An Architecture of Fog Computing in Smart Cities: the middleware E2BS in emergency calls

Sediane Carmem Lunardi Hernandes
Universidade Tecnológica Federal do Parana (UTFPR)
Guarapuava – PR, Brasil
sedianec@utfpr.edu.br

Marcelo Eduardo Pellenz
Programa de Pós-Graduação em Informática (PPGIa)
Pontifícia Universidade Católica do Paraná
Curitiba – PR, Brasil
marcelo@ppgia.pucpr.br

Alcides Calsavara
Programa de Pós-Graduação em Informática (PPGIa)
Pontifícia Universidade Católica do Paraná
Curitiba – PR, Brasil
alcides@ppgia.pucpr.br

***Abstract*— The use of smart objects in an urban context will be offers services that will help the cities. Thereby, it is possible to implement the Smart Cities concept. Among the services that could be offered is the emergency service calls (e.g., police and firefighters). In an emergency situation, the most appropriate mobile units may be required. Therefore, a middleware can be used to support the activation of the most suitable mobile units. The bigger question is which the vantage of choice the most suitable mobile unit compared of the choice randomic. In this way this paper presents a new event model and a middleware called Event to Best Subscribers (E2BS) aim to answer this question through two use cases.**

Keywords*— middleware publish-subscribe, fog computing, event routing, smart cities.*

I. INTRODUÇÃO

A população mundial vem aumentando significantemente nas últimas décadas e para 2050 é estimado que a população urbana alcance os 70% [1]. Para 2030 prevê-se que mais de 90% da população brasileira viva nas cidades e na América Latina estima-se que, em 2050, a população seja 86% urbana [2]. Essa rápida urbanização e o aumento populacional terão vários impactos negativos, como o lixo que tende a aumentar, a escassez de recursos como, por exemplo, água e luz, a poluição do ar e sonora, além de preocupações com a saúde, o tráfego urbano e a educação, a mobilidade, entre tantos outros.

Assim, Cidades Inteligentes originaram-se para resolver esses problemas nas cidades modernas e espera-se enfrentar estes desafios com o uso das TICs (*Information Communication Technologies* - ICT) [3]. Por isso, [4] afirma que as TICs são uma parte essencial do desenvolvimento urbano e são necessárias para todas as Cidades Inteligentes. A Internet das Coisas (*Internet of Things - IoT*) aplicada em um ambiente urbano vem ao encontro a estas necessidades e torna possível que vários serviços inteligentes sejam oferecidos aos habitantes, como, por exemplo, cuidados médicos inteligentes (*Smart Health care)*, energia inteligente (*Smart Energy*), governança inteligente (*Smart Governance*), serviços de emergência inteligentes (*Smart emergency´s services*), entre outros.

IoT é uma rede de objetos físicos interconectados (objeto inteligentes) incluindo computadores, smartphones, sensores, atuadores, casas, edifícios, estruturas, veículos e sistemas de energia [5]. Em IoT, objetos inteligentes se comunicam entre si, tomam decisões com base em dados recebidos, reagem a eventos, entre outros. Em serviços de emergência inteligentes um evento corresponde a ocorrência de uma situação de emergência (e. g., acidente, assassinato, roubo, terrorismo, incêndio, sequestro). Como administrar esses eventos é uma questão a ser gerenciada. Além disso, a questão do atraso deve ser considerada, uma vez que para aplicações de saúde e de emergência atrasos não são tolerados. Pensando nisso, vários trabalhos relacionados a *Fog Computing* (Computação na borda da rede) foram analisados para propor uma arquitetura que ajudasse a minimizar o atraso no envio de eventos, bem como limitar o envio de mensagens para um serviço de *Cloud*. Além disso, vários middlewares *publish-subscribe* foram analisados objetivando seu uso em Cidades Inteligentes, especialmente serviços em que exigem o envio somente a um subconjunto dos assinantes, como serviços de cuidados médicos e de situações de emergência. O que foi observado é que o envio de eventos somente aos assinantes mais adequados, como é necessário para algumas aplicações de Cidades Inteligentes, não se faz presente nos trabalhos analisados. Assim, a seguinte hipótese de pesquisa foi delineada para a condução do trabalho: Se o envio de eventos a um subconjunto de assinantes é melhor do que uma escolha aleatória então as situações de emergência serão atendidas em um espaço de tempo menor. Como consequência, os prejuízos humanos e materiais poderiam ser minimizados. Desta forma, este trabalho apresenta uma arquitetura de *Fog Computing*, um modelo de eventos para o envio de eventos a um subconjunto de assinantes mais adequados (protocolo *Event to Best Subscribers* – E2BS), sua implementação por meio do Middleware E2BS e o uso do middleware em dois cenários diferentes. De uma forma geral, procurar-se-á mostrar que o envio dos eventos somente aos assinantes mais apropriados é melhor do que uma escolha aleatória de assinantes.

O artigo é dividido com segue. Na seção II são descritos conceitos de Cidades Inteligentes e sua relação com IoT. *Fog Computing* é apresentado na seção III. Middlewares *publish-subscribe* são avaliados na seção IV. A Seção V especifica o modelo de eventos e o middleware que o implementa, bem como descreve os dois casos de uso (isto é, bombeiros e viaturas policiais) em que o middleware foi validado com seus

XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE

respectivos resultados computacionais. Algumas considerações finais são realizadas na seção VI.

II. CIDADES INTELIGENTES

Apesar de existir um aumento na frequência do uso de Cidade Inteligente, uma definição concreta de uma Cidade Inteligente ainda está emergindo e várias definições têm sido dadas com diferentes pontos de vista [6]; entretanto, convergindo para um ponto em comum, o aumento da qualidade de vida dos cidadãos.

Em termos genéricos Cidade Inteligente é um ambiente urbano que utiliza TCIs e outras tecnologias relacionadas para aumentar a eficiência e a qualidade dos serviços das operações regulares da cidade oferecida aos cidadãos urbanos [7]. Uma definição mais formal pode ser encontrada em [8], a qual apresenta uma Cidade Sustentável Inteligente (*Smart Sustainable City - SSC*) como uma cidade que usa as TCIs e outros meios para melhorar a qualidade de vida dos cidadãos, a eficiência dos serviços e operações urbanas e a competitividade, enquanto assegura que ela atenda às necessidades das gerações presente e futura em relação a aspectos econômicos, sociais e ambientais. A qualidade de vida pode ser medida em termos de bem-estar financeiro e emocional de seus habitantes. Para [5], em uma Cidade Inteligente, as tecnologias digitais traduzem-se na melhoria dos serviços públicos para os habitantes e melhor uso dos recursos enquanto reduz os impactos ambientais. Por outro lado, [7] menciona que uma Cidade Inteligente é uma aplicação da noção de Internet das Coisas, uma vez que herda os mecanismos operacionais (subjacentes) de IoT. A Internet das coisas, também chamada de Internet de tudo ou Internet industrial, é um novo paradigma tecnológico em que uma rede global de máquinas e dispositivos (objetos inteligentes) é formada, em que esses objetos inteligentes são capazes de interagir uns com os outros [9]. A IoT permite que os objetos físicos vejam, ouçam, pensem e executem tarefas fazendo com que eles "conversem" uns com os outros, compartilhem informações e coordenem decisões [10]. IoT transforma estes objetos de tradicionais em inteligentes, explorando suas tecnologias subjacentes, como computação ubíqua e disseminada, dispositivos embutidos, tecnologias de comunicação, redes de sensores, protocolos da Internet e aplicações. O núcleo da implementação de Cidades Inteligentes é a Internet das Coisas, ou seja, IoT é o *backbone* de Cidades Inteligentes [5]. Em resumo, IoT pode ser utilizada para construir várias aplicações em Cidades Inteligentes como o transporte inteligente, a gerência de lixo inteligente, o estacionamento inteligente, o cuidado da saúde estrutural de construções, a monitoração da qualidade do ar e da poluição sonora, o atendimento a situações de emergência, entre outros.

A. Arquiteturas para Cidade Inteligentes

Várias arquiteturas estão sendo propostas para IoT, as quais podem ser utilizadas em um contexto urbano. Entretanto, não existe uma arquitetura padronizada ainda.

Em [11], uma arquitetura genérica para IoT em 5 camadas é apresentada, sendo composta das camadas de percepção, rede, Middleware, aplicação e negócios. A camada de percepção é conhecida como camada de dispositivos, consiste de objetos físicos (e.g., tags RFID, código de barras, sensores infravermelhos) e sensores (e.g., localização, temperatura, umidade, vibração) e possui como função tratar da identificação e coleta de informações especificas dos objetos físicos e sensores. A informação coletada é passada para a camada de rede. A camada de rede, também conhecida como camada de transmissão, tem por função transferir com segurança a informação dos dispositivos físicos e sensores para o sistema de processamento da informação. Os meios de transmissão podem ser diversos com fio ou sem fio (e.g., 3G, 4G, Wifi, Bluetooth, infrared, ZigBee). Desta forma, a camada de rede transfere a informação da camada de percepção para a camada do Middleware, sendo esta a responsável pelo gerenciamento de serviços. O Middleware recebe as informações da camada de rede e pode armazenar em um banco de dados, realizar o processamento de informações, tomar decisões automáticas com base nos resultados, entre outros. A camada de aplicação oferece gerenciamento global das aplicações baseada nos objetos de informação processados na camada do Middleware. Por fim, a camada de negócios é responsável pela gerência do sistema IoT ajudando a determinar ações futuras e estratégias de negócios.

Em [7] é proposta uma arquitetura genérica para Cidades Inteligentes, a qual é dividida em 4 camadas: sensoriamento (coleta de dados), transmissão, gerência de dados e aplicação. Na camada de coleta de dados, encontra-se os dispositivos físicos e infraestrutura, redes de sensores sem fio, entre outros. Para conectar as fontes de dados com as estações de gerência, a camada de transmissão age como um *backbone* da arquitetura. A camada de transmissão é a convergência de várias redes de comunicação e consiste de várias tecnologias com fio, sem fio e satélite. Essa camada é dividida em camada de acesso à rede e camada de transmissão de rede. Como tecnologias de acesso a rede que oferecem cobertura de curto alcance estão o Bluetooth, ZigBee, *near field communication* (NFC), *Machine-to-Machine* (M2M), RFID e Zwave. Como exemplo de tecnologias de transmissão de redes com cobertura mais ampla estão a 3G, 4G *long-term evolution* (LTE), 5G e *low-power wide area networks* (LP-WAN). A camada de gerência de dados pode ser caracterizada pela fusão, análise, processamento, armazenamento de dados e gerência de decisão e eventos. A gerência de eventos pode ser realizada por um middleware, pois muitos eventos são gerados pelas aplicações IoT e devem ser gerenciados como parte desta camada [12]. A camada de aplicação, por outro lado, executa decisões sobre dados recebidos da camada de gerência de dados. É a camada que interage com o usuário e contém as aplicações para Cidades Inteligentes.

Em [13] é proposta uma arquitetura para IoT em 3 camadas (percepção, rede e aplicação). A camada de percepção possui como função coletar as informações dos objetos inteligentes, redes de sensores sem fio, entre outros. A camada de rede possui como função transmitir a informação a uma longa distância, sendo formada de várias redes. Além disso, possui como função o processamento inteligente das informações. Dessa forma, a camada de rede compreende as camadas de transmissão e gerência de dados da arquitetura encontrada em

[7]. A camada de aplicação fornece um serviço específico aos usuários com base nos dados.

As arquiteturas apresentadas, basicamente, possuem camadas com funções equivalentes. Desta forma, a Figura 1 apresenta uma adaptação das arquiteturas apresentadas em [7], [11] e [13], a qual será utilizada neste trabalho.

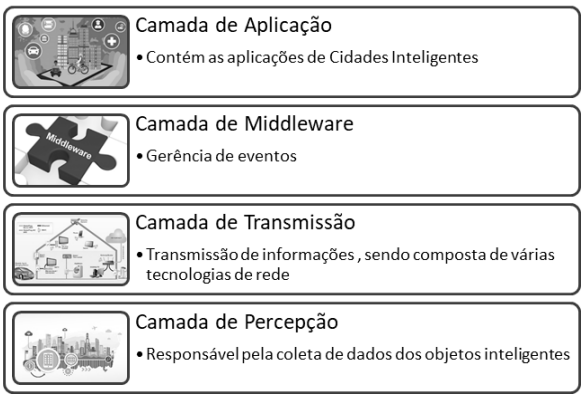

Figura 1 – Arquitetura em camadas de uma Cidade Inteligente Genérica (adaptação de [7], [11] e [13]**).**

Na camada de transmissão, uma plataforma adicional pode ser utilizada como *Fog Computing*. Essa plataforma é necessária para que alguns requisitos de aplicações IoT (e. g., Cidades Inteligentes) sejam conseguidos como baixa latência, distribuição geográfica, mobilidade, um grande número de nós, acesso predominantemente sem fio, aplicações de tempo real e heterogeneidade [14]. As camadas de Middleware e Aplicação, desta forma, podem fazer parte de uma camada única, em um dispositivo inteligente no *Fog*.

III. *FOG COMPUTING*

O advento da Internet das Coisas (IoT) tem possibilitado a introdução de novas arquiteturas de rede que visam aprimorar o paradigma da Computação em Nuvem atualmente implementado. Objetos do dia a dia, cada um com um identificador único, se conectarão automaticamente a várias redes e farão upload de uma grande quantidade de dados de diversos tipos. A infraestrutura da nuvem é incapaz de lidar com o volume, a variedade e a velocidade de dados de IoT, especialmente levando em consideração a latência introduzida ao tentar transferir conjuntos de dados em massa para servidores distantes ou a largura de banda necessária para essas transferências [15]. As redes atuais devem se adaptar para lidar com os requisitos específicos das aplicações de IoT (e.g., baixa latência), uma vez que os recursos podem ser requisitados sob demanda simultaneamente pelos inúmeros dispositivos em diferentes locais [16]. Portanto, faz-se necessário estabelecer um modelo de computação que venha a diminuir essas desvantagens para aplicações IoT como, por exemplo, *Fog Computing*.

Fog Computing é uma plataforma altamente virtualizada que oferece processamento, armazenamento e serviços de rede entre dispositivos finais e os tradicionais Data Centers de Computação em Nuvem (*Cloud Computing Data Centers*) normalmente, mas não exclusivamente, localizados nas bordas da rede [14]. *Fog Computing* é um paradigma computacional distribuído especialmente colocado entre os sensores/dispositivos de IoT e os *Cloud DataCenters* [15]. É uma plataforma projetada principalmente para casos de uso de IoT [17]. O *Fog Computing* estende o paradigma tradicional da Computação em Nuvem até a borda [18]. Essa ampliação acontece ao migrar o processamento de dados para mais perto do local de produção, acelerando a capacidade de resposta do sistema aos eventos e eliminando a ida e volta dos dados para a Nuvem [15]. A proximidade física da infraestrutura de Nuvem com os sensores ligados a recursos de qualquer aplicativo relacionado à IoT, permite latência limitada, além de menor consumo de largura de banda [15]. O descarregamento de grandes conjuntos de dados para a rede principal não é mais uma necessidade. É um paradigma de *Micro Data Center* (MDC) [18]. A arquitetura de *Fog Computing* é mostrada na Figura 2, a qual contém 3 camadas: a camada da Nuvem, a camada *Fog* e a camada dos dispositivos. A camada *Fog* pode conter múltiplas camadas de acordo com os requisitos. Os nodos do *Fog* podem ser pequenas estações base, veículos, pontos de acesso WiFi e terminais do usuário. Os dispositivos escolhem o nodo do *Fog* mais apropriado para associarem-se a ele.

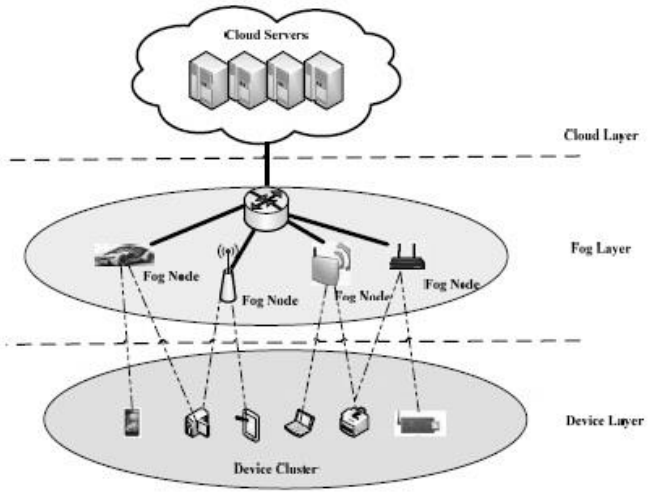

Figura 2 - Arquitetura de *Fog Computing* [17]**.**

A distribuição geográfica em que serviços e aplicações na *Fog* demandam implantação amplamente distribuída é uma característica que torna *Fog Computing* uma extensão não trivial da Nuvem, a qual é mais centralizada [14], [18]. O *Fog*, por meio de *proxies* e pontos de acesso posicionados de forma distribuída pela cidade, pode fornecer informações de qualidade para aplicações de IoT, como aplicações para Cidades Inteligentes. O *Fog* fornece entrega de dados em tempo real, especialmente para serviços relacionados à saúde, sensíveis ao atraso e de emergência. Ele pode ajudar os nós com recursos limitados a descarregar tarefas, pré-processar dados brutos e notificar a nuvem, antes que a nuvem possa adaptá-los aos serviços aprimorados. A principal vantagem do *Fog* é suportar a rede na borda, junto com todos os serviços críticos em atraso que podem ser implantados nessa camada [15]. Serviços de emergência ou de monitoramento da saúde são exemplo em que um grande atraso pode impactar

significantemente em seu desempenho [16]. Situações em que vida está em perigo requerem baixo atraso e espaços de armazenamento seguro por questões de privacidade [19]. Situações de emergência e de monitoramento de saúde demandam comunicação eficiente. Para aplicações IoT com restrições de tempo real, a baixa latência pode ser crucial e, além disso, o número de saltos entre o serviço alocado e o dispositivo final deve ser diminuído [16].

A. Fog Computing em Serviços de Emergência

Em [18], os autores apresentam um serviço baseado em smartphone chamado *Emergency Help Alert Mobile Cloud* (E-HAMC) que é utilizado para notificar a emergência ao departamento adequado que irá tratá-la e aos familiares da vítima. O usuário escolhe por meio de uma interface gráfica o tipo de evento (e.g., acidente, assassinato, incêndio, roubo, terrorismo, cuidados médicos e colapso de construções) e o aplicativo automaticamente envia as notificações, bem como a localização exata da situação de emergência. O serviço de *Fog Computing* é utilizado para o envio dos alertas e para posteriormente enviar ao *Cloud* os dados pré-processados e filtrados. Os autores realizaram avaliação de desempenho baseando-se nas redes Wifi, WiBro, Broadband e 4G. A avaliação considerou especialmente o atraso de upload (arquivos multimídia e dados em massa) do nodo final para o *Fog* e do nodo final para o *Cloud*, sendo que o atraso foi menor do nodo final para o *Fog* justamente por este ficar mais próximo ao nodo final.

Em [15], um sistema de monitoramento de saúde baseado em *Fog Computing* é apresentado, o qual suporta auto monitoramento do paciente e monitoramento on-line e off-line pelo médico do paciente inclusive dentro de hospitais e centros de saúde. A aplicação protótipo foi desenvolvida na plataforma *Spark IoT* e focou na extração, análise e armazenamento de sinais de eletrocardiograma (ECG). Um dispositivo ECG foi colocado nos pacientes e a cada 10 segundos os dados eram analisados e produziam sinais de alertas (e.g., possível hipertrofia ventricular, infarto agudo do miocárdio, arritmia) sendo que o registro e os alertas produzidos com seus *timestamps* eram armazenados na memória interna do dispositivo. Uma aplicação (*Fog Gateway*) instalada nos *smartphones* dos pacientes e conectados sem fio aos dispositivos vestíveis obtinham os registros. À medida que os dados eram recebidos do dispositivo vestível um componente de alerta era ativado para processar e analisar os dados e fornecer os alertas. Os autores avaliaram o tempo de resposta fim a fim na emissão de um comando simples e de relatórios completos calculando a média do tempo de resposta e o tempo máximo de resposta de cada um. Além disso, avaliaram a questão de armazenamento (tamanho de memória necessária no smartphone que executa a aplicação móvel para armazenamento dos registros de ECG) e vazão dos registros de ECG para o sistema.

Em [19], as características da comunicação de Redes Centradas em Conteúdo (*Content-Centric Networking* - CCN) são utilizadas para criar uma rede de *Fogs o*ferecendo troca de dados entre eles. Os autores realizaram simulações com o simulador *ndnSIM* utilizando 9 nodos com distâncias iguais entre eles. Mostraram que o atraso em milissegundos da rede de *Fogs* utilizando CCN foi menor do que as tradicionais redes CCN. Assim, concluem que o desempenho das redes CCN baseadas em *Fog* são melhores e podem ser adaptados ao contexto de emergências em saúde.

Em [16], uma formulação utilizando programação linear é apresentada para o problema de alocação de serviços de aplicações IoT em *Fog Computing* visando avaliar o provisionamento de recursos em cenários de Cidades Inteligentes. Consideram uma aplicação IoT composta de múltiplos serviços de comunicação. Dispositivos finais enviavam requisições para as aplicações IoT através de gateways sem fio. Os gateways se comunicam com a infraestrutura de *Fog Computing*, a qual gerencia um conjunto de recursos computacionais. Em resumo, a formulação visa decidir em qual *Fog* e em qual hardware físico um recurso específico relacionado a uma aplicação IoT pode ser alocado objetivando baixa latência e eficiência energética. Cada serviço pode ser oferecido por uma máquina virtual. Os autores mencionam que um dos principais desafios é a alocação apropriada dos serviços de uma aplicação IoT, uma vez que eles podem ser colocados em uma localização altamente congestionada ou longe dos dispositivos finais, o que pode resultar em uma alta latência de comunicação, além de que restrições de tempo real das aplicações de IoT para Cidades Inteligentes devem ser consideradas. O modelo de programação linear teve como objetivos maximizar o número de requisições das aplicações IoT e o serviço de largura de banda, minimizar as migrações dos serviços durante as interações da simulação, o número de nós computacionais ativos, o número de gateways ativos, o número de saltos entre os nós computacionais e os dispositivos finais, a perda de pacotes pelo caminho. Para as simulações, dentro do escopo do *Testbed Antwerp's City of Thinks*, dois cenários de avaliação foram considerados, um cenário estático e outro cenário dinâmico. O caso de uso do cenário estático é uma aplicação IoT relacionada ao nível de esgoto (sensores para controle do nível da água podem futuramente ser instalados em esgotos) e o caso de uso do cenário dinâmico é uma aplicação IoT para monitorar a qualidade do ar (sensores da qualidade do ar tem sido instalados em veículos) na cidade de Antwerp. Uma área retangular de 216 Km^2, correspondente a área de Antwerp, foi dividida em 9 áreas de 24 Km^2, as quais foram consideradas como possíveis localizações dos recursos de *Fog Computing*. Assim, para a avaliação foram considerados 9 *Fog Clouds*, cada um gerenciando 5 computadores. No cenário estático, 1000 sensores dos níveis da água foram distribuídos aleatoriamente na cidade. Para o cenário dinâmico, 100 carros foram considerados, sendo que cada carro dirigia em uma velocidade média de 30 Km/h. Para ambos os cenários, os dispositivos finais realizavam requisições para as aplicações de IoT. Diferentes modelos de configuração para cada cenário foram avaliados, cada qual visando atingir os objetivos do modelo. Os resultados mostraram que a latência foi minimizada quando o número de saltos entre os nós computacionais e os dispositivos finais foi menor devido aos serviços das aplicações de IoT ficarem mais próxima ao dispositivo final. Entretanto, maior latência foi observada quando alocações de serviços foram consideradas.

Em todos os trabalhos apresentados a questão do atraso foi considerada muito importante para situações de atendimento a emergência e cuidados em saúde.

IV. MIDDLEWARE BASEADOS EM EVENTOS

O paradigma *publish-subscribe* é frequentemente utilizado como um mecanismo de comunicação em aplicações baseadas em eventos e está difundindo devido à expansão dos serviços e aplicações de IoT [20]. Entretanto, aplicações para Cidades Inteligentes impõe novos requisitos para middlewares *publish-subscribe*. Seis são as áreas relacionadas ao domínio de Cidades Inteligentes, dentre elas: economia, pessoas, governança, mobilidade, ambiente e moradia [20]. Cada área possui diferentes requisitos para middlewares *publish-subscribe*. Logo, o contexto a ser utilizado o middleware deve ser analisado. Em [20] são apresentadas as características requeridas para um middleware *publish-subscribe* a ser utilizado em Cidades Inteligentes, tanto para aplicações que utilizam dados estáticos quanto móveis. Dentre os requisitos apontados constam: arquitetura descentralizada, baixo atraso na entrega de pacotes, suporte à mobilidade, semântica de entrega do tipo *best-effort* (com exceção dos serviços de emergência e desastre), endereçamento IP, assinatura baseada em conteúdo, qualidade de serviço na aquisição dos dados, entre outros. Os autores analisaram duas aplicações (estacionamento inteligente e *crowd sensing* móvel) para apresentação dessas características. Para algumas aplicações de Cidades Inteligentes é importante que ocorra alguma forma de seleção dos assinantes escritos para a recepção de cada evento, como é o caso de serviços de emergência e desastre.

Um estudo de trabalhos relacionados a middlewares *publish-subscribe* foi realizado buscando especialmente identificar se os middlewares tratavam da notificação de eventos a um subconjunto dos assinantes mais adequados, dentro de um grupo de assinantes cadastrados para receber os eventos de interesse. Os middlewares [21], [22], [23], [24], Mires [25], [26], [27], [28], [29] e [30] foram analisadas em relação a esse quesito e se mostraram pouco flexíveis no envio de eventos somente a um subconjunto de assinantes. Além disso, [27], [29] e [23] possuem arquitetura centralizada, [22] não suporta mobilidade, [21], [25] e [30] não suportam assinatura baseada em conteúdo. Os middlewares [26] e [24] foram projetados para uso somente em redes de sensores sem fio, Siena tem uso mais geral; entretanto, o roteamento de eventos não permite seleção de assinantes.

V. O MIDDLEWARE EVENT TO BEST SUBCRIBERS (E2BS)

O middleware E2BS implementa um modelo de eventos que visa o envio de eventos a um subconjunto dos melhores assinantes, ou seja, os assinantes mais adequados. Foi proposto visando o atendimento a situações de emergência em Cidades Inteligentes.

A. *Entidades básicas do middleware E2BS*

Brokers e unidades móveis são as entidades básicas do middleware E2BS. O *broker* publica eventos e as unidades móveis emitindo subscrições consomem os eventos de acordo com o modelo de eventos implementado pelo middleware. Mais especificamente:

- Os *brokers* são processos de software que tomam a decisão sobre quais unidades móveis devem ser acionadas para atender a emergência. Podem ser instalados em um servidor ou objeto inteligente com capacidade de processamento e comunicação.
- As unidades móveis são veículos devidamente equipados com material de emergência e aparelhos e que possuem pessoal qualificado e preparado para os diversos tipos de emergência. Elas se deslocam até a ocorrência e prestam o atendimento necessário dependendo da ocorrência.

Os *brokers* oferecem pontos de acesso para subscrições e para o acionamento das situações de emergência. Desta forma, um agente externo se faz necessário para que o middleware possa publicar os eventos, ou seja, notificar as unidades móveis mais adequadas. Esse agente externo é um atendente de situações de emergência que se comunica com os *brokers*.

B. *Arquitetura Fog Computing para o acionamento de unidades móveis em situações de emergência*

A arquitetura de *Fog Computing* utilizada como cenário de utilização do middleware é composta por um conjunto de *brokers (smarts dispatching)* que ficam distribuídos em diferentes regiões da cidade, um para cada região sem sobreposição. Assim:

1. Uma rede de *Fogs* é criada para o envio de eventos;
2. Um *broker* executa em um *Fog node* dentro de um *Fog*;
3. As unidades móveis escolhem o *Fog* mais próximo (isto é, o *broker* associado ao *Fog* escolhido) para subscreverem-se de modo a receberem os eventos;
4. A comunicação entre os *Fog nodes* acontece através da Internet, assim cada *Fog node* pode se comunicar diretamente com outro *Fog node*;
5. A comunicação entre as unidades móveis e os *Fog nodes* acontece através de rede sem fio (4G);
6. O atendente das situações de emergência se comunica com os *Fogs* através da Internet.

A arquitetura de *Fog Computing* está representada na Figura 3. Cada antena abrange uma determinada região da cidade. Desta forma, os *brokers* ficam distribuídos em regiões distintas nos *Fog nodes*. Vários pontos de referência estão presentes em cada *Fog*, ou seja, em cada região abrangida pelo *broker*. Na Figura 3, a estrutura de *Cloud* não é ilustrada, mas pode ser utilizada para armazenamento de dados permitindo uma posterior análise dos mesmos.

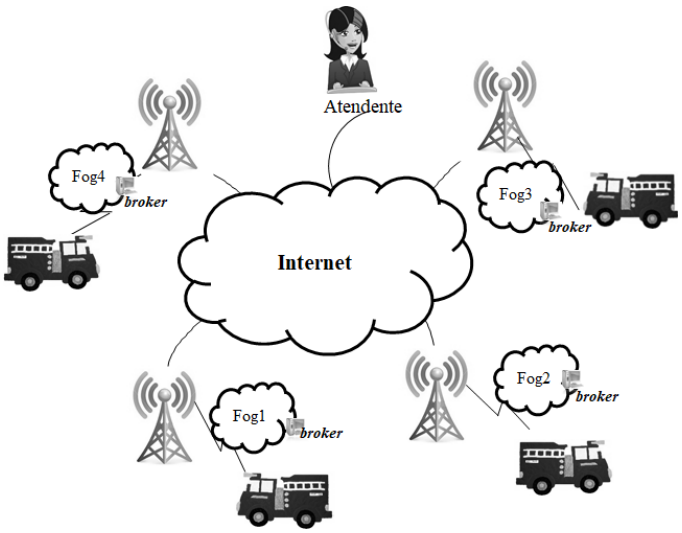

Figura 3 - Uma arquitetura de *Fog Computing* para o middleware E2BS.

Acima da rede de *Fogs*, várias redes overlay de *brokers*, as quais são utilizadas para a notificação de eventos, são formadas conectando *brokers* e unidades móveis. As redes overlay são os fundamentos do middleware E2BS. Desta forma, para cada serviço (e.g., bombeiro, polícia, SAMU) e ponto de referência uma rede overlay é criada em cada *broker*, sendo representada por uma árvore lógica de disseminação de eventos. Para a Figura 3, considerando somente um serviço e ponto de referência em cada *Fog*, por consequência em cada *broker*, quatro redes overlay seriam criadas, uma em cada *broker*. As árvores lógicas de disseminação de eventos dos *brokers* presente no *Fog1(broker B1)* e *Fog2 (broker B2)* são apresentadas na Figura 4. As unidades móveis (neste contexto, os caminhões de bombeiro) seguem a numeração de seu respectivo *Fog*. As árvores dos *brokers* B3 e B4 seguem o mesmo padrão apresentado na Figura 4.

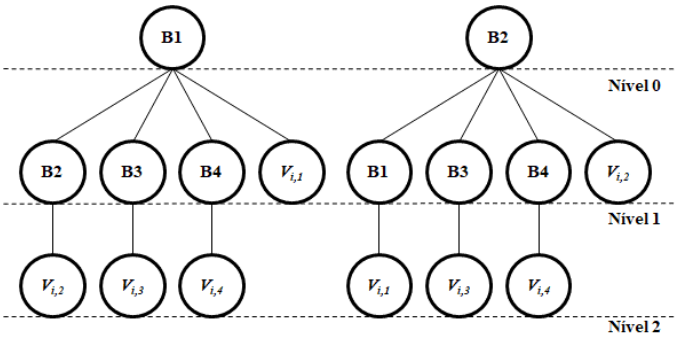

Figura 4 – Árvore lógica de disseminação de eventos de B1 e B2.

O *broker* de nível 0 é um *broker* raiz (responsável pela região em que acontece a situação de emergência) e o *broker* de nível 1 é um *broker* intermediário em uma árvore lógica de disseminação de eventos.

C. O modelo de eventos Event to Best Subscribe (E2BS)

Um modelo de eventos consiste de um conjunto de regras descrevendo um modelo de comunicação, o qual é baseado em eventos [31]. Assim, o modelo de eventos E2BS especifica um conjunto de regras descrevendo como a comunicação baseada em eventos acontece entre *brokers* e unidades móveis. A especificação do modelo segue a abordagem apresentada em [32], a qual é descrita abaixo.

$\Gamma = \{S_1, ..., S_n\}$: o conjunto dos serviços disponíveis, $n \geq 1$.

$\Phi = \{\alpha_1, ..., \alpha_r\}$: o conjunto dos pontos geográficos de referência de uma cidade (cada ponto corresponde ao centro de uma região da cidade), $r \geq 1$.

ε: a situação de emergência.

$\Psi \in \Phi$: o ponto geográfico de referência associado a ε.

$\theta_i = \{v_{i,1}, ..., v_{i,m}\}$: o conjunto das unidades móveis de $S_i \in \Gamma$, $m \geq 1$.

$\Delta_{i,\varepsilon} \in \mathbb{N}$: a intensidade de $S_i \in \theta_i$ para atuar em ε, dada em número de unidades móveis.

$$\beta_{i,j,\varepsilon} = \begin{cases} 1, & \text{se } v_{ij} \in \theta_i \text{ está selecionada para atender } \varepsilon \\ 0, & \text{caso contrário} \end{cases}$$

$\Omega_{i,j,\alpha_k} \in [0,1]$: o nível de adequação da unidade móvel $v_{i.j} \in \theta_i$ relativo a $\alpha_k \in \Phi$.

Para atender ε, $\beta_{i,j,\varepsilon}, 1 \leq j \leq |\theta_i|$ precisa ser determinado para cada serviço $S_i \in \Gamma$, tal que o nível de adequação geral das unidades móveis selecionadas de θ_i para atuar em ε, denominado $\lambda_{i,,\varepsilon}$, seja maximizado. Assim, o modelo possui a seguinte formulação:

$$\text{Maximizar} \quad \lambda_{i,\varepsilon} = \sum_{j=1}^{|\theta_i|} \left(\Omega_{i,j,\Psi_\varepsilon} x \beta_{i,j,\varepsilon}\right)$$

$$\text{Sujeito a:} \quad \sum_{j=1}^{|\theta_i|} \beta_{i,j,\varepsilon} \leq \Delta_{i,\varepsilon}$$

Dessa forma, o subconjunto de unidades móveis do serviço $S_i \in \Gamma$ selecionadas para atuação é dado por:

$$\{v_{i,j} \in \theta_i \mid \beta_{i,j,\varepsilon} = 1\}$$

A Figura 5 mostra duas regiões da cidade, sendo que cada região foi mapeada em 4 células quadradas iguais com 4 pontos geográficos de referência em cada uma. A região em que acontece uma ocorrência relacionada a emergência é a região com fundo branco. Supondo que uma situação de emergência aconteça no ponto P = (0.5, *0.75*), seu ponto de referência será $\Psi_\varepsilon = p = (1, 1)$. Isso porque o ponto $p = (1, 1)$ é referência para a área definida pelo quadrado com vértices (*0, 0*), (*0, 1*), (*1, 0*) e (*1,1*) e a emergência aconteceu dentro deste espaço. O serviço S_i demandado dispõe de três unidades móveis para atuação: $\theta_i = \{v_{i,1}, v_{i,2}, v_{i,3}\}$. Logo, se $\Delta_{i,\varepsilon} = 2$, considerando o modelo de eventos proposto, as unidades mais adequada em relação ao ponto $\Psi_\varepsilon = (1, 1)$ serão acionadas. Desta forma, se os níveis de adequação forem $\Omega_{i,1,\Psi\varepsilon} = 0.5$,

$\Omega_{i,2,\Psi\varepsilon}$ = 0.8 e $\Omega_{i,3,\Psi\varepsilon}$ = 0.2, as unidades móveis $\Omega_{i,1,\Psi\varepsilon}$ e $\Omega_{i,2,\Psi\varepsilon}$ serão selecionadas para atender ao chamado.

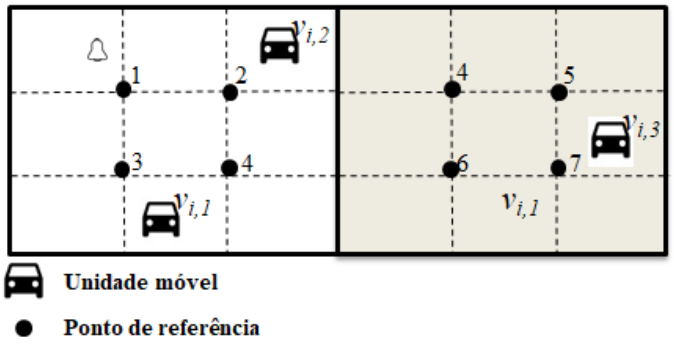

Figura 5 – Exemplo de situação emergencial.

Para o cálculo dos níveis de adequação de cada unidade móvel $v_{i,j}$ a cada ponto de referência α_r foi considerado a distância e a situação de trânsito entre a unidade móvel $v_{i,j}$ e o ponto de referência α_r (coeficiente de trânsito), além das condições da unidade móvel (disponível ou indisponível para o atendimento a chamados). A formalização do cálculo dos níveis de adequação, bem como de todos os algoritmos utilizados no desenvovimento do trabalho podem ser consultados em [33].

D. Atualização dos níveis de adequação

O *broker* recebe as subscrições das unidades móveis e calcula os níveis de adequação de cada unidade móvel em relação a cada ponto de referência de sua região e das outras regiões da cidade. Os níveis de adequação relacionados aos pontos de referência da região do *broker* são armazenados e os outros níveis de adequação calculados são enviados aos respectivos *brokers* de acordo com sua região de abrangência. Entretanto, se o *broker* tiver várias unidades móveis ele envia somente os maiores níveis de adequação de cada ponto de referência ao *broker* do qual o ponto faz parte. Isso para que quando aconteça uma situação de emergência ele possa acionar a unidade móvel mais apropriada diretamente ou através de outro *broker*. Para melhorar a precisão do acionamento da unidade móvel, os níveis de adequação podem ser atualizados constantemente seguindo assim uma abordagem pró-ativa. Isto significa que quando acontecer uma situação de emergência o *broker* já terá condições de tomar a decisão sobre quais unidades móveis devem se deslocar até a ocorrência para seu atendimento. Os níveis de adequação das unidades móveis já estão à disposição do *broker* para esta escolha.

E. Primitivas do middleware

As mensagens enviadas por um *broker* à(s) unidade(s) móvel(is) são:

- *announce*: mensagem de anúncio de serviço enviada em *broadcast* por um *broker* para todas as unidades móveis sob sua área de abrangência.
- *notification*: mensagem que aciona o atendimento a situação de emergência.

As mensagens enviadas por uma unidade móvel para um *broker* são:

- *subscribe*: mensagem de inscrição (subscrição) para um serviço respondendo a uma mensagem *announce*.
- *bind*: mensagem de vínculo emitida de forma periódica informando que a unidade móvel mantém-se na região de cobertura do *broker*. Enviada somente se a mensagem *subscribe* for recebida pela unidade móvel.
- *unsubscribe*: mensagem que sinaliza que a unidade móvel está cancelando sua subscrição para um serviço específico.
- *ack_notification*: mensagem de confirmação de acionamento em resposta a uma mensagem *notification*.

As mensagens enviadas entre *brokers* são:

- *publish*: mensagem enviada por um *broker* raiz para um *broker* intermediário solicitando que ele acione uma de suas unidades móveis para atendimento à emergência.
- *ack_publish*: mensagem que confirma o acionamento de unidade móvel. Esta mensagem é enviada por um *broker* intermediário a um *broker* raiz em resposta a uma mensagem *publish*.
- *update*: mensagem enviada por um *broker* intermediário a um *broker* raiz contendo os níveis de adequação atualizados das unidades móveis de sua região. Os níveis de adequação enviados dizem respeito aos maiores valores entre todas as unidades móveis de um *broker* intermediário para cada ponto de referência pertencente ao *broker* raiz.

Há uma mensagem que é enviada por um atendente da situação de emergência para um *broker* raiz. Essa mensagem é uma mensagem de ativação de situação de emergência (*publish_activation*). Ela indica que o *broker* deve acionar uma quantidade *n* de unidades móveis, pertencentes a determinado serviço, de sua árvore lógica de disseminação de eventos para atender a emergência em um ponto específico. A cada unidade móvel acionada, o atendente que enviou a mensagem recebe uma mensagem de confirmação *ack_publish_activation*. Desta forma, o atendente controla se o pedido é cumprido. Enquanto o pedido não é cumprido, o agente envia novas mensagens de *publish_activation* de 2 em 2 minutos durante o período de 1 hora. Se durante o reenvio de um pedido não atendido, uma nova mensagem é gerada, a mesma é colocada em uma fila para posterior envio.

F. Simulador

Para a validação do middleware E2BS o simulador Sinalgo foi utilizado (https://disco.ethz.ch/apps) devido as suas características, como: suporte à mobilidade e ao envio de eventos, documentação e tutoriais, código fonte das classes do simulador e suporte a uma linguagem de programação orientada a objetos.

Para as simulações, alguns parâmetros do simulador foram configurados. O modelo de mobilidade dos *brokers* adotado foi sem mobilidade (*NoMobility*). As unidades móveis utilizaram uma extensão da classe *PerfectRWP*, no qual as unidades móveis selecionam um ponto aleatório da área de simulação (algumas vezes, o ponto da situação de emergência) e se movimentam até ele. Os nós, isto é, os *brokers*, as unidades móveis e o atendente poderiam enviar mensagens enquanto estavam recebendo-as. Os *brokers* foram distribuídos em grade e as unidades móveis em pontos fixos na área de simulação. O tamanho da área de simulação utilizado foi de 1000 pixels, o que equivaleu a uma área urbana de uma cidade de 36Km^2. Para as simulações dos cenários escolhidos, os parâmetros configurados encontram-se na Tabela 1.

Tabela 1 – Parâmetros dos cenários escolhidos.

Cada *round* do simulador equivalendo a 1 segundo (1s)
Cada simulação correspondendo a 1 dia (86.400s)
Área de cobertura dos *brokers* em 100% da área urbana
Tipo de serviço: bombeiro e policial
Quantidade de unidades móveis para atendimento as emergências: variável aleatória entre 1 e 3
Tempo de deslocamento: dependente da distância entre a unidade móvel e o ponto da situação de emergência e das condições de trânsito (leve, moderado e intenso). A velocidade das unidades móveis simulada ficou entre 30Km/h e 60Km/h
Condições de trânsito para cálculo dos níveis de adequação: (i) Para trânsito leve: coeficiente entre 0 à 0.10; (ii) Para trânsito moderado: coeficiente entre 0.10 à 0.25; e, (iii) Para trânsito intenso: coeficiente entre 0.25 à 0.50
Tempo de atendimento da unidade móvel: variável aleatória com distribuição exponencial negativa entre 2 minutos e 30 segundos a 8 horas para bombeiro e entre 20 a 40 minutos para polícia
Número de *Fogs* e pontos de referência por *broker*: 4
Número de *Fog Nodes*: 9
Intervalo de envio das mensagens de *announce e bind*: 1 minuto
Representação do tempo na forma *hora:minuto:segundo*

Além dos parâmetros acima, um modelo de chamadas foi criado e seguiu a distribuição Poisson. Para o modelo definiu-se: (i) o próximo tempo de acionamento; (ii) o ponto de referência da emergência, e; (iii) o número de unidades móveis necessárias para o atendimento da emergência. Para o serviço de bombeiro, o intervalo médio de acionamento foi de 2 horas e para a polícia foi de 2 horas e 40 minutos. Para a definição dos parâmetros dos cenários, a Corporação X e a Corporação Y (nomes fictícios devido a termo de confidencialidade firmado) forneceram um resumo dos dados reais (ano de 2016) das situações de emergência.

G. Casos de uso

Cabe salientar que os casos de uso foram realizados visando provar ou refutar a hipótese de que o envio de eventos a um subconjunto de assinantes é melhor do que uma escolha aleatória. Neste sentido, dois cenários foram simulados, um parcialmente estático (bombeiro) e outro dinâmico (serviço policial). Para ambos os cenários foram realizadas simulações com escolha aleatória de unidades móveis (isto é, qualquer unidade móvel com nível de adequação diferente de zero) e com escolha das unidades móveis mais apropriadas. O método de validação foi comparação de cenários e as métricas utilizadas foram o tempo de deslocamento da unidade móvel até a situação de emergência e o tempo total de atendimento da unidade móvel, isto é, o tempo de deslocamento somado ao tempo de atendimento. Para o serviço bombeiro, o tempo de retorno à base foi considerado também. O tempo de deslocamento foi uma métrica escolhida para provar ou refutar a hipótese de pesquisa, pois o valor do nível de adequação influencia diretamente no tempo de deslocamento. Para a coleta de dados foram realizadas várias simulações. Para cada conjunto de valores de unidade móvel, 3, 6, 9, 12 e 15, foram executadas 10 simulações e obtidas as médias. Em todas as simulações, no instante inicial, as unidades móveis foram dispostas exatamente nos mesmos lugares na área de simulação. O número de unidades móveis foi escolhido levando em consideração os dados coletados de situações reais. Foram executadas 200 simulações no total.

1) Acionamento de veículos de bombeiro

Os veículos de bombeiro possuem um posto de atendimento e se movimentam até a situação de emergência quando são acionados, retornando posteriormente ao seu respectivo posto de atendimento. A Figura 6 mostra a média do tempo de deslocamento de veículos de bombeiro até à situação de emergência considerando a escolha das unidades móveis de forma aleatória e a escolha das unidades móveis mais apropriadas. Como pode ser observado no comportamento do gráfico, o tempo de deslocamento na escolha aleatória foi maior em todos os casos. Isso porque a escolha aleatória faz com que muitas vezes uma unidade móvel distante seja acionada.

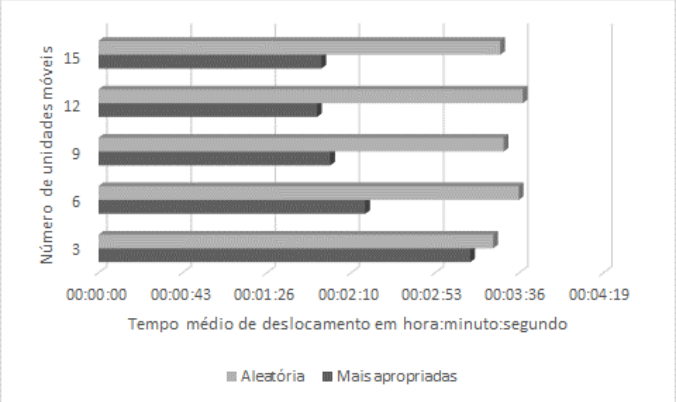

Figura 6 – Média do tempo de deslocamento dos veículos de bombeiro até a emergência.

O gráfico da Figura 7 mostra a média do tempo total do atendimento da emergência considerando a média do tempo de deslocamento até a emergência, a média do tempo de atendimento e a média do tempo de deslocamento de volta até o posto de atendimento, a qual é importante para reposição de materiais de socorro, reabastecimento de água. Na Figura 7, fica mais claro que o tempo de atendimento influencia no comportamento do gráfico. Assim, o tempo total de

atendimento com escolha aleatória de veículos de bombeiro ficou maior em 100% dos casos.

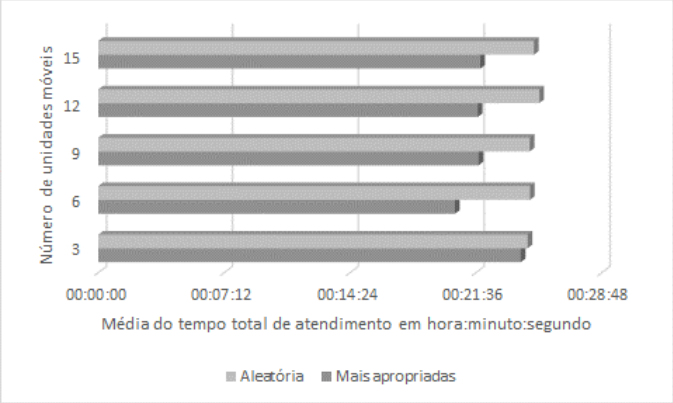

Figura 7 – Média do tempo total de atendimento dos veículos de bombeiro.

2) Acionamento de viaturas policiais

Viaturas policiais se movimentam pela cidade parando eventualmente, por certo tempo, em locais que requeiram atenção. A Figura 8 mostra um gráfico com as médias dos tempos de deslocamento das viaturas policiais até à emergência, considerando a escolha das unidades móveis de forma aleatória e a escolha das unidades móveis mais apropriada.

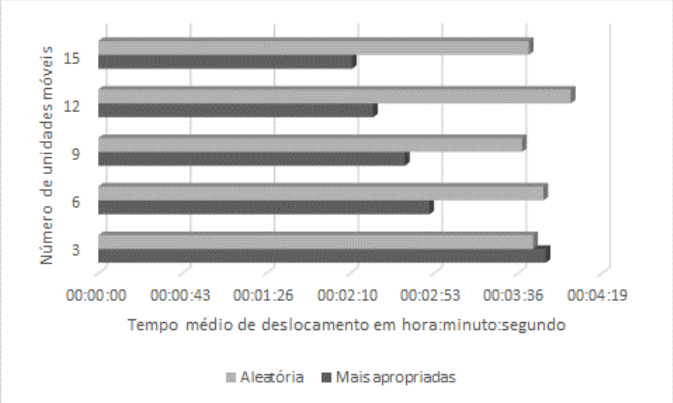

Figura 8 - Média do tempo de deslocamento das viaturas policiais até a emergência.

Como pode ser observado no gráfico da Figura 8, a escolha das unidades móveis de forma aleatória apresentou um tempo maior de deslocamento em relação à escolha das unidades móveis mais apropriadas. O único caso em que a escolha das unidades móveis mais apropriadas teve um tempo maior de deslocamento foi com 3 unidades móveis. Isso se deve a vários fatores, como a mobilidade associada ao número reduzido de unidades móveis disponíveis e ao número de unidades móveis solicitadas. O número reduzido de unidades móveis faz com que, muitas vezes, a unidade móvel mais distante seja acionada, especialmente quando o número de unidades móveis solicitadas é maior ou igual ao número de unidades móveis disponíveis. Assim, qualquer unidade móvel, independente da escolha ser aleatória ou das mais apropriadas, é enviada até a situação de emergência.

O gráfico da Figura 9 mostra a média do tempo total de atendimento das viaturas policiais (tempo de deslocamento até à emergência mais o tempo de atendimento). As unidades móveis depois do atendimento não retornam a um posto de atendimento como no bombeiro, mas podem permanecer por um tempo na região em que se encontram. No gráfico (Figura 9) pode-se perceber que a média do tempo total é maior na maioria dos casos.

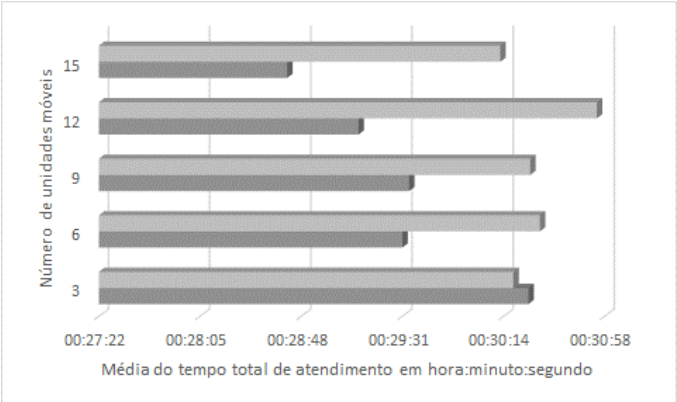

Figura 9 - Média do tempo total de atendimento das viaturas policiais.

Com os resultados apresentados, pode-se dizer que em um cenário parcialmente estático, como é o caso dos bombeiros, e em um cenário dinâmico, como é o serviço policial, a escolha das melhores unidades móveis para atendimento a situação de emergência se mostrou melhor do que uma escolha aleatória.

VI. CONSIDERAÇÕES FINAIS

A Computação em Nuvem, juntamente com a computação *Fog*, pode desempenhar um papel muito importante no gerenciamento geral de emergências. A plataforma de computação em nuvem fornece servidores virtuais gerenciáveis e escalonáveis, recursos de armazenamento, recursos de computação, redes virtuais e largura de banda de rede, de acordo com o requisito e a acessibilidade do cliente. Também fornece solução para processar o conteúdo distribuído. Além disso, os dados podem ser acessados sem o problema de manter grandes dispositivos de armazenamento e computação. Desta forma, neste trabalho foi utilizada uma arquitetura baseada em *Fog Computing* para validar o middleware E2BS em situações de emergência. O middleware E2BS gerencia o envio de eventos aos assinantes mais adequados em Cidades Inteligentes, o qual foi validado em dois cenários com diferentes características. Os resultados mostraram que a escolha das unidades móveis mais adequadas é melhor do que uma escolha aleatória para ambos os cenários apresentados. Assim, mostrou-se que a escolha das unidades móveis mais apropriadas é melhor do que uma escolha aleatória. Como trabalhos futuros pretende-se utilizar novas métricas para o cálculo dos níveis de adequação e adotar uma abordagem reativa para o acionamento das unidades móveis.

AGRADECIMENTOS

A Coordenação de Aperfeiçoamento de Pessoal (CAPES) pela bolsa auxílio mensalidade.

REFERÊNCIAS

[1] A. Alkandari and I. F. T. Alshekhly, “Smart Cities : Survey,” *J. Adv. Comput. Sci. Technol. Res.*, vol. 2, no. 2, pp. 79–90, 2012.

[2] O. Habitat, *Estado de las Ciudades de América Latina y el Caribe 2012, Rumbo a una nueva transición urbana*, no. 17. 2012.

[3] A. Ortegon-Sanchez and N. Tyler, “Constructing a Vision for an ‘Ideal’ Future City: A Conceptual Model for Transformative Urban Planning,” *Transp. Res. Procedia*, vol. 13, pp. 6–17, 2016.

[4] H. Schaffers, N. Komninos, M. Pallot, B. Trousse, M. Nilsson, and A. Oliveira, “Smart cities and the future internet: Towards cooperation frameworks for open innovation,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 6656, pp. 431–446, 2011.

[5] S. P. Mohanty, U. Choppali, and E. Kougianos, “Everything you wanted to know about smart cities: The Internet of things is the backbone,” *IEEE Consum. Electron. Mag.*, vol. 5, no. 3, pp. 60–70, 2016.

[6] C. Yin, Z. Xiong, H. Chen, J. Wang, D. Cooper, and B. David, “A literature survey on smart cities,” *Sci. China Inf. Sci.*, vol. 58, no. 10, pp. 1–18, 2015.

[7] B. N. Silva, M. Khan, and K. Han, “Towards sustainable smart cities: A review of trends, architectures, components, and open challenges in smart cities,” *Sustain. Cities Soc.*, vol. 38, no. August 2017, pp. 697–713, 2018.

[8] ITU-T Focus Group on Smart Sustainable Cities, “TR-Definitions,” *Smart sustainable cities: An analysis of definitions*. TELECOMMUNICATION STANDARDIZATION SECTOR OF ITU, p. 11, 2014.

[9] I. Lee and K. Lee, “The Internet of Things (IoT): Applications, investments, and challenges for enterprises,” *Bus. Horiz.*, vol. 58, no. 4, pp. 431–440, 2015.

[10] A. Al-fuqaha, S. Member, M. Guizani, M. Mohammadi, and S. Member, “Internet of Things : A Survey on Enabling,” vol. 17, no. 4, pp. 2347–2376, 2015.

[11] R. Khan, S. U. Khan, R. Zaheer, and S. Khan, “Future Internet : The Internet of Things Architecture , Possible Applications and Key Challenges,” 2012.

[12] M. A. Razzaque, M. Milojevic-jevric, A. Palade, and S. Clarke, “Middleware for Internet of Things : A Survey,” vol. 3, no. 1, pp. 70–95, 2016.

[13] Zhihong Yang, Yufeng Peng, Yingzhao Yue, Xiaobo Wang, Yu Yang, and Wenji Liu, “Study and application on the architecture and key technologies for IOT,” *2011 Int. Conf. Multimed. Technol.*, pp. 747–751, 2011.

[14] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, “Fog computing and its role in the internet of things,” *Proc. first Ed. MCC Work. Mob. cloud Comput. - MCC '12*, p. 13, 2012.

[15] O. Akrivopoulos, I. Chatzigiannakis, C. Tselios, and A. Antoniou, “On the Deployment of Healthcare Applications over Fog Computing Infrastructure,” *2017 IEEE 41st Annu. Comput. Softw. Appl. Conf.*, pp. 288–293, 2017.

[16] J. Santos, T. Wauters, B. Volckaert, and F. De Turck, “Resource provisioning for IoT application services in smart cities,” *2017 13th Int. Conf. Netw. Serv. Manag.*, pp. 1–9, 2017.

[17] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, “A Survey on Mobile Edge Networks: Convergence of Computing, Caching and Communications,” *IEEE Access*, vol. 5, pp. 6757–6779, 2017.

[18] M. Aazam and E. N. Huh, “E-HAMC: Leveraging Fog computing for emergency alert service,” *2015 IEEE Int. Conf. Pervasive Comput. Commun. Work. PerCom Work. 2015*, pp. 518–523, 2015.

[19] D. Guibert, J. Wu, S. He, M. Wang, and J. Li, “CC-fog: Toward content-centric fog networks for E-health,” *2017 IEEE 19th Int. Conf. e-Health Networking, Appl. Serv.*, pp. 1–5, 2017.

[20] A. Antonic, M. Marjanovic, P. Skocir, and I. P. Zarko, “Comparison of the CUPUS middleware and MQTT protocol for smart city services,” *Proc. 13th Int. Conf. Telecommun. ConTEL 2015*, 2015.

[21] M. Musolesi, C. Mascolo, and S. Hailes, “EMMA: Epidemic Messaging Middleware for Ad hoc networks,” *Pers. Ubiquitous Comput.*, vol. 10, no. 1, pp. 28–36, 2006.

[22] P. R. Pietzuch, “Hermes: A scalable event-based middleware,” *Univ. Cambridge Comput. Lab.*, no. 590, 2004.

[23] J. R. Silva *et al.*, “PRISMA: A publish-subscribe and resource-oriented middleware for wireless sensor networks,” *Adv. Int. Conf. Telecommun. AICT*, vol. 2014–July, no. July, pp. 87–97, 2014.

[24] S. Lai, J. Cao, and Y. Zheng, “PSWare: A publish / subscribe middleware supporting composite event in wireless sensor network,” *7th Annu. IEEE Int. Conf. Pervasive Comput. Commun. PerCom 2009*, 2009.

[25] E. Souto *et al.*, “Mires: A publish/subscribe middleware for sensor networks,” *Pers. Ubiquitous Comput.*, vol. 10, no. 1, pp. 37–44, 2006.

[26] P. Costa *et al.*, “The RUNES middleware for networked embedded systems and its application in a disaster management scenario,” *Proc. - Fifth Annu. IEEE Int. Conf. Pervasive Comput. Commun. PerCom 2007*, pp. 69–78, 2006.

[27] R. Meier and V. Cahill, “STEAM: Event-based middleware for wireless ad hoc networks,” *Proc. - Int. Conf. Distrib. Comput. Syst.*, vol. 2002–Janua, pp. 639–644, 2002.

[28] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf, “Design and Evaluation of a Wide-Area Event Notification Service,” vol. 19, no. 3, pp. 332–383, 2001.

[29] A. D. R. L. Ribeiro, L. C. Freitas, C. R. L. Francês, and C. W. A. Costa, “SensorBus – A Policy-Based Middleware for Wireless Sensor Networks,” vol. 6, no. 7, pp. 647–654, 2008.

[30] P. Boonma and J. Suzuki, “TinyDDS: An Interoperable and Configurable Publish/Subscribe Middleware for Wireless Sensor Networks,” *Wirel. Technol. Concepts, Methodol. Tools Appl.*, pp. 819–846, 2011.

[31] R. Meier and V. Cahill, “On event-based middleware for location-aware mobile applications,” *IEEE Trans. Softw. Eng.*, vol. 36, no. 3, pp. 409–430, 2010.

[32] S. C. L. Hernandes, A. Calsavara, and L. A. P. L. Jr, “Acionamento Inteligente de Unidades Móveis em Situações de Emergência em Cidades,” in *Anais do I CoUrb - Workshop de Computação Urbana do XXXV SBRC*, 2017, pp. 58–71.

[33] S. C. L. Hernandes, A. Calsavara, and L. L. Jr, “Serviços de Emergência em Cidades Inteligentes: o Problema de Acionamento de Unidades Móveis,” *Rev. Eletrônica Sist. Informação*, vol. 16, no. 2, pp. 1–30, 2017.