assumir três estados, sendo (i) *ND*, que significa estado não definido; (ii) *MVC*, que representa os veículos membros da VC; e (iii) *LVC*, que representa o líder da VC. Antes de qualquer processo de formação, todos os veículos têm estado definido como ND. Sendo assim, ao receber uma mensagem de aviso da BS, o veículo verifica seu estado atual e se for ND significa que o veículo ainda não faz parte de nenhuma VC (Linha 4). No entanto, ele guarda a informação de qual BS mandou essa mensagem (Linha 5). Então, o veículo verifica se seu número de identificação consta nessa mensagem, se sim, seu estado muda para LVC (Linha 7), e inicia a liderança nessa VC (Linha 8). O LVC deve informar seus vizinhos sobre seu novo papel na VC, para isso cria uma mensagem de líder, adiciona sua identificação, a identificação da BS que lhe dá suporte, o número de saltos até o LVC (que nesse caso é 0 pois ele mesmo é o LVC) e envia a mensagem na rede (Linhas 9 a 13). Se ao receber uma mensagem de aviso da BS o estado do veículo não for ND, ele verifica se já atua no papel de LVC. Caso positivo, verifica se após o novo processo de formação de VC ele ainda consta como LVC e se não for, o veículo deixa de liderar a VC nesse momento (Linha 17).

Após o LVC informar seus vizinhos sobre seu papel, algumas ações devem ser tomadas. O veículo receptor tem duas possibilidades ao receber mensagem de outro veículo, a primeira é esse vizinho ser um LVC e a segunda é ele ser um MVC. No entanto, as ações tomadas para ambos os casos são as mesmas. Primeiramente o veículo verifica se seu estado é ND ou MVC. Se sim, verifica se sua BS é a mesma que consta na mensagem recebida, isso é um controle para que o veículo descarte mensagens que não são de seu interesse, ou seja, de outras VCs. Caso a mensagem seja da mesma BS, o veículo muda seu status para MVC, computa seu LVC e cria uma mensagem para retransmitir a informação de liderança na VC, com a adição de informações como a BS que os cobre, o LVC e

Algoritmo 2: Recebe mensagem de aviso da BS

```
   Entrada: msg
1  início
       ▷ RECEBE MENSAGEM DA BS
2      estadoAtual ← ND
3      se mensagem de aviso da BS então
4          se estadoAtual == ND então
5              minhaBS ← msg.bsId
6              se meuId == msg.LVC então
7                  estadoAtual ← LVC
                   ▷ INICIA SUA LIDERANÇA A PARTIR DO TEMPO ATUAL
8                  INICIAVC(TEMPOATUAL())
                   ▷ CRIA MENSAGEM DE LÍDER
9                  msg.LVC ← meuId
10                 msg.BS ← minhaBS
11                 msg.saltos ← 0
12                 msg ← add(LVC)
                   ▷ ENVIA MENSAGEM INFORMANDO VIZINHOS
13                 ENVIAMENSAGEMDELVC(msg)
14         senão
15             se estadoAtual == LVC então
16                 se meuId ≠ msg.LVC então
17                     DESTROIVC(TEMPOATUAL())
```

o número de saltos até o LVC (acrescentando o salto da retransmissão que fará). Isso é importante para que o MVC tenha conhecimento de quantos saltos o separa do LVC. Como nessa etapa o ambiente é distribuído, o veículo pode receber a mesma mensagem diversas vezes, então é necessário verificar se nesse intervalo o veículo já retransmitiu essa mensagem. E, caso negativo, o veículo retransmite a mensagem na rede.

Em relação a previsão de mobilidade, neste trabalho foi considerado o modelo de previsão bastante difundido na literatura, chamado ARIMA. Vale a pena realçar que o NEMESIS funciona com qualquer algoritmo de predição de mobilidade, e nesse trabalho foi considerado o ARIMA, por ser de simples implantação e possuir complexidade linear [Sun et al. 2018]. O ARIMA é um modelo estatístico para analisar e prever séries temporais e funciona tomando valores de série e tornando-os estacionários, se necessário. Uma série temporal estacionária não tem tendência e a amplitude de suas variações em torno da média é constante. No modelo ARIMA, os valores futuros das séries são considerados uma combinação linear de valores passados e médias móveis passadas [Sun et al. 2018].

O ARIMA é descrito como uma tupla (p, d, q), onde p corresponde ao número de medições passadas, d consiste no número de séries de diferenciação para tornar estatisticamente estacionário, e q corresponde ao número de médias móveis anteriores. A formulação básica do modelo é dada pela Equação 1. Denotamos os termos anteriores como k, as médias móveis anteriores como μ, enquanto θ e Φ são pesos individuais para cada termo e serão treinados pelo modelo.

$$k_t = \theta_0 + \Phi_1 k_{t-1} + \Phi_2 k_{t-2} + \cdots + \Phi_p k_{t-p} - \theta_1 \mu_{t-1} - \theta_2 \mu_{t-2} - \cdots - \theta_p \mu_{t-p} \quad (1)$$

Em resumo, o número de termos de valores anteriores e médias móveis anteriores depende da série considerada, onde algumas séries são principalmente dependentes de valores anteriores ponderados e não precisam de quaisquer termos de média móvel. O modelo pode ser representado pela notação ARIMA(4, 1, 0), o que significa que utiliza-se quatro termos anteriores, realiza-se uma diferenciação e não considera-se médias móveis anteriores, por exemplo. O ARIMA é utilizado para previsão de séries temporais de variável única, sendo necessária uma etapa de treinamento da latitude e longitude dos

veículos de forma separada. Sendo assim, o modelo é treinado para cada veículo e suas respectivas coordenadas geográficas.

4. Avaliação

As simulações foram realizadas no *framework* Veins[1] do OMNeT++, versões 4.7.1 e 5.6, respectivamente. O Veins implementa a pilha de protocolo do padrão IEEE 802.11p para comunicação V2X e atenuação de sinal. Para estabelecer um cenário de avaliação, utilizou-se uma área de 2 km $\times$ 2 km da cidade de Manhattan nos Estados Unidos. Essa região foi obtida através do OpenStreetMap[2] e importada pelo *Simulation of Urban MObility* (SUMO)[3] para gerar os registros de movimentação dos veículos.

Os efeitos de atenuação de sinais causados por edifícios foram considerados, onde assumiu-se que cada bloco possui um obstáculo de 300m x 300m para representação de quarteirões com edifícios altos. Considerou-se 4 BSs implantadas no cenário e cada BS é capaz de cobrir até 4 quarteirões, possuindo raio de comunicação de 700m. Para quantificar a eficiência da solução, fixou-se a densidade veicular em 100 veículos/km^2. A velocidade dos veículos respeita os limites impostos pelo cenário, onde variou-se em $18, 36, 54, 72$ e 90 km/h. Esse tipo de avaliação é importante pois mostra o quanto a mobilidade veicular influencia na eficiência das soluções.

A taxa de bits na camada MAC foi definida em 6 Mbit/s e a potência de transmissão para 2.2mW. Esses parâmetros, junto do modelo de propagação *Two-Ray ground*, fornecem um raio de comunicação de 300m para os veículos. A frequência de envio de *beacons* foi de 1 Hz. Cada simulação foi executada 33 vezes com diferentes sementes de aleatoriedade e os resultados apresentam os valores com um intervalo de confiança de 95%. O tempo de simulação foi de 200 segundos. Foram consideradas duas soluções de formação de VCs para comparação, sendo a primeira baseada em coeficiente de grau, denominada GRAU [Zhao et al. 2017], e a segunda que seleciona o LVC que estiver mais próximo ao centróide da VC, denominada CENTROIDE [Hagenauer et al. 2019].

Para a previsão de mobilidade, foi considerado o ARIMA(1,1,0) nesse conjunto de dados, ou seja, foi considerado um valor passado, a série é direcionada uma vez para torná-la estacionária e nenhum termo de média móvel. Esses parâmetros foram encontrados usando um estimador *Grid Search*. A cada intervalo de formação de VCs, que foi definido em 5 segundos, os dados passados de mobilidade são guardados para servirem de entrada para previsões futuras. O modelo de previsão foi implementado na linguagem Python 3.5 com a biblioteca *statsmodels* versão 0.14.0 e conectado ao Veins via interface de sistema operacional OS.SYSTEM() da linguagem C++.

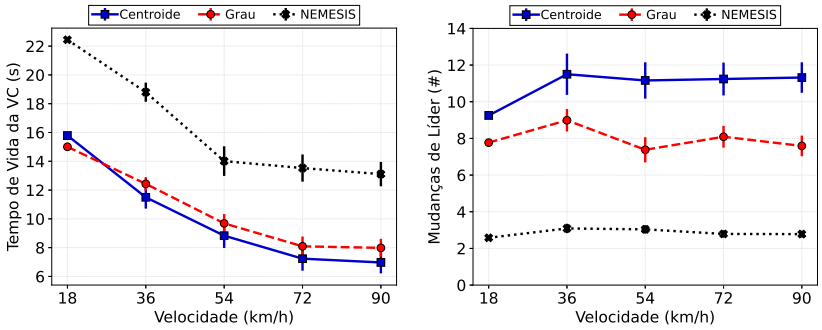
Para uma melhor apresentação e discussão dos resultados, duas Subseções foram consideradas. Na primeira, quatro métricas foram consideradas, sendo elas: *i) Tempo de vida da VC* representa a contabilização de quantas unidades de tempo as VCs permaneceram sem alterações em seu gerenciamento; *ii) Mudanças de Líder* representa o número de vezes que o veículo LVC mudou na VC; *iii) Mensagens transmitidas* diz respeito ao número de mensagens que são enviadas na rede, onde neste caso é considerado apenas as de comunicação entre veículos; e *iv) Colisão de pacotes* representa a soma de pacotes perdidos ao receber/transmitir (RxTx) e por interferência de sinal. Já na segunda avaliação, considerou-se a métrica *i) Sucesso de escalonamento* que contabiliza a porcentagem de tarefas escalonadas e executadas com sucesso nas VCs.

[1] `http://veins.car2x.org/`
[2] `http://www.openstreetmap.org/`
[3] `http://sumo.dlr.de`

4.1. Avaliação de Formação de VCs

A Figura 3(a) mostra os resultados obtidos em relação ao tempo de duração médio das VCs criadas. Pode-se notar que as VCs formadas com o NEMESIS possuem maiores tempos de vida em todos os cenários observados. Ou seja, o fato de selecionar o veículo que passará mais tempo na cobertura da BS, faz com que a VC exista durante esse tempo de viagem do veículo. O que difere das outras abordagens comparadas. No entanto, nota-se que a medida que a velocidade média dos veículos aumenta, o tempo de vida das VCs diminui. Essa diminuição no desempenho é natural e esperada, já que os nós ficarão menos tempo em contato uns com os outros e completarão suas viagens mais rapidamente. O NEMESIS empregou melhoria geral de 34.61% e 39.79% em relação ao GRAU e CENTROIDE, respectivamente.

A Figura 3(b) mostra a estabilidade geral das VCs criadas. Pode-se observar que o NEMESIS possui menos mudanças de líder do que as outras abordagens. O fato do algoritmo selecionar o veículo que passará mais tempo na cobertura da BS para liderar a VC, implica uma maior estabilidade para o gerenciamento da VC. O CENTROIDE tem o pior desempenho, pois seleciona o veículo que está sempre mais próximo do centroide da região. E espera-se que dada a mobilidade veicular, esses veículos mais próximos ao centroide mudem com uma certa frequência. O mesmo acontece com a abordagem baseada em GRAU, que seleciona o veículo com maior vizinhança. Ou seja, dada a sua movimentação, a densidade de vizinhança desse veículo também muda com certa frequência. Dessa forma, o NEMESIS emprega melhoria nessa métrica em 64.02% e 73.72% em relação ao GRAU e CENTROIDE, respectivamente.



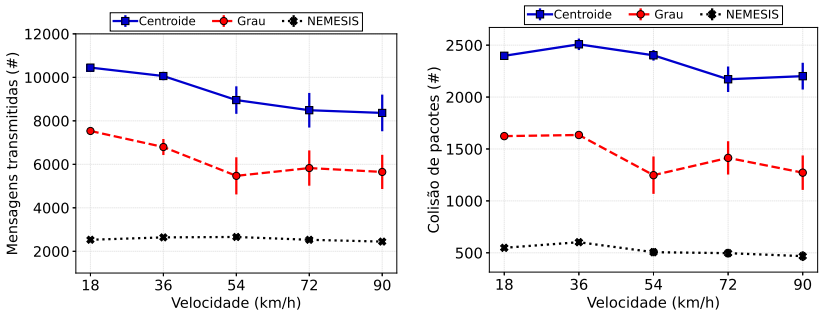
(a) Tempo de vida da VC (b) Mudanças de líder

Figura 3. Resultado da comparação das técnicas de previsão

A Figura 4(a) apresenta resultados referentes ao número de mensagens transmitidas na rede. É considerado nessa avaliação apenas as mensagens transmitidas entre os veículos, pois o número de mensagens transmitidas pela BS é o mesmo em ambas as abordagens, pois os intervalos de formação de VCs são iguais. No entanto, conforme visto nos Algoritmos 1 e 2, o número de mensagens transmitidas está relacionado com o número de mudanças de LVC. Dessa forma, pode-se observar que o NEMESIS transmite um número inferior de mensagens na rede em comparação com as abordagens consideradas. Esse resultado pode ser relacionado com o resultado mostrado na Figura 3(b), pois quanto menor o número de mudanças de LVCs, menos mensagens de aviso são enviadas na rede. Em resumo, o NEMESIS emprega melhoria de 58.46% e 72.16% em relação ao GRAU e CENTROIDE, respectivamente.

A Figura 4(b) mostra a relação entre mensagens transmitidas e colisão de pacotes na rede. Ressalta-se que todas as abordagens utilizam um algoritmo de *Flooding* para disseminação das mensagens de aviso na rede. Optou-se por utilizar essa abordagem

por ser mais simples e, em termos gerais, alcançar bons resultados em relação à taxa de entrega. No entanto, podemos observar que o simples fato de manter a seleção dos líderes estável já implica melhorias significativas nas métricas de rede. O NEMESIS obteve o menor número de colisão de pacotes, pois transmite menos mensagens de aviso por conta do baixo número de mudanças de LVC. Sendo assim, o NEMESIS emprega melhoria de 63.42% e 77.61% em relação ao GRAU e CENTROIDE, respectivamente.



(a) Mensagens transmitidas (b) Colisão de pacotes

Figura 4. Resultado da comparação das técnicas de previsão

4.2. Avaliação de Escalonamento de Tarefas

Para avaliar o impacto da estabilidade das VCs criadas, se fez necessário simular uma aplicação prática do cenário VEC. Tal aplicação diz respeito à utilização dos recursos computacionais veiculares que foram agregados no processo de Formação de VCs. Essa utilização é denominada escalonamento de tarefas e deve considerar as restrições de tempo de processamento de cada uma das tarefas. Como não se tem restrição de ordem para execução das tarefas que chegam ao sistema, considerou-se abstrações de aplicações *Bag-of-Tasks* para esta avaliação.

Dessa forma, modelou-se a avaliação de escalonamento da seguinte maneira: existe um conjunto de tarefas onde cada tarefa conta com um tempo de processamento necessário para sua conclusão. Quando uma tarefa é escalonada, seu sucesso de escalonamento só é computado quando o tempo total de processamento é atingido. Caso o LVC mude, a tarefa é automaticamente cancelada e sua interrupção é computada. O Controlador presente na rede se encarrega de selecionar quais VCs irão processar um determinado conjunto de tarefas. Por outro lado, os LVCs se encarregam de gerenciar o processamento dessas tarefas entre os seus MVCs.

O Algoritmo 3 apresenta o processo de escalonamento. O Controlador recebe o conjunto de VCs V e o conjunto de tarefas T. Para cada tarefa é verificado se o peso dela é suportado na VC atual, e caso positivo, a tarefa é escalonada nessa VC (Linha 4). O Controlador informa o LVC, através da BS correspondente, que a tarefa foi escalonada em sua VC e decrementa a quantidade de recurso computacional referente à tarefa (Linha 5). Ou seja, mesmo que o principal fator seja o tempo de processamento da tarefa, ainda assim é necessário ter recurso de armazenamento para executar essa tarefa durante o tempo requerido por ela. Ao receber uma mensagem de resposta do LVC, o Controlador verifica se a tarefa foi executada com sucesso (Linha 6). Se sim, a capacidade da VC é acrescida do peso da tarefa concluída (Linha 7). Se ainda existirem tarefas pendentes, o escalonador reexecuta os passos iniciais (Linha 8). Caso a tarefa não tenha sido concluída com sucesso, o processo inicial é executado novamente para tentar reescaloná-la em uma das novas VCs que serão formadas (Linha 9).

Algoritmo 3: Mecanismo escalonador de tarefas do Controlador

```
   Entrada: conjunto de nuvens veiculares V e conjunto de tarefas T
1  início
       ▷ CONTROLADOR ESCALONA A TAREFA NA VC
2      para cada vc ∈ V faça
3          para cada tarefa ∈ T faça
4              se tarefa.peso ≤ vc.capacidade então
                   ▷ tarefa É ESCALONADA EM vc
                   ▷ INFORMA LVC DE vc PELA SUA BS CORRESPONDENTE
5                  vc.capacidade ← vc.capacidade − tarefa.peso
       ▷ CONTROLADOR RECEBE RESPOSTA DO LVC
6      se tarefa executada com sucesso então
7          vc.capacidade ← vc.capacidade + tarefa.peso
8          se T ≠ ∅ então
               ▷ REPETE PROCESSO ENTRE LINHA 2 E 5
9      senão
           ▷ REPETE PROCESSO ENTRE LINHA 2 E 5
```

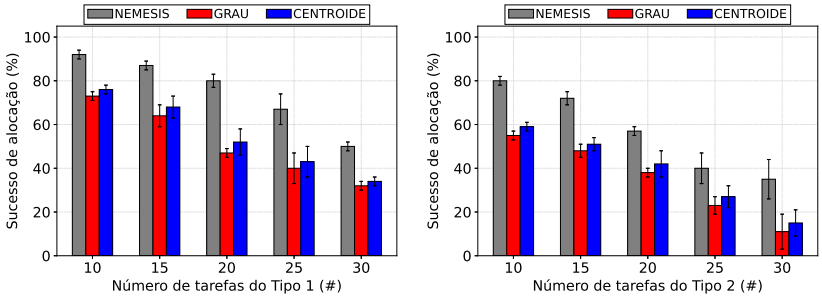
Complementarmente, o Algoritmo 4 apresenta uma abstração do que ocorre no veículo LVC ao receber a mensagem do Controlador. Ao receber a mensagem de escalonamento, o LVC inicia um temporizador para controlar quando a tarefa concluirá sua execução (Linha 2), sendo este incrementado a cada unidade de tempo passada (Linha 4). Como o processo de formação da VC é independente do processo de escalonamento, se faz necessário a cada instante de tempo verificar se o veículo atual ainda é o LVC. Se sim, ele verifica se o temporizador de controle é igual ao tempo de processamento da tarefa e, caso positivo, a tarefa foi executada completamente (Linha 6). O LVC informa ao controlador sobre a execução da tarefa e computa o sucesso de escalonamento. No entanto, caso aconteça do veículo deixar de ser LVC durante o processo de execução da tarefa, é verificado se o seu temporizador de controle é menor que o tempo de processamento da tarefa e, caso positivo, a tarefa foi interrompida (Linha 8). O veículo informa o Controlador sobre essa interrupção, zerando o seu temporizador de controle (Linha 9).

Algoritmo 4: Abstração da execução da tarefa da VC

```
   Entrada: msg
1  início
       ▷ LVC RECEBE MENSAGEM DE ALOCAÇÃO
       ▷ LVC INICIA UM TIMER PARA CONTROLE DA EXECUCAO
2      tempoEscalonamento ← 0
3      para cada instante de tempo faça
4          tempoEscalonamento ← tempoEscalonamento + 1
5          se estadoAtual == LVC então
6              se tempoEscalonamento == msg.tempo então
                   ▷ EXECUCAO COMPLETA DA TAREFA
                   ▷ INFORMA CONTROLADOR PELA BS
                   ▷ CONTABILIZA SUCESSO
7              senão
8                  se tempoEscalonamento ≠ msg.tempo então
                       ▷ TAREFA INTERROMPIDA
                       ▷ INFORMA CONTROLADOR PELA BS
                       ▷ CONTABILIZA FRACASSO
9                  tempoEscalonamento ← 0
```

As configurações de simulação foram as mesmas utilizadas na avaliação anterior. No entanto, para avaliar a eficiência das VCs, considerou-se dois tipos de aplicações, o primeiro (Tipo 1) com tempo de processamento que varia de 1 a 3 segundos. E o segundo tipo (Tipo 2) conta com tempo de processamento que varia de 5 a 10 segundos. Como o

objetivo é avaliar a eficiência das VCs em relação à estabilidade, outras métricas além do tempo das tarefas e VCs não foram consideradas. Variou-se o número de tarefas de cada tipo em $10, 15, 20, 25$ e 30. A velocidade veicular escolhida foi a de 18 km/h, pelo fato das abordagens terem os melhores desempenhos nesse cenário, conforme visto na Subseção 4.1. O tempo de simulação também foi de 200 segundos.



(a) Tarefas Tipo 1 (b) Tarefas Tipo 2

Figura 5. Resultados da avaliação considerando o tempo de vida das VCs

A Figura 5(a) apresenta os resultados para as tarefas do Tipo 1. Pode-se observar que os resultados obtidos corroboram com os resultados obtidos na avaliação de formação de VCs, em relação ao tempo de vida médio das nuvens. Ou seja, quanto maior é o tempo de vida da VC, maior é a chance dela atender uma maior quantidade de tarefas no sistema. E não apenas isso, mas a estabilidade da VC é crucial para aumentar a porcentagem de tarefas que são executadas com sucesso. Conforme esperado, à medida que o número de tarefas cresce no sistema, a chance de que tarefas não consigam ser escalonadas também aumenta. No entanto, em todos os cenários observados, o NEMESIS consegue se manter superior e escalonar 28.59% e 33.10% mais tarefas do que as abordagens CENTROIDE e GRAU, respectivamente.

A Figura 5(b) segue o mesmo comportamento da Figura 5(a). No entanto, o desempenho de ambas abordagens é degradado pelo tempo de processamento superior das tarefas do Tipo 2. Como a média de tempo de processamento das tarefas é quase 3 vezes mais do que as tarefas do Tipo 1, o tempo de vida médio das VCs impacta em aplicações desse tipo. Mas é importante observar que o NEMESIS ainda mantém um desempenho próximo de 80% de tarefas alocadas nos cenários com menor número de tarefas no sistema. De maneira geral, o NEMESIS se mostrou 34.28% e 41.80% mais eficiente do que as abordagens CENTROIDE e GRAU, respectivamente.

5. Conclusão

Este artigo apresentou o NEMESIS, um mecanismo baseado em previsão de mobilidade para formação eficiente de VCs. O NEMESIS considera um método de previsão de séries temporais para estimar o tempo de permanência dos veículos nas VCs e selecionar os veículos mais estáveis para gerenciar a utilização dos recursos dessas nuvens. Com base nesse mecanismo, aplicações de VANETs podem ser escalonadas nas VCs e terem garantias de que seus processos serão concluídos com sucesso durante o tempo necessário. Os resultados mostram que o NEMESIS consegue aumentar o tempo de vida médio das VCs, diminuir a troca de líderes, e enviar menos mensagens na rede, causando menor número de colisões de pacotes, em relação à outras abordagens da literatura. Como trabalhos futuros, pretende-se explorar outras técnicas para previsão de mobilidade e considerar outras métricas para avaliação do escalonamento de tarefas.

Agradecimentos

Os autores agradecem o apoio concedido pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e pela Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) por meio dos processos de Nº #2018/16703-4 e #2015/24494-8.

Referências

Abdel-Halim, I. T., Fahmy, H. M. A., and Bahaa-El Din, A. M. (2019). Mobility prediction-based efficient clustering scheme for connected and automated vehicles in vanets. *Computer Networks*, 150:217–233.

Boukerche, A. and Soto, V. (2020). Computation offloading and retrieval for vehicular edge computing: Algorithms, models, and classification. *ACM Computing Surveys (CSUR)*, 53(4):1–35.

da Costa, J. B. D., Meneguette, R. I., Rosário, D., and Villas, L. A. (2020). Combinatorial optimization-based task allocation mechanism for vehicular clouds. In *Proceedings of the IEEE 91st Vehicular Technology Conference (VTC Spring)*, pages 1–5. IEEE.

Duarte, J. M., Kalogeiton, E., Soua, R., Manzo, G., Palattella, M. R., Maio, A. D., Braun, T., Engel, T., Villas, L. A., and Rizzo, G. A. (2018). A multi-pronged approach to adaptive and context aware content dissemination in vanets. *Mobile Networks and Applications*, 23(5):1247–1259.

Hagenauer, F., Higuchi, T., Altintas, O., and Dressler, F. (2019). Efficient data handling in vehicular micro clouds. *Ad Hoc Networks*, 91:101871.

Long, W., Li, T., Xiao, Z., Wang, D., Zhang, R., Regan, A., Chen, H., and Zhu, Y. (2022). Location prediction for individual vehicles via exploiting travel regularity and preference. *IEEE Transactions on Vehicular Technology*.

Luo, Q., Li, C., Luan, T., and Shi, W. (2021). Minimizing the delay and cost of computation offloading for vehicular edge computing. *IEEE Transactions on Services Computing*, 1374:1–12.

Meneguette, R., De Grande, R., Ueyama, J., Filho, G. P. R., and Madeira, E. (2021). Vehicular edge computing: Architecture, resource management, security, and challenges. *ACM Computing Surveys (CSUR)*, 55(1):1–46.

Olariu, S. (2019). A survey of vehicular cloud research: Trends, applications and challenges. *IEEE Transactions on Intelligent Transportation Systems*, 21(6):2648–2663.

Pannu, G. S., Ucar, S., Higuchi, T., Altintas, O., and Dressler, F. (2021). Dwell time estimation at intersections for improved vehicular micro cloud operations. *Ad Hoc Networks*, 122:102606.

Sun, G., Song, L., Yu, H., Chang, V., Du, X., and Guizani, M. (2018). V2v routing in a vanet based on the autoregressive integrated moving average model. *IEEE Transactions on Vehicular Technology*, 68(1):908–922.

Wu, X., Zhao, S., Zhang, R., and Yang, L. (2020). Mobility prediction-based joint task assignment and resource allocation in vehicular fog computing. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6. IEEE.

Zhao, P., Feng, L., Yu, P., Li, W., and Qiu, X. (2017). A social-aware resource allocation for 5g device-to-device multicast communication. *IEEE Access*, 5:15717–15730.