Revista Eletrônica de Sistemas de Informação
ISSN 1677-3071

v. 16, n. 2

mai-ago 2017 - Edição temática sobre Computação Urbana

DOI: https://doi.org/10.21529/RESI.2017.1602

Sumário

Computação Urbana

Esta revista é (e sempre foi) eletrônica para ajudar a proteger o meio ambiente, mas, caso deseje imprimir esse artigo, saiba que ele foi editorado com uma fonte mais ecológica, a *Eco Sans*, que gasta menos tinta.

This journal is (and has always been) electronic in order to be more environmentally friendly. Now, it is desktop edited in a single column to be easier to read on the screen. However, if you wish to print this paper, be aware that it uses Eco Sans, a printing font that reduces the amount of required ink.

SERVIÇOS DE EMERGÊNCIA EM CIDADES INTELIGENTES: O PROBLEMA DE ACIONAMENTO DE UNIDADES MÓVEIS

EMERGENCY SERVICES IN SMART CITIES: THE PROBLEM OF ACTIVATING MOBILE UNITS

(artigo submetido em agosto de 2017)

Sediane C. Lunardi Hernandes
Mestre em Ciência da Computação pela Pontifícia Universidade Católica do Rio Grande do Sul (PUC-RS)
Professora da Univ. Tecnológica Federal do Paraná, campus Guarapuava (UTFPR)
sedianec@utfpr.edu.br

Alcides Calsavara
Doutor em Ciência da Computação pela Newcastle University
Professor do Programa de Pós-Graduação em Informática (PPGIa) da Pontifícia Universidade Católica do Paraná (PUC-PR)
alcides@ppgia.pucpr.br

Marcelo Eduardo Pellenz
Doutor em Engenharia Elétrica pela Universidade Estadual de Campinas (Unicamp)
Professor do Programa de Pós-Graduação em Informática (PPGIa) da Pontifícia Universidade Católica do Paraná (PUC-PR)
marcelo@ppgia.pucpr.br

Luiz Augusto de Paula Lima Júnior
Doutor em Ciência da Computação pelo Université d'Evry-Val-d'Essonne
Professor do Programa de Pós-Graduação em Informática (PPGIa) da Pontifícia Universidade Católica do Paraná (PUC-PR)
laplima@ppgia.pucpr.br

RESUMO

Cidades inteligentes buscam melhorar a qualidade de vida dos cidadãos por meio do uso das Tecnologias da Informação e Comunicação (TICs). Esforços vem sendo realizados para que sistemas computacionais sejam desenvolvidos para tratar os grandes problemas das cidades. Um dos serviços importantes é o atendimento a chamados de emergência (ex.: polícia e hospitais), o qual demanda o deslocamento de unidade móveis. O problema consiste em realizar o acionamento das unidades móveis mais adequadas para atuar em uma situação em particular, considerando, entre outros fatores, o tempo necessário para se chegar ao local da emergência. Um *middleware* pode ser utilizado nesse contexto para oferecer suporte a esse acionamento, gerenciando, desta forma, a notificação de eventos relacionados a situações de emergência. Vários *middlewares* baseados em eventos foram analisados para verificar sua adequação quanto ao suporte para o acionamento das unidades móveis mais adequadas para atuar em situações de emergência em cidades inteligentes. Entretanto, se mostraram pouco flexíveis em relação ao envio somente para as unidades móveis mais adequadas. Assim, este artigo apresenta e formaliza o problema de acionamento de unidades móveis, bem como sua solução, avalia *middleware*s baseados em eventos e propõe um novo *middleware* baseado em eventos que pode ser utilizado como suporte para serviços de emergência em cidades inteligentes, apresentando a implementação de uma das abordagens propostas.

Palavras-chave: Cidades inteligentes; serviços de emergência; *middleware*; *middleware publish-subscribe;* roteamento de eventos.

ABSTRACT

Smart cities seek to improve the quality of life of citizens with the use of Information and Communication Technologies (ICTs). Efforts have been undertaken to build computer systems that address the big problems of cities. An important city service is the emergency service call (e.g., police and hospitals), which demands the moving of mobile units. The problem is to perform the activation of the mobile units which are the most appropriate ones to act in a particular situation by considering, amongst other factors, the time to reach the emergency site. A middleware can be used in this context to support the activation of mobile units by managing the notification of events related to emergencies. Several event-based middleware were analyzed regarding their suitability for the activation of the most suitable mobile units to act in emergencies in Smart Cities. However, they showed little flexibility with respect to event notification only to the most suitable mobile units. Thus, this paper presents and formalizes the problem of activating mobile units, as well as a corresponding solution, evaluates event-based middleware and proposes a more suitable event-based middleware to support emergency services in Smart Cities presenting the implementation of one of the proposed approaches.

Key-words: Smart cities; emergency services; middleware, publish-subscribe middleware; event routing.

1 INTRODUÇÃO

Atualmente, a população mundial que vive nas cidades passa dos 50% e espera-se para 2050 que esse percentual aumente para 70% (ALKANDARI; ALSHEKHLY, 2012). Com isso, novos desafios precisam ser enfrentados, especialmente em cidades maiores e mais populosas, como a gerência de resíduos que tende a aumentar, a escassez de recursos como, por exemplo, água e luz, a questão da poluição do ar e da sonora, preocupações com a saúde, gerência do tráfego urbano, educação, desemprego, entre outros. Assim, novas formas de ajudar a gerenciar as cidades com o auxílio das Tecnologias de Informação e Comunicação – TICs (ICT – *Information Communication Technologies*) se faz necessário. Dentro desse contexto, surgem as Cidades Inteligentes (*Smart Cities*), as quais buscam explorar as TICs na gestão dos assuntos públicos a fim de melhorar a qualidade de vida dos cidadãos (CENEDESE *et al.*, 2014).

A Internet das Coisas (*Internet of Things*), a qual também pode ser chamada de Internet dos Objetos Inteligentes, aplicada em um contexto urbano responde a muitos anseios governamentais em utilizar TICs para auxiliar na gestão de assuntos públicos, tornando possível, assim, automatizar muitos serviços de uma cidade. A chamada *Urban IoT* trata do uso de objetos inteligentes em um contexto urbano de modo a oferecer serviços que venham a auxiliar a administração pública das cidades, as empresas e os cidadãos (ZANELLA *et al.*, 2014). Nos últimos anos, considerável esforço de pesquisa e desenvolvimento tem sido realizado em diversas partes do mundo para se criar sistemas computacionais que auxiliem a enfrentar os grandes problemas das cidades causados, principalmente, pelo aumento populacional.

Um dos serviços de uma cidade que pode se beneficiar dessa tecnologia é o atendimento em situações de emergência, como incêndios, acidentes de trânsito, salvamento, entre outros. No entanto, ainda há poucos relatos na literatura sobre teorias e sistemas desenvolvidos para a implementação desse serviço. Trabalhos na área de *Emergency Response*, tais como Chitumalla *et al.* (2008), Patsakis *et al.* (2015), Milis *et al.* (2016), Abu-Elkheir, Hassanein e Oteafy (2016) e Dragoicea, Patrascu e Serea (2014), focam em apoiar a fase de execução do serviço, especialmente em caso de catástrofes, e dão pouca ênfase no apoio ao processo de tomada de decisão sobre o deslocamento de unidades móveis (veículos propriamente equipados e com profissionais qualificados) para o local de uma emergência, isto é, sobre o problema de acionamento das unidades móveis mais adequadas para atuar em uma situação em particular. Assim, este artigo discute esse problema, chegando a uma definição formal do problema e sua solução, avalia alternativas de projeto e de implementação; e, finalmente, propõe um novo *middleware* baseado em eventos para ser usado como plataforma para o acionamento de unidades móveis, apresentando a implementação de uma das abordagens propostas.

O restante deste artigo está organizado como segue. A seção 2 demonstra a complexidade do problema de prestar serviços de emergência

em cidades e mostra como pode ser dividido em problemas menores. A seção 3 formaliza o problema de acionamento de unidades móveis em situações de emergência, com a sua respectiva solução. A seção 4 descreve os requisitos de um sistema computacional para o acionamento de unidades móveis. A seção 5 discorre sobre a adequação dos *middleware*s baseados em eventos como plataforma para sistemas computacionais em cidades inteligentes, em especial para o acionamento de unidades móveis. A seção 6 apresenta a proposta de um novo *middleware* baseado em eventos como plataforma para um sistema computacional de acionamento de unidades móveis e a seção 7 apresenta o desenvolvimento de uma das abordagens propostas de implementação para o *middleware*. Considerações do trabalho realizado são descritas na seção 8.

2 SERVIÇOS DE EMERGÊNCIA EM CIDADES INTELIGENTES

As situações possíveis de emergência em uma cidade são inúmeras e podem variar significativamente quanto as necessidades de urgência, de intensidade e de diversidade de ação, como ilustram os seguintes exemplos:

- uma situação de assalto em um estabelecimento comercial requer mais *urgência* de ação por parte da polícia do que uma situação de vandalismo contra edificações.
- uma situação de incêndio de grandes proporções em um edifício requer uma ação de maior *intensidade*, isto é, com a participação de um número maior de bombeiros e equipamentos contra incêndio do que em uma situação de princípio de afogamento de uma pessoa em um rio.
- uma situação de acidente de trânsito, pode requerer uma ação conjunta entre polícia, bombeiros e hospitais, isto é, essa ação apresenta uma *diversidade* de três tipos de serviços, e não de apenas um tipo, como nos exemplos anteriores (somente polícia ou somente bombeiros).

Invariavelmente, uma situação de emergência requer o deslocamento de uma ou mais unidades móveis, isto é, veículos apropriadamente equipados e com pessoal técnico especializado.

Como critério de classificação pode-se posicionar uma situação de emergência em um espaço tridimensional com os seguintes eixos: *Serviço*, *Urgência* e *Intensidade*. No eixo *Serviço* estão representados os vários tipos de serviços, tais como polícia, bombeiros e hospitais. No eixo *Urgência*, o grau de urgência da situação, tais como baixa, média, alta e muito alta, sendo que cada tipo de serviço pode ter seu próprio grau de urgência. No eixo *Intensidade*, a quantidade de unidades móveis alocadas, sendo que pode ser definida uma intensidade diferente para cada tipo de serviço. Assim, o posicionamento de uma situação de emergência é definido por um conjunto de pontos, sendo cada um associado a um tipo

de serviço e representado no plano definido por urgência e intensidade. A Figura 1 mostra um exemplo de classificação de uma situação de emergência que requer uma diversidade de três tipos de serviços: polícia, bombeiros e hospitais. Para cada um desses serviços, há um grau específico de urgência e de intensidade.

Figura 1 – Exemplo de classificação de uma situação de emergência

Fonte: elaborada pelos autores

Na prática, observa-se que, independentemente do grau de urgência, espera-se que o provimento dos serviços ocorra sempre com a maior brevidade possível; qualquer atraso no provimento é justificável somente se houver outra situação com maior grau de urgência sendo atendida, de forma que as unidades moveis sejam, momentaneamente, insuficientes. O provimento de serviços em uma situação de emergência ocorre em, ao menos, duas fases, a saber:

1. *Fase de acionamento dos serviços*: Nesta fase, os correspondentes serviços devem ser comunicados da situação de maneira rápida e confiável. A decisão sobre quais serviços comunicar pode ocorrer em um único passo, quando todos os serviços necessários são comunicados por um único agente (pessoa ou sistema) que tenha detectado a situação de emergência, ou em uma sequência de passos, quando esse agente comunica somente parte dos serviços necessários, os quais, então, decidem comunicar outros, e assim sucessivamente, até que todos os serviços necessários sejam comunicados. Quando comunicado, um serviço determina a urgência e a intensidade da sua ação específica e, com isso, decide quantas e quais unidades moveis serão enviadas ao local da emergência.
2. *Fase de execução dos serviços*: Nesta fase, as unidades móveis deslocam-se até o local da emergência e, quando chegam, executam a ação propriamente dita. Nessa execução, as unidades móveis podem precisar de informações atualizadas sobre a situação de emergência para fazer ajustes nas suas ações. Ainda nesta fase, é possível que novos acionamentos de serviço ocorram como consequência de revisão das dimensões diversidade e intensidade

da ação. Ou seja, a classificação da situação de emergência pode mudar dinamicamente.

Portanto, o problema de provimento de serviços em situações de emergência é complexo e, para efeito de análise e pesquisa, pode ser dividido em problemas menores, porém fundamentais, da seguinte forma:

1. *Definição da estratégia inicial de ação*: Classificar a situação de emergência, determinando os tipos de serviços necessários, bem como a urgência e a intensidade de cada um.
2. *Acionamento de unidades móveis*: Selecionar e comunicar um conjunto de unidades móveis que satisfaça a necessidade de serviços (em diversidade e intensidade), de forma que os serviços sejam executados pelas unidades mais adequadas no momento, considerando as suas características particulares, incluindo disponibilidade de equipamentos e técnicos, bem como a sua distância relativa do local da emergência para que o serviço seja executado com a maior brevidade possível. Apesar de móvel, uma unidade poderia operar de forma que, em princípio, fosse acionada somente quando estivesse parada num local padrão (por exemplo, uma ambulância somente seria acionada quando estivesse parada no hospital), isto é, o fato de estar fora do local padrão (parada ou em movimento) significa que já está atuando em uma situação de emergência, logo está indisponível para atuar em outras situações. No entanto, é possível que, por uma definição de estratégia de ação, ocorra o cancelamento de um acionamento enquanto a unidade móvel já atua em uma ação e, então, seja acionada para atuar em outra situação de emergência, possivelmente mais urgente. Portanto, no caso geral, o problema de acionamento deve considerar que as unidades móveis possam se deslocar constantemente.
3. *Apoio na execução*: Supervisionar a execução da ação e orientar as unidades móveis na execução das suas ações específicas, fornecendo informações atualizadas sobre a situação de emergência.
4. *Avaliação permanente da estratégia de ação*: Avaliar permanentemente a eficácia da estratégia de ação em execução e, se necessário, readequar a estratégia através de alterações na sua classificação, isto é, nos tipos de serviços e correspondentes urgência e intensidade.

Uma cidade inteligente deve dispor de sistemas computacionais que auxiliem na solução de cada um desses problemas, preferencialmente, com o mínimo de intervenção humana, tanto para se obter maior eficácia no provimento dos serviços, como para o uso mais eficiente dos recursos, em especial, as unidades móveis.

3 PROBLEMA DE ACIONAMENTO DE UNIDADES MÓVEIS

O problema de acionamento de unidades móveis é formalizado como segue. Seja:

- $\Gamma = \{S_1, ..., S_n\}$ o conjunto dos $n \geq 1$ serviços disponíveis.
- $\Phi = \{\alpha_1, ..., \alpha_r\}$ o conjunto de $r \geq 1$ pontos geográficos de referência de uma cidade (cada ponto corresponde ao centro de uma região da cidade).
- ε uma situação de emergência.
- $\Psi \in \Phi$, o ponto geográfico de referência associado a ε.
- $\theta_i = \{v_{i,1}, ..., v_{i,m}\}$ o conjunto de $m \geq 1$ unidades móveis de $S_i \in \Gamma$.
- $\Delta_{i,\varepsilon} \in \mathbb{N}$ 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 $\alpha_k \in \Phi$.

Então, a solução do problema de acionamento de unidades móveis para atender ε consiste em determinar $\beta_{i,j,\varepsilon}, 1 \leq j \leq |\theta_i|$, para cada serviço $S_i \in \Gamma$ que faz parte da estratégia de ação, 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 problema pode ser formalizado da seguinte forma:

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

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

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

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

A **Figura 2** mostra um exemplo de situação de emergência. A área da cidade é mapeada em uma matriz 8 x 8 de células quadradas idênticas. Como consequência desse mapeamento é definido um sistema de coordenadas sobre a área da cidade. São definidos 16 pontos geográficos

de referência de forma que cada um seja a referência da areia definida pelas quatro células à sua volta. O ponto $p = (3, 3)$ é referência para a área definida pelo quadrado com vértices $(2, 2)$, $(2, 4)$, $(4, 4)$ e $(4, 2)$. A situação de emergência ocorre no ponto $P = (2.5, 3.5)$, logo o seu ponto de referência é $\Psi_\varepsilon = p = (3, 3)$. O serviço S_i demandado dispõe de quatro unidades móveis para atuação: $\theta_i = \{v_{i,1}, v_{i,2}, v_{i,3}, v_{i,4}\}$. Se a intensidade da situação de emergência for dois ($\Delta_{i,\varepsilon} = 2$), a resolução do problema selecionará as duas unidades móveis mais adequadas com relação ao ponto $\Psi_\varepsilon = (3, 3)$. Assim, se os níveis de adequação forem $\Omega_{i,1,\Psi\varepsilon} = 0.4$, $\Omega_{i,2,\Psi\varepsilon} = 0.8$, $\Omega_{i,3,\Psi\varepsilon} = 0$ e $\Omega_{i,4,\Psi\varepsilon} = 0.5$, as unidades $\Omega_{i,2,\Psi\varepsilon}$ e $\Omega_{i,4,\Psi\varepsilon}$ serão selecionadas para atuar na situação de emergência.

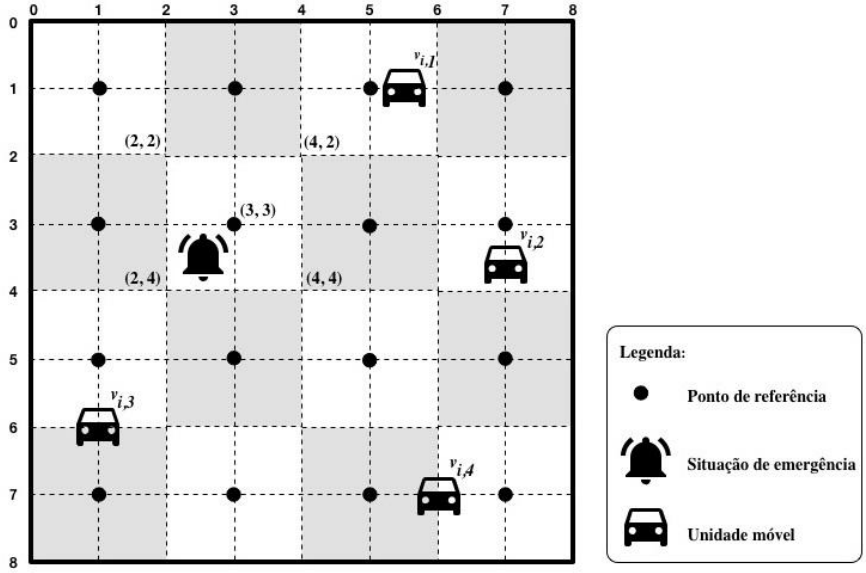

Figura 2 - Exemplo de uma situação de emergência

Fonte: elaborada pelos autores

Pela formulação do problema, a principal dificuldade na resolução de um serviço $S_i \in \Gamma$ está na determinação de $\Omega_{i,j,\Psi\varepsilon}$ para todo $v_{i,j} \in \theta_i$. Em outras palavras, a dificuldade está na determinação do nível de adequação de cada unidade móvel de um serviço para a sua atuação em uma situação de emergência detectada em certo ponto geográfico de referência. Tendo-se tal informação, a resolução do problema é relativamente trivial do ponto de vista matemático.

A dificuldade em determinar o nível de adequação de uma unidade móvel decorre de vários fatores. Primeiramente, o nível de adequação varia com a condição corrente da unidade móvel: quando está livre, o seu nível de adequação pode ser qualquer valor entre zero (0) e um (1), mas, enquanto estiver atuando em uma situação de emergência, o seu nível de adequação deve ser necessariamente zero (0), uma vez que está indisponível para atuar em novas situações de emergência. Segundo, há uma variação intrínseca desse nível de adequação à medida que uma unidade móvel se aproxima ou se afasta de um ponto geográfico de

referência (quanto mais próximo, maior é o nível de adequação). Finalmente, pode variar de acordo com características particulares de cada unidade móvel, tais como a sua capacidade padrão de prestar serviço (por exemplo, um veículo novo é mais adequado que um veículo antigo, pois desloca-se mais rapidamente), e a disponibilidade ou indisponibilidade momentânea de certos equipamentos ou técnicos.

Desta forma, como métrica para o cálculo do nível de adequação de cada unidade móvel serão consideradas a condição corrente da unidade móvel (disponível ou indisponível), a distância da mesma em relação a cada ponto de referência p e as condições de tráfego (trânsito leve, moderado e intenso). Assim, tem-se:

- $\tau_{v_{i,j}} = \begin{cases} 0, & \text{se } v_{i,j} \in \theta_i \text{ está atendendo uma situação de emergência} \\ 1, & \text{caso contrário} \end{cases}$
- $d_{v\,i,j,\alpha k} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$, a distância entre a unidade móvel $v_{i,j}$, localizada no ponto x_1, y_1 e o ponto de referência x_2, y_2.
- $\eta_{i,j,\alpha_k} = \{d_{v_{i,j,\alpha_1}}, \ldots d_{v_{i,j,\alpha_r}}\}$, o conjunto das distâncias de $v_{i,j}$ a cada ponto de referência $\alpha_k \in \Phi$.
- $Z_{\alpha_k} = \{Z_{\alpha_{i,1}}, \ldots Z_{\alpha_{i,r}}\}$, o conjunto das distâncias do ponto de referência α_i a todos os outros $\alpha_k \in \Phi$.
- $M_{\alpha_k} = \{Z_{\alpha_1}, \ldots Z_{\alpha_{i,k}}\}$, a distância máxima de cada conjunto de Z_{α_k} referente ao ponto $\alpha_k \in \Phi$.
- c_{i,j,α_k}: coeficiente referente as condições de trânsito da unidade móvel $v_{i,j}$ para cada ponto de referência $\alpha_k \in \Phi$, como por exemplo, se trânsito leve:

$$c_{i,j,\alpha_k} = \begin{cases} 1, & \text{se trânsito leve} \\ 0.5, & \text{se trânsito moderado} \\ 0.25, & \text{se trânsito intenso} \end{cases}$$

Logo, para o cálculo dos níveis de adequação precisa-se encontrar a máxima distância entre quaisquer dois pontos de referência da cidade. Assim, tem-se:

$$D_M = \max\{M_{\alpha_k}\}, \text{ para todo } i = \alpha_i, \ldots, \alpha_r$$

Assim, o nível de adequação de cada unidade móvel $v_{i,j}$ para cada ponto de referência é calculado da seguinte forma:

$$\Omega_{i,j,\alpha_k} = \begin{cases} \{(1 - (d_{v_{i,j,\alpha_k}} * c_{i,j,\alpha_k}) / D_M)\}, & \text{se } \tau_{v_{i,j}} \neq 0 \\ 0, & \text{caso contrário} \end{cases}$$

Para fins de exemplo de cálculo do nível de adequação, considere a Figura 3. A máxima distância entre todos os conjuntos de distâncias aos pontos de referência da cidade é $d_m = 8.49$. A situação de emergência ocorre no ponto $p = (2.5, 3.5)$. Assim, o ponto de referência é $\Psi_\varepsilon = p = (3, 3)$. Por conseguinte, as distâncias euclidiana de cada unidade móvel $v_{i,j}$ a este ponto de referência em relação ao serviço S_i será de: $d_l = \{d_1 = 3.61, d_2 = 4, d_3 = 4.47, d_4 = 2\}$. Logo, o nível de adequação para cada $\theta_i = \{v_{i,1}, v_{i,2}, v_{i,3}, v_{i,4}\}$ é de $\Omega_{i,1,\Psi\varepsilon} = 0.79$, $\Omega_{i,2,\Psi\varepsilon} = 0.76$, $\Omega_{i,3,\Psi\varepsilon} = 0.74$ e $\Omega_{i,4,\Psi\varepsilon} = 0.88$. Assim, se a intensidade da situação de emergência for dois ($\Delta_{i,\varepsilon} = 3$), as unidade móveis $v_{i,1}$, $v_{i,2}$ e $v_{i,4}$ seriam selecionadas para atuar na situação de emergência, considerando $\tau_{vi,j} \neq 0$ para todas as unidades móveis e condições de tráfego com trânsito moderado em toda a cidade. Quanto maior for o valor do nível de adequação de $v_{i,j}$, mais próxima ao ponto de referência se encontrará e apresentará melhores condições de trânsito para atendê-la.

4 SISTEMA COMPUTACIONAL PARA ACIONAMENTO DE UNIDADES MÓVEIS

A eficácia do sistema computacional que resolve o problema de acionamento de unidades móveis está diretamente relacionada com a atualização do estado referente ao nível de adequação das unidades móveis, pois é o fator determinante para a precisão do resultado obtido com a resolução do problema. Essa atualização do estado para um dado serviço pode ser feita segundo uma das seguintes abordagens:

- *Atualização reativa*: o estado é atualizado apenas no momento em que é necessário, isto é, na resolução do problema.
- *Atualização proativa*: o estado é permanentemente atualizado, de forma que esteja pronto sempre que for necessário resolver o problema.

A abordagem proativa permite que o processo de tomada de decisão, isto é, a resolução do problema e consequente determinação do subconjunto de unidades moveis selecionadas, ocorra de forma imediata, uma vez que todos os dados necessários estão disponíveis e atualizados, enquanto que a abordagem reativa impõe um atraso no processo (tempo para atualização do estado) que pode ser significativamente grande. Por outro lado, a abordagem proativa implica em consumo relativamente muito maior de recursos de sistema, incluindo processamento e comunicação.

Outra questão a ser considerada na arquitetura do sistema diz respeito à abordagem do protocolo de tomada de decisão, a qual pode ser uma das seguintes:

- *Decisão centralizada*: a tomada de decisão é realizada por um único processo que concentra todo o estado do sistema (níveis de adequação das unidades móveis para todos os pontos de referência).

- *Decisão descentralizada*: a tomada de decisão é realizada por um conjunto de processos cooperantes, sendo que cada processo mantém uma parte do estado do sistema.

Esses processos, denominados *brokers*, são distribuídos, tal que cada um gerencia uma região da cidade, sem que haja sobreposição. As funções de gerenciamento de um *broker* incluem:

- Monitorar as unidades móveis presentes na sua região para a atualização do correspondente estado (nível de adequação de cada unidade móvel para cada ponto de referência da cidade).
- Iniciar o processo de tomada de decisão para todas as situações de emergência na sua região; a tomada de decisão baseia-se em um protocolo de consenso entre os *brokers*.

A Figura 3 mostra um exemplo de arquitetura na qual a tomada de decisão é feita em cooperação por quatro *brokers*, sendo cada um responsável por uma região correspondente a 1/4 da área da cidade. Assim, a tomada de decisão para uma situação de emergência que demanda o serviço S_2 é iniciada pelo *broker* B_1, mas envolve todos os demais *brokers* em cujas regiões esteja alguma unidade móvel daquele serviço, isto é, *brokers* B_3 e B_4. Da mesma forma, a tomada de decisão para a situação de emergência que demanda o serviço S_1 é iniciada pelo *broker* B_4, mas envolve os *brokers* B_2 e B_3.

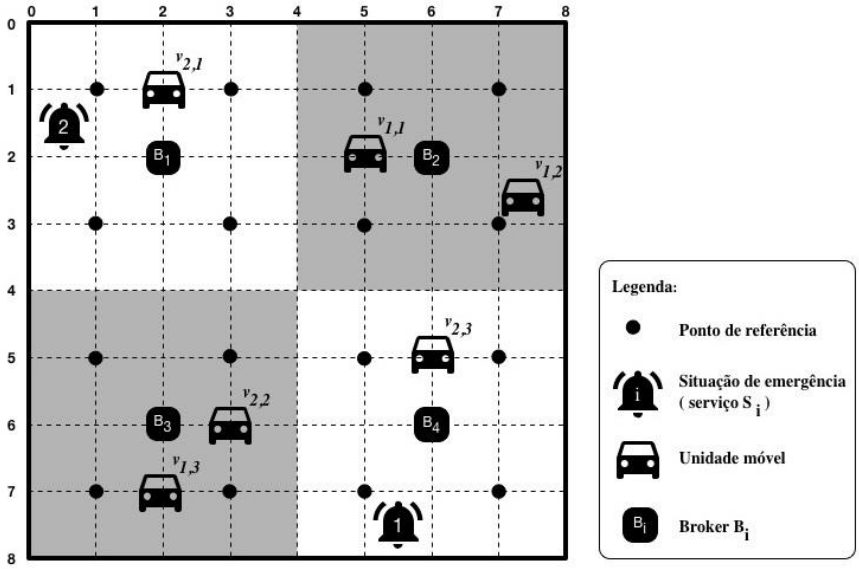

Figura 3 – Exemplo de arquitetura descentralizada para tomada de decisão

Fonte: elaborada pelos autores

A função da camada física do sistema computacional na abordagem centralizada para tomada de decisão é prover um servidor para hospedar o processo que toma as decisões e a comunicação (para fins de atualização de estado e acionamento das unidades) entre esse servidor e as unidades móveis. Na abordagem descentralizada, a função da camada

física depende da estratégia de alocação dos *brokers*, que pode ser baseada em cluster ou distribuída por regiões da cidade, como segue:

- *Alocação baseada em cluster*: A estratégia é alocar todos os *brokers* em um *cluster* implementado como um data center próprio ou através de um provedor de computação em nuvem. Assim, a função da camada física é semelhante à sua função na abordagem centralizada: prover um *cluster* e a comunicação entre esse e as unidades móveis (a comunicação entre os *brokers* é resolvida internamente no próprio *cluster*).
- *Alocação distribuída por regiões da cidade*: A estratégia é alocar cada *broker* em um servidor próprio que esteja fisicamente instalado na correspondente região gerenciada. Assim, a camada física deve prover um servidor para cada região, bem como a comunicação dentro de cada região (entre o respectivo servidor e as unidades móveis presentes na região) e entre os servidores.

Na decisão descentralizada, a atualização de estado devido a deslocamento é tratada especificamente pelo *broker* da região onde se encontra a unidade móvel após o deslocamento, enquanto que, na abordagem centralizada, todo e qualquer deslocamento é tratado globalmente pelo processo único do sistema. Ainda, na decisão descentralizada, a tomada de decisão envolve apenas um subconjunto dos *brokers* (da região da situação de emergência e das regiões com unidades móveis pertinentes), podendo haver muitas tomadas de decisão em paralelo, enquanto que a abordagem centralizada permite apenas uma tomada de decisão por vez. Ou seja, na decisão descentralizada, o processamento é realizado em paralelo por muitos processos, enquanto que, na decisão centralizada, todo o processamento é feito em único ponto, o que pode comprometer a escalabilidade do sistema. Além disso, a decisão descentralizada com alocação distribuída de *brokers* favorece a disponibilidade e a escalabilidade do sistema, pois, em princípio, não apresenta um ponto único falha e nenhum ponto crítico de sobrecarga (de processamento, armazenamento ou comunicação). A decisão descentralizada com alocação de *brokers* em *cluster* também favorece a disponibilidade do sistema, uma vez que *clusters*, em geral, são tolerantes a falhas, mas o favorecimento da escalabilidade fica restrito ao aspecto de processamento, uma vez que a comunicação é altamente concentrada em um único ponto da rede: o próprio *cluster*. Finalmente, a abordagem descentralizada, em especial com alocação distribuída de *brokers*, é naturalmente mais complexa, tanto no seu desenvolvimento quanto na sua operacionalização. O **Quadro 1** resume as principais vantagens e desvantagens esperadas das abordagens, considerando as combinações possíveis das suas alternativas.

Tomada de decisão	Atualização do estado	
	Reativa	*Proativa*
Centralizada	• Baixa complexidade • Poucas mensagens • Decisão demorada • Ponto único de falha • Baixa escalabilidade	• Baixa complexidade • Muitas mensagens • Decisão rápida • Ponto único de falha • Baixa escalabilidade
Descentralizada com alocação de brokers em cluster	• Média complexidade • Poucas mensagens • Decisão demorada • Alta disponibilidade • Média escalabilidade	• Média complexidade • Muitas mensagens • Decisão rápida • Alta disponibilidade • Média escalabilidade
Descentralizada, com alocação distribuída de brokers	▪ Alta complexidade ▪ Poucas mensagens ▪ Decisão demorada ▪ Alta disponibilidade ▪ Alta escalabilidade	▪ Alta complexidade ▪ Muitas mensagens ▪ Decisão rápida ▪ Alta disponibilidade ▪ Alta escalabilidade

Quadro 1 - Vantagens esperadas das abordagens centralizada e distribuída

Fonte: elaborada pelos autores

A escolha entre as alternativas de abordagens para que o sistema computacional apresente uma relação custo-benefício satisfatória deve se basear nas características particulares da cidade, nas restrições de recursos e na qualidade desejada para os serviços. Entre as variáveis determinantes para essa escolha, destacam-se as seguintes:

- *Quantidade de pontos de referência:* uma quantidade maior de pontos propicia maior precisão na resolução do problema, mas com maior custo de processamento e comunicação.
- *Quantidade de unidades móveis*: uma quantidade maior de unidades móveis permite obter soluções mais satisfatórias para o problema, mas aumenta o impacto em processamento e comunicação.
- *Frequência de situações de emergência*: quanto maior a frequência, maior é a chance de não ser possível resolver o problema por falta de unidades móveis.
- *Mobilidade das unidades*: no caso de abordagem proativa, quanto maior a mobilidade, maior é a necessidade de atualização de estado, aumentando o consumo de processamento e comunicação.
- *Frequência de atualização de estado*: no caso de abordagem proativa, quanto mais frequente for a atualização, mais precisa é a solução do problema, ao custo de mais processamento e comunicação.

Portanto, a configuração que tende a propiciar mais qualidade de serviço, logo de uso mais geral, é a de decisão descentralizada com alocação distribuída de *brokers*, especialmente no caso de Cidades Inteligentes, onde a disponibilidade e a escalabilidade do sistema são primordiais. O

grande desafio está em assegurar as vantagens esperadas e, ao mesmo tempo, mitigar suas desvantagens.

5 *MIDDLEWARE* BASEADO EM EVENTOS

Um *middleware* baseado em eventos é uma infraestrutura independente da aplicação que suporta a construção de sistemas baseados em eventos, no qual publicadores de eventos (*publishers*) notificam eventos para a infraestrutura (o próprio *middleware*) e assinantes de eventos (*subscribers*) subscrevem-se com o *middleware* para receber notificações de seu interesse (CARZANIGA; ROSENBLUM; WOLF, 2001). Em um *middleware* baseado em eventos, componentes, aplicações e todos os outros participantes interagem através de eventos (RAZZAQUE *et al.*, 2016).

No contexto de *Urban IoT*, um *middleware* poderia ser utilizado para gerenciar o grande número de eventos que os objetos inteligentes, presentes no ambiente urbano, geram espontaneamente. Em (RAZZAQUE *et al.*, 2016) uma análise de 61 *middleware*s quanto à sua adequação para IoT é apresentada. Os autores concluem que *middleware*s baseados em eventos são apropriados para aplicações móveis e reativas, pois permitem forte desacoplamento entre os componentes de uma aplicação distribuída, nesse caso, assinantes e publicadores de eventos. Mas, apontam, como desvantagens, o cumprimento insatisfatório de alguns requisitos, tais como interoperabilidade e segurança. Além disso, salientam que o paradigma de programação provido por esses *middleware*s não é suficientemente flexível, isto é, não se aplicaria para muitas aplicações relevantes.

A pouca flexibilidade do paradigma de programação dos *middleware*s baseados em eventos deve-se, basicamente, aos modelos de envio de eventos que implementam: o modelo ponto-a-ponto e o modelo *publish-subscribe*. No modelo ponto-a-ponto, produtores enviam mensagens para filas, enquanto recebedores podem estar registrados em algumas filas específicas para, assincronamente, recuperar as mensagens e, então, confirmá-las (MUSOLESI; MASCOLO; HAILES, 2006). Cada mensagem é entregue somente uma vez, para somente um consumidor, isto é, somente um consumidor pode obter a mensagem (MAGNONI, 2015). Essa forma de comunicação é também conhecida como um-para-um, na qual cada mensagem produzida é consumida apenas uma vez, sendo que isso é realizado pela fila de mensagens (EL RHEDDANE *et al.*, 2014). Esse modelo de comunicação permite múltiplos consumidores conectarem-se à fila, mas somente um dos consumidores pode consumir a mensagem. Assim, mensagens são sempre entregues e podem ser armazenadas na fila até que um consumidor esteja pronto para recuperá-la. No modelo *publish-subscribe*, assinantes (*subscribers*) ou consumidores de eventos subscrevem-se em eventos de interesse e publicadores (*publishers*) ou produtores de eventos produzem eventos, os quais são assincronamente enviados a todos os assinantes inscritos (EUGSTER; FELDER; KERMARREC, 2003). Ou seja, publicadores enviam mensagens contendo eventos para o

middleware e este os entrega para os assinantes, os quais expressaram seu desejo em recebê-los (OZALP; TEKIN, 2014). É função do *middleware* rotear as mensagens de subscrição até os publicadores e as mensagens com eventos até os assinantes. Para El Rheddane *et al.* (2014), o modelo *publish-subscribe* é usado como sinônimo do paradigma de comunicação um-para-muitos, no qual, para cada mensagem, seu recebimento é garantido para todos os assinantes subscritos através de tópicos. Uma principal vantagem desse tipo de *middleware* é o forte desacoplamento de espaço, tempo e sincronismo entre os publicadores e os assinantes dos eventos (EUGSTER; FELBER; KERMARREC, 2003). Desacoplamento de espaço porque publicadores não precisam saber quem são os assinantes e estes, por sua vez, não necessitam tomar conhecimento de quem são os publicadores. Desacoplamento de tempo porque as partes interagindo podem não participar ativamente na interação ao mesmo tempo. Além de que, no desacoplamento de sincronismo publicadores e consumidores podem realizar atividade simultânea enquanto eventos são produzidos e consumidos. Como consequência, produtores e consumidores são independentes e a arquitetura global do *middleware* se torna mais complexa (MAGNONI, 2015).

Essa pouca flexibilidade foi confirmada no estudo de *middleware*s *publish-subscribe* Hermes (PIETZUCH; BACON, 2002), Steam (MEIER; CAHILL, 2002), Siena (CARZANIGA; ROSENBLUM; WOLF, 2001), EMMA (MUSOLESI; MASCOLO; HAILES, 2006), Mires (SOUTO *et al.*, 2006), SensorBus (RIBEIRO *et al.*, 2008), RUNES (COSTA *et al.*, 2006), PSWare (LAI; CAO; ZHENG, 2009), PRISMA (SILVA *et al.*, 2014) e TinyDDS (BOONMA; SUZUKI, 2011). Buscou-se compreender a arquitetura, os modelos de assinatura, o método de recebimento de mensagens, a topologia dos servidores e, principalmente, como acontece o roteamento de eventos em cada trabalho. Os *middleware*s Hermes, EMMA, Siena, Mires, RUNES, PSWare e TinyDDS possuem arquitetura distribuída, o *middleware* Steam arquitetura parcialmente distribuída e os *middleware*s SensorBus e PRISMA arquitetura centralizada. Hermes utiliza para assinatura de eventos uma adaptação da assinatura baseada em tópico e conteúdo; STEAM utiliza assinatura baseada em tópico, conteúdo e localização; Siena, SensorBus, Runes e PSWare assinatura por conteúdo; enquanto todos os outros *middleware*s utilizam assinatura baseada em tópico. SensorBus e PRISMA possuem método de recebimento de mensagens *pull*, enquanto todos os outros *middleware*s *push*. Quanto à topologia dos servidores, Hermes e Mires possuem topologia *peer-to-peer* cíclica; Steam e Siena topologia *peer-to-peer* acíclica; Runes, PSWare e PRISMA topologia hierárquica. Em relação ao roteamento de eventos, Hermes utilizou roteamento de eventos seletivo do tipo Rendezvous, STEAM e Siena *flooding subscription*; entretanto, Siena considera o caminho mais curto para o envio dos eventos. EMMA utiliza o roteamento epidêmico, Mires *multi-hop* e TinyDDS *flooding event*. A maioria dos *middleware*s foram desenvolvidos para utilização em redes de sensores sem fio, exceto Hermes e Siena para redes com fio, Steam e EMMA para

redes ad-hoc. Com exceção de EMMA que implementa os modelos de mensagens ponto-a-ponto e *publish-subscribe*, todos os outros *middleware*s analisados implementam o modelo de mensagens *publish-subscribe*.

Na revisão de literatura realizada, observou-se que os *middleware*s analisados não permitem, por exemplo, criar uma aplicação na qual somente os *n* assinantes de maior prioridade ou mais adequados (segundo algum critério específico da aplicação) recebam um evento, como é requisito para um sistema computacional para acionamento de unidades móveis, conforme discutido na seção 3. Os *middleware*s não implementam esse modelo de notificação de eventos diferenciado. Além disso, as abordagens tradicionais de eleição por meio de algoritmos distribuídos possuem como desvantagem o fato da eleição acontecer a cada escolha de assinante. O algoritmo de eleição deveria ser executado tantas vezes quanto o número de unidades móveis necessárias para o atendimento a situação de emergência. Como consequência, eleger os assinantes mais adequados toda vez que uma situação de emergência acontecer poderia consumir demasiado tempo, *overhead* de processamento e mensagens para descobrir quem seria o líder em cada rodada. Ao final da execução de cada algoritmo, um ranking poderia ser realizado. Contudo, a atualização do ranking aconteceria de tempos em tempos e, considerando o contexto do trabalho, essa atualização deve ser de forma continua, ou seja, toda vez que as unidades móveis mudarem sua localização geográfica, estiverem atendendo a uma emergência ou acontecer algum fato como, por exemplo, a unidade móvel precisar de manutenção ou reposição de algum material. Ainda, como o *middleware* é uma camada de software que normalmente fica entre a camada de aplicação e as camadas mais baixas de uma pilha de protocolos, como as camadas de transporte e rede, ele poderia utilizar qualquer protocolo da camada de rede para transferência de dados, incluindo protocolos para IoT e protocolos desenvolvidos para Redes Veiculares. Diante do exposto, propõe-se um novo *middleware*, o qual é apresentado na seção 6.

6 O *MIDDLEWARE* E2BS

O *middleware* chamado *Event to Best Subscribers* (E2BS), aqui proposto, visa cobrir a lacuna identificada quanto a *middleware*s baseados em eventos que ofereçam suporte ao desenvolvimento de aplicações baseadas em eventos, conforme discutido na seção 5. O objetivo é utilizá-lo como suporte para a implementação de um sistema computacional para acionamento de serviços de emergência em Cidades Inteligentes, segundo a abordagem de tomada de decisão descentralizada com alocação distribuída de *brokers* (abordagem proativa), a qual foi descrita na seção 4. O *middleware* E2B tem como base, para o roteamento de eventos, o trabalho anterior dos autores Calsavara e Lima (2010) sobre Campos Magnéticos Virtuais, inicialmente desenvolvido para o roteamento de mensagens em rede *ad hoc*, de forma a garantir o balanceamento de carga entre réplicas de servidores.

O princípio do *middleware* B2BS consiste em criar um conjunto de redes overlay sobre a rede física que conecta *brokers* e unidades móveis, tal que haja uma rede overlay para cada combinação de ponto de referência e tipo de serviço em cada região da cidade. A função de uma rede overlay para um ponto de referência *P* e um serviço *S* em uma região *R* é fazer com que um evento referente a uma situação de emergência associada a um ponto de referência *P* pertencente a *R* seja notificado a um subconjunto de unidades móveis de *S*, selecionadas de acordo com a definição do problema descrito na seção 3. A entrega do evento a uma unidade móvel a aciona para atuar na situação de emergência. O conjunto de nós da respectiva rede overlay é composto pelo *broker* que gerencia a região da cidade que contém os vários pontos de referência (*broker* raiz), os *brokers* que possuem na sua região de gerenciamento alguma unidade móvel de *S* (*brokers* intermediários) e as próprias unidades móveis de *S*. A rede overlay gera uma árvore de disseminação de eventos em três níveis: *broker* raiz, *brokers* intermediários e unidades móveis. Logo, a árvore é criada por meio da rede overlay de roteamento (*overlay routing network*). Para Pietzuch e Bacon (2002), uma árvore de disseminação de eventos deve ser construída dinamicamente para que os eventos possam ser roteados dos publicadores para todos os assinantes interessados (PIETZUCH; BACON, 2002). No caso do *middleware* E2BS, os eventos serão roteados para os assinantes mais prioritários ou adequados, o que implica que nem todos os assinantes interessados receberão o evento pretendido.

A **Figura 4** mostra um exemplo de como as redes overlay, na qual cada rede é representada por uma árvore de disseminação de eventos, são configuradas para a cidade da **Figura 3**. Como a cidade representada pela **Figura 3** possui 16 pontos de referência e 2 serviços, devem ser configuradas 32 redes overlay, 8 redes overlay em cada região da cidade, ou seja, 4 redes para cada combinação de serviço e ponto de referência. Assim, as topologias de rede ilustradas na **Figura 4**, uma para o serviço S_1 e outra para o serviço S_2, aplicam-se aos quatro pontos de referência da região gerenciada pelo *broker* B_1, sendo o estado (nível de adequação das unidades móveis em relação ao ponto de referência) mantido por cada uma diferente.

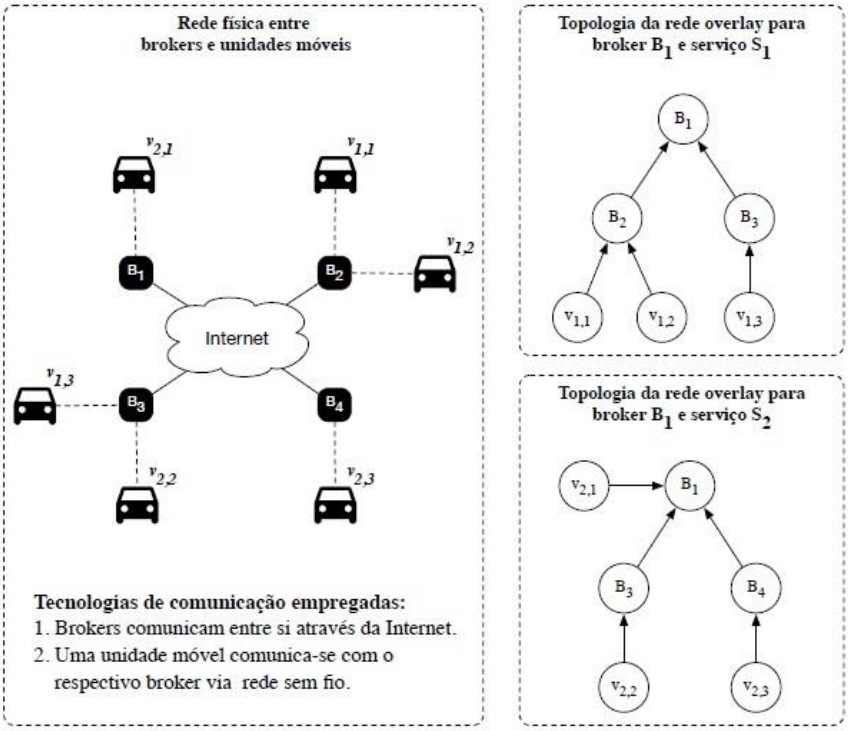

Figura 4 – Exemplo da topologia do *middleware* proposto

Fonte: elaborada pelos autores

Como exemplo de publicação de evento na **Figura 4** por B_1, com solicitação de duas unidades móveis do serviço S_1 para o ponto de referência P da rede overlay em questão, o mesmo será encaminhado as duas unidades móveis de S_1 mais adequadas a P, ou seja, duas entre $v_{1,1}$, $v_{1,2}$ e $v_{1,3}$, fazendo roteamento do evento através de B_2 e B_3. Os arcos representados em cada rede overlay da **Figura 4** estão no sentido da propagação de informação sobre adequação das unidades móveis, e no sentido inverso da disseminação dos eventos.

Na rede física, assume-se que todos os *brokers* estejam conectados à Internet, o que é válido para a abordagem de alocação de *brokers* distribuídos por regiões da cidade. Caso a alocação seja baseada em *cluster*, a comunicação entre *brokers* é normalmente feita por uma rede local de alta velocidade, mas também pode ser via Internet. A comunicação entre unidades móveis e *brokers* é, necessariamente, via rede sem fio, podendo ser completada com comunicação via rede cabeada, incluindo a própria Internet (não mostrado na figura). Para fins de limitação de escopo do trabalho, falhas em *links* de comunicação não serão tratadas neste trabalho.

6.1 SUBSCRIÇÕES PARA EVENTOS

As subscrições ou inscrições para eventos (isto é, situações de emergência) são importantes para a criação das redes overlay, sendo realizadas quando a unidade móvel entra no sistema e quando a mesma muda de região. É por meio das subscrições que as topologias das redes

são criadas e modificadas dinamicamente. Elas permitem que um evento seja atendido com a maior brevidade possível por pelo menos uma das unidades móveis disponíveis. Unidades móveis pertencentes a um serviço *S* se inscrevem para atender a uma situação de emergência no *broker* da região onde se encontram. Logo, inscrições para eventos possuem como conteúdo o serviço *S* que a unidade móvel está habilitada a tratar. Além disso, a unidade móvel encaminha juntamente com a subscrição sua disponibilidade corrente (sim para disponível ou não para indisponível) e localização geográfica. Logo, a operação *subscribe* é invocada com os seguintes argumentos:

- *s*: a identificação do serviço requisitado
- *p*: a posição da unidade móvel
- *disponibilidade*: a condição corrente da unidade móvel (sim ou não)

Com base na localização da unidade móvel e de sua disponibilidade, o *broker* da região na qual ocorre a subscrição, calcula os níveis de adequação para cada ponto de referência *P* da cidade para o serviço *S* da subscrição. A situação de trânsito de cada região da cidade (obtida em tempo real) também é utilizada para este cálculo. Com os níveis de adequação calculados para todos os pontos de referência *P* da cidade, o *broker* cria árvores de disseminação de eventos para o serviço *S* e ponto de referência *P* sob sua abrangência. Os níveis de adequação calculados para os outros pontos da cidade são encaminhados aos outros *brokers* de acordo com os pontos de referência sob a abrangência de cada um. Por consequência, caso aconteça uma situação de emergência em uma região *R* da cidade que não existir pelo menos uma unidade móvel subscrita para o serviço *S*, qualquer unidade móvel de *S*, subscrita em qualquer outro *broker* da cidade, pode atender a ocorrência mesmo não se encontrando naquela região. Nesse caso, o *broker* da região da situação de emergência notifica o *broker* em que a unidade móvel se subscreveu e o mesmo a aciona para o atendimento. O algoritmo do Quadro 2 apresenta o cálculo do nível de adequação em relação a um ponto de referência *P* da cidade para uma unidade móvel pertencente a um serviço *S*. A maior distância entre todos os pontos de referência da cidade é calculada e oferecida como entrada para o algoritmo, bem como as condições de trânsito.

Sendo uma árvore de disseminação de eventos construída em cada *broker* para cada par de serviço *S* e ponto de referência *P*, uma questão que deve ser considerada é quando uma unidade móvel se inscreve em um *broker* de uma região específica. Conforme mencionado, o *broker* recebendo a subscrição, calcula os níveis de adequação da unidade móvel em relação a todos os pontos de referência da cidade e os armazena. Os níveis de adequação calculados, exceto os que fazem parte de sua região, são enviados aos outros *brokers* de acordo com os pontos de referência da região de cada um. Isso porque este *broker* é um *broker* intermediário das árvores de disseminação de eventos das outras regiões.

Algoritmo 1: Algoritmo para o cálculo do nível de adequação em relação a um ponto de referência.

Entrada: localização da unidade móvel (x_2, y_2), disponibilidade da unidade móvel (disponibilidade), condições de trânsito (trânsito), maior distância entre os pontos de referência da cidade (maior), ponto de referência (x_1, y_1)
Saída: nível de adequação de uma unidade móvel para um ponto de referência específico da cidade

```
início
    distância ← sqrt( (sqr(x2 - x1) + sqr(y2 - y1) );
    se (disponibilidade = "sim") então
        se (trânsito = "intenso") então
            coeficiente ← 0.25;
        senão
            se (trânsito = "moderado") então
                coeficiente ← 0.50;
            senão
                se (trânsito = "leve") então
                    coeficiente ← 1.00;
                fim_se;
            fim_se;
        fim_se;
    senão
        coeficiente ← 0;
    fim_se;
    nível_adequação[ ponto] ← (1 - distância/maior * coeficiente);
fim.
```

Quadro 2 – Algoritmo para cálculo do nível de adequação

Fonte: elaborada pelos autores

Várias unidades móveis podem se inscrever em um *broker*. Assim, quando um *broker* envia os níveis de adequação para outro, ele considera o maior conjunto dos níveis de adequação de todas as unidades móveis em relação a cada ponto de referência *P* e serviço *S* da região de abrangência do *broker* que receberá os dados. Para o *broker* que recebe as subscrições, faz-se necessário manter o estado de todos os níveis de adequação das unidades móveis subscritas. Primeiramente, isso é preciso para que o *broker* realize o processo de escolha quando envia os níveis de adequação aos outros *brokers*. Segundo, porque se o *broker* for notificado, por outro *broker*¸ que é preciso o acionamento de alguma de suas unidades móveis, ficará fácil escolher a mais apropriada para o ponto de referência *P* da situação de emergência. O algoritmo do **Quadro 3** descreve este processo de escolha do maior conjunto dos níveis de adequação.

Algoritmo 2: Algoritmo para seleção do maior conjunto dos níveis de adequação das unidades móveis de serviço *S* para os pontos de referência de um *broker* de outra região da cidade.

Entrada: número de unidades móveis (N), níveis de adequação da unidade móvel em relação aos pontos de referência pertencentes a um *broker* (nível_adequação), número de pontos de referência pertencentes a *broker* de outra região (nr_pontos_referência)
Saída: conjunto T com os maiores níveis de adequação das unidades móveis subscritas em um *broker* para os pontos de referência específicos de outro *broker*

```
início
    para ponto ← 1 até nr_pontos_referência faça
        maior ← 0;
        para i ← 1 até N faça
            se (nível_adequação[ ponto] > maior) então
                maior ← nível_adequação[ ponto];
            fim_se;
        fim_para;
        T[ ponto] ← maior;
    fim_para;
fim.
```

Quadro 3 - Algoritmo para descobrir o maior conjunto dos níveis de adequação de um serviço S em relação aos pontos de referência de outro *broker*

Fonte: elaborada pelos autores

Ainda, as unidades móveis se movimentam pela cidade, atendem a chamados de emergência, podem apresentar problemas ficando inoperantes, entre outros. Assim, o nível de adequação de uma unidade móvel para um ponto de referência *P* e serviço *S* precisa ser atualizado. O *middleware* gerencia isso para que o sistema computacional de acionamento de unidades móveis funcione corretamente e a situação de emergência seja tratada com a brevidade exigida para a mesma.

6.2 PUBLICAÇÃO E NOTIFICAÇÃO DE EVENTOS

Uma publicação de evento corresponde a ocorrência de uma situação de emergência. A publicação leva em consideração vários fatores, tais como o serviço requisitado, o ponto de referência da situação de emergência e a intensidade da ação, em número de unidades móveis. Desta forma, o sistema de acionamento de unidades móveis deve invocar a operação *publish* oferecida pelo *broker* para cada serviço *S*, provendo os seguintes argumentos:

- *s*: a identificação do serviço requisitado
- *p*: o ponto de referência da situação de emergência
- *N*: o número de unidades móveis requisitadas

A operação *publish* é executada pelo *broker* que gerencia a região na qual a situação de emergência ocorre, ou seja, o *broker* raiz da árvore de disseminação de eventos. Esse *broker* também pode ser chamado de *broker* publicador (*publisher broker*). Como consequência, o evento é notificado (informado) para uma ou mais unidades móveis adequadas para seu atendimento. A operação seleciona a rede overlay do serviço *S* e

depois o ponto em que ocorreu a situação de emergência correspondendo ao par (*p, s*). Por consequente, o *broker* publicador envia *N* mensagens de eventos, sendo uma mensagem para cada unidade móvel. Antes de publicar cada evento, o *broker* publicador avalia o nível de adequação de cada *broker* intermediário, bem como das unidades móveis que se inscreveram para receber eventos diretamente com ele, e então encaminha o evento para aquele *broker* intermediário ou unidade móvel que tiver o maior nível de adequação em relação ao ponto de referência da situação de emergência. Esse processo acontece de forma recursiva até que todas as mensagens de eventos sejam enviadas as unidades móveis selecionadas ou até que o número de unidades móveis se esgote, o que ocorrer primeiro. Entretanto, se o nível de adequação for igual a zero em algum momento, para todos os *brokers* intermediários ou unidades móveis, o pedido não é cumprido. De outra forma, o *broker* publicador confirma cada unidade móvel acionada para o atendimento à situação de emergência ao sistema de acionamento de unidades móveis. Consequentemente, o número de unidades móveis que irão efetivamente atender a situação de emergência é contabilizado pelo sistema que recebe a confirmação.

O algoritmo do Quadro 4 ilustra o código da operação *publish* para o envio de eventos, a qual é executada pelo *broker* publicador. A rede overlay do serviço *S* para o ponto de referência *P* é localizada. Isso se faz necessário porque existe uma rede overlay para cada serviço *S* e ponto de referência *P* da região gerenciada pelo *broker* publicador. Depois disso, o maior nível de adequação em relação ao ponto de referência *P* em que ocorreu a situação de emergência é encontrado. A identificação do *broker* intermediário ou da unidade móvel a que pertence o maior nível de adequação é armazenada. Por consequente, a unidade móvel ou o *broker* intermediário são notificados sobre a situação de emergência.

Quando um *broker* (intermediário) recebe uma notificação de evento, ele precisa acionar alguma de suas unidades móveis, a mais apropriada, para o atendimento à situação de emergência. O serviço *S* solicitado precisa ser informado, pois a unidade móvel pode estar subscrita em mais de um serviço. Além disso, o ponto de referência *P* em que a situação de emergência ocorreu necessita ser comunicado. Desta forma, a unidade móvel se deslocará e realizará o atendimento ao chamado. O algoritmo do Quadro 5 ilustra o código de *notification*, o qual é executado por um *broker* intermediário.

Algoritmo 3: Algoritmo para notificação de eventos pelo *broker* publicador.

Entrada: número de unidades móveis necessárias para o atendimento à situação de emergência (N), serviço(*s*), ponto em que ocorreu a situação de emergência (*p),* nível de adequação de cada unidade móvel/*broker* intermediário em relação ao ponto de referência da situação de emergência (nível_adequação[ponto])
Saída: envio de evento para as unidades móveis e retorno do número de unidades móveis acionadas

```
início
    cont ←0;
    existe_change ←V;
    se ( (serviço = s) e (ponto = p) ) então
        enquanto ( (cont < N) e (existe_chance = V ) )
            maior ← -1;
            para i ← 1 ate nr_filhos_árvore faça
               se (disponibilidade = "sim") então
                    nível_adequação_p ← nível_adequação[ ponto];
                    se  (nível_adequacao_p > maior)  então
                        maior ← nível_adequação_p;
                        filho_selecionado ← i;
                    fim_se;
               fim_se;
            fim_para;
            escreva("O filho selecionado foi: ", filho_selecionado);
            se (maior != -1) então
                 // notifica o filho selecionado para o serviço s (o filho pode estar subscrito
para mais de
                // um servico) e o ponto que ele tem que atender a ocorrência
                ack = notification(filho_selecionado, s, p);
                se (ack = V) então
                    cont ← cont + 1;
                    atualiza_disponibilidade(filho_selecionado, "não");
                fim_se;
            senão
                existe_chance ← F;
            fim_se;
        fim_enquanto;
    fim_se;
    retorna(cont);
fim.
```

Quadro 4 - Algoritmo para notificação de eventos pelo *broker* publicador

Fonte: elaborada pelos autores

Uma questão que o *broker* trata, tanto o *broker* publicador quanto o *broker* intermediário, é quando a unidade móvel a ser acionada sai de sua região de abrangência. A unidade móvel quando muda de região envia uma mensagem específica para isso (mensagem *bye-bye*). O *broker*, como consequência, detecta sua saída. Logo, as árvores de disseminação de eventos são atualizadas. No caso dos algoritmos do **Quadro 4** e do **Quadro 5**, essa saída é demonstrada pelo retorno da chamada à operação *notification* (ack = F). A operação *notification*, antes de enviar a notificação sobre o evento, verifica se a unidade móvel se encontra na região. Em caso afirmativo, envia a mensagem e aguarda confirmação; caso contrá-

rio, outra unidade móvel é selecionada e notificada para atendimento aquela situação de emergência.

Algoritmo 4: Algoritmo para notificação de eventos à unidade móvel pertencente a um *broker* intermediário.

Entrada: número de unidades móveis subscritas no *broker* (N), serviço (*s*), ponto em que ocorreu a situação de emergência (*p),* nível de adequação de cada unidade móvel em relação ao ponto de referência da situação de emergência (nível_adequação[ponto])
Saída: notificação do evento para a unidade móvel que apresentar o maior nível de adequação, de todas as unidades móveis de um *broker* intermediário, para o ponto de referência em que a situação de emergência ocorreu

```
início
      existe_change ← V;
      se ( (serviço = s) e (ponto = p) ) então
          enquanto (existe_chance) faça
              maior ← -1;
              para i  ← 1 até N faça
                  se (disponibilidade = "sim") então
                      nível_adequação_p ← nível_adequação[ ponto];
                      se  (nível_adequação_p > maior)  então
                          maior ← nível_adequação_p;
                          unidade_móvel_selecionada ← i;
                      fim_se;
                  fim_se;
              fim_para;
              se (maior != -1) então
                   ack ← notification(unidade_móvel_selecionada, s, p);
                   se (ack = V) então
                       existe_chance ← V;
                       atualiza_disponibilidade(filho_selecionado, "não");
                   fim_se;
               senão
                  existe_chance ← F;
              fim_se;
          fim_enquanto;
   fim_se;
   retorna(existe_chance);
fim.
```

Quadro 5 - Algoritmo para notificação de eventos à unidade móvel pertencente a um *broker* intermediário

Fonte: elaborada pelos autores

6.3 GERÊNCIA DA MOBILIDADE DOS ASSINANTES

As unidades móveis podem desconectar-se de um *broker* e reconectar-se a outro *broker* depois de certo tempo devido a sua movimentação pela cidade. Essa mobilidade é gerenciada como parte do *middleware*. Como consequência, as árvores de disseminação de eventos devem ser atualizadas para que as situações de emergência sejam tratadas pelas unidades móveis mais apropriadas disponíveis no momento. Por sua vez, cada unidade móvel emite periodicamente mensagens de vínculo (*bind*) para confirmar que ainda se encontra conectada ao *broker* da região em que se subscreveu. Essa mensagem contém a localização exata da unida-

de móvel, bem como o serviço que a mesma se inscreveu e sua disponibilidade atual. Logo, a operação *bind* é invocada com os seguintes argumentos:

- *s*: a identificação do serviço requisitado
- *p*: a posição da unidade móvel
- *disponibilidade*: a condição corrente da unidade móvel (sim ou não)

A operação *bind* possui os mesmos argumentos da operação *subscribe*, isso porque os argumentos para a atualização dos níveis de adequação são os mesmos utilizados quando a unidade móvel se inscreve para um serviço *S*. Com as informações emitidas por *bind*, o *broker* recalcula os níveis de adequação da unidade móvel em relação aos pontos de referência da cidade e os atualiza em cada árvore de disseminação de eventos. O algoritmo do **Quadro 6** descreve o que acontece quando o *broker* recebe uma mensagem *bind* de uma unidade móvel e recalcula os níveis de adequação para um ponto de referência *P* e serviço *S*.

Algoritmo 5: Algoritmo para controle da mobilidade de uma unidade móvel.

Entrada: localização da unidade móvel (x_2, y_2), disponibilidade (disponibilidade), serviço (*s*), ponto de referência (*p*)
Saída: topologia da rede overlay atualizada para o serviço *s* e ponto de referência *p*

```
início
    se ( (serviço = s) e (ponto = p) ) então
        vinculada ← bind(s, x2, y2, disponibilidade);
        se (vinculada = V) então
            nível_adequação[ ponto] = calcula_nível_adequação(s, x2, y2, disponibilidade);
            atualiza_rede_overlay(nível_adequação[ ponto]);
        fim_se;
    fim_se;
fim.
```

Quadro 6 - Algoritmo para controle de mobilidade de unidade móvel

Fonte: elaborada pelos autores

Cabe salientar que quando os níveis de adequação se modificam, o *broker* avalia a necessidade do envio de um novo conjunto de níveis de adequação aos *brokers* de outras regiões da cidade. Isso é realizado para garantir a eficácia computacional do sistema de acionamento de unidades móveis.

7 DESENVOLVIMENTO DO *MIDDLEWARE* E2BS

Para o desenvolvimento do *middleware* E2BS, o método experimentação foi escolhido, mais especificamente a simulação. Na simulação, o pesquisador executa o produto com dados artificiais frequentemente em um modelo do ambiente real (ZELKOWITZAYB; WALLACE, 1997). A limitação da simulação está em quão bem o modelo corresponde ao ambiente real. Assim, quanto mais próximo o modelo estiver do ambiente real melhor. Desta forma, a simulação está sendo utilizada neste trabalho devido às

limitações na utilização de outros métodos controlados para alcançar o objetivo proposto.

Dentro deste contexto, o simulador selecionado para a implementação e validação do *middleware* E2BS foi o Sinalgo (*Simulator for Network Algorithms*)[1]. Sinalgo é um *framework* de simulação que permite a implementação de algoritmos de rede, testes e sua validação. Com o *framework*, um nó da rede pode enviar mensagens de forma *unicast* ou *broadcast* a outros nós, reagir a mensagens recebidas, definir temporizadores para agendamento de ações posteriores, entre outros. Sinalgo é livre e está publicado sob a licença BSD.

No simulador, alguns pacotes foram desenvolvidos, dentre os quais se destaca: (i) O pacote de mensagem: mensagens são enviadas entre os nós e precisam ser tratadas (por exemplo, mensagens de inscrição e de notificação de eventos); (ii) O pacote de implementação dos nós: os nós propriamente ditos como, por exemplo, os *brokers* e as unidades móveis que recebem, enviam e tratam as mensagens recebidas.

7.1 ATENDENTE

O atendente possui como papel acionar as unidades móveis para que atuem em situações de emergência. Ele aciona o *broker* raiz responsável pela região em que acontece a situação de emergência enviando uma mensagem *publish_activation*. Essa mensagem indica que o *broker* deve acionar uma quantidade *n* de unidades móveis de sua árvore de disseminação de eventos para atender a emergência. A cada unidade móvel acionada, o atendente recebe uma mensagem de confirmação *ack_publish_activation*. O atendente controla se o pedido é cumprido por meio dessas mensagens. Enquanto o pedido não é cumprido, o atendente envia novas mensagens de *publish_activation* de 2 em 2 minutos durante o período de 1 hora. Novas mensagens de *publish_activation* são geradas em intervalos aleatórios de tempo. Assim, se durante o reenvio de um pedido não atendido, uma nova mensagem é gerada, a mesma é colocada em uma fila para posterior envio.

7.2 *BROKERS* E UNIDADES MÓVEIS

Os *brokers* são processos distribuídos que desempenham funções importantes no *middleware* como, por exemplo, o cálculo dos níveis de adequação das unidades móveis, monitoramento das unidades móveis para atualização de seus estados e envio de mensagens de publicação de situações de emergência. Unidades móveis, por outro lado, possuem como papel atender as situações de emergência, deslocando-se até os locais das emergências.

As mensagens trocadas entre *broker* e unidade móvel são:

[1] http://disco.ethz.ch/projects/sinalgo

- *invite*: mensagem de convite enviada por um *broker* em *broadcast* a todas as unidades móveis que estão dentro da sua área de cobertura.
- *subscribe*: mensagem de inscrição para evento enviada por uma unidade móvel a um *broker*, em resposta a uma mensagem *invite.*
- *bind:* mensagem de vínculo emitida pela unidade móvel informando que ainda se encontra na região de abrangência do *broker*. Enviada periodicamente pela unidade móvel depois que a mensagem *subscribe* é recebida pela unidade móvel.
- *bye-bye*: mensagem que sinaliza que a unidade móvel está saindo da região de cobertura do *broker*.
- *notification*: mensagem enviada por um *broker* a uma unidade móvel para acioná-la.
- *ack_notification*: mensagem de confirmação de acionamento enviada por uma unidade móvel a um *broker*, em resposta a uma mensagem *notification.*
- *publish*: mensagem enviada por um *broker* raiz a um *broker* intermediário, indicando que o *broker* intermediário deve acionar uma de suas unidades móveis para atendimento.
- *ack_publish:* mensagem de confirmação de acionamento 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 presentes em sua região de cobertura.

7.3 SIMULAÇÃO

Para as simulações foram definidos alguns parâmetros, dentre eles:

1. Cada *round* (rodada) do simulador equivalendo a 1 segundo (também para cada movimento de unidade móvel).
2. Cada simulação correspondendo a 1 dia (86.400 *rounds*).
3. Área de cobertura dos *brokers* é de 100% da área urbana; entretanto, as unidades móveis podem eventualmente se deslocar para a área rural, ficando sem conectividade.
4. Tipo de serviço simulado como sendo o policial (unidades móveis se movimentam pela cidade, eventualmente parando por certo tempo em locais que requeiram atenção).
5. Acionamento do serviço realizado em intervalos de 30 minutos a 2 horas, escolhidos aleatoriamente.
6. Número de unidades móveis para atendimento a emergências com valores aleatórios entre 1 e 3.

7. Tempo de deslocamento dependente das condições de trânsito e da distância entre a unidade móvel e o ponto da situação de emergência.
8. O tempo de atendimento de cada unidade móvel variando de 20 a 40 minutos (aleatório para todos os casos).
9. O trânsito é considerado leve, moderado em intenso, dependente dos horários do dia, para o cálculo dos níveis de adequação.
10. Número de pontos de referência por *broker* definido em 4, conforme exemplo apresentado na seção 4.
11. Ponto da situação de emergência escolhido aleatoriamente na cidade.
12. Tempo de envio de mensagens de *invite* pelo *broker* às unidades móveis definido em intervalos de 1 minuto.
13. Tempo de envio de mensagens de *bind* pelas unidades móveis ao *broker* em intervalos de 5 minutos.

Desta forma, para demonstrar o funcionamento do *middleware* optou-se por realizar 4 simulações. Em cada simulação os *brokers* são acionados de forma circular. Os dados obtidos encontram-se na **Tabela 1**. Para os tempos considerou-se a representação hora:minuto:segundo.

Analisando a **Tabela 1**, nas quatro simulações realizadas, o número total de acionamentos ficou na média em 21. Na simulação 4, o número de acionamentos foi menor porque o tempo médio entre um acionamento e outro foi maior. Além disso, existe uma relação entre o número de acionamentos e as médias dos tempos de deslocamento e atendimento das unidades móveis. O número de unidades móveis tem influência direta sobre o percentual de acionamento concluídos com sucesso, ou seja, quanto mais unidades móveis, maior é esse valor. Quanto ao número de mensagens trocadas, percebe-se que quanto maior o número de unidades móveis e de acionamentos com sucesso, maior o número total de mensagens na simulação. Isso se deve as mensagens trocadas entre *brokers* e unidades móveis, em especial as mensagens: *bind, publish* e *ack_publish, notification* e *ack_notification.* O número de mensagens *ack_publish_activation*, as quais são trocadas entre os *brokers* e o atendente, também aumenta. Assim, quanto maior o número de unidades móveis, mais mensagens desse tipo são trocadas. Como consequência, menor é o número de tentativas de acionamento e melhor é o desempenho do sistema como um todo. Ainda, pode-se observar que o tempo médio entre um acionamento e outro é maior do que as médias dos tempos de deslocamento e atendimento das unidades móveis, influenciando diretamente a taxa de sucesso no atendimento a eventos. Se o tempo médio de deslocamento e atendimento fosse maior, certamente a taxa de sucesso seria menor.

Tabela 1 – **Dados da Simulação E2BS**

	Simulação 1	Simulação 2	Simulação 3	Simulação 4
Número de unidades móveis	1	2	3	4
Total de mensagens de acionamentos	21	21	22	20
Total de tentativas realizadas para conclusão das mensagens de acionamento	483	256	10	15
Percentual de acionamentos com sucesso (sem tentativas)	23%	33%	77%	90%
Percentual de acionamentos concluídos com tentativas	76%	67%	23%	10%
Total de mensagens	36.043	43.055	46.456	49.867
Tempo médio entre um acionamento e outro	1:07:38	0:41:46	01:07:45	01:11:40
Número de unidades móveis solicitadas	39	47	40	33
Número de unidades móveis acionadas	20	47	40	33
Percentual de sucesso no atendimento a eventos	51%	100%	100%	100%
Média de tempo de deslocamento	00:02:25	00:03:22	00:05:55	00:05:37
Média de tempo de atendimento	00:30:52	00:29:34	00:29:07	00:29:16
Média de tempo entre deslocamento e atendimento	00:33:16	00:32:49	00:35:18	00:34:53
Média do tempo de resposta dos pedidos ao atendente	0:0:17	0:0:18	0:0:25	0:0:17
Média do tempo de tomada de decisão pelos *brokers*	0:0:11	0:0:12	0:0:18	0:0:10

Fonte: elaborada pelos autores

Avaliando qualitativamente, constatou-se que a implementação do *middleware* E2BS foi complexa pelo fato de ser totalmente distribuída. O número de mensagens trocadas pelos *brokers*, unidades móveis e o atendente foi alto, pois o estado das unidades móveis foi atualizado constantemente. A decisão de escolha das unidades móveis foi rápida (levando em conta a média do tempo de tomada de decisão pelos *brokers*)

e o tempo de resposta ao atendente foi satisfatório. Além disso, o *middleware* se mostrou disponível e escalável. Disponível pelo fato de atender em 75% das simulações a 100% dos pedidos e escalável, pois apresentou comportamento satisfatório quando o número de unidades móveis e solicitações de acionamento aumentou.

8 CONSIDERAÇÕES FINAIS

A Internet das Coisas permite que objetos inteligentes se conectem por meio das TICs para tornar possível uma ampla faixa de serviços. Em um contexto urbano (*Urban IoT)*, o uso da grande quantidade e variedade de dados gerados por tais objetos pode ser utilizada para oferecer novos serviços aos cidadãos, companhias e administração pública. Dentre os serviços que poderiam ser oferecidos estão os cuidados com a saúde estrutural de construções, a gerência de resíduos, a monitoração da qualidade do ar e de ruído, o transporte inteligente, a gerência do consumo de energia, o estacionamento inteligente e o atendimento a situações de emergência.

Desta forma, a compreensão das questões envolvendo os serviços de emergência em cidades foi fundamental para o isolamento do problema de acionamento de unidades móveis. A formalização desse problema e correspondente solução permitiu especificar os requisitos para a construção do respectivo sistema computacional que dispense a intervenção humana, em consonância com o conceito de Cidades Inteligentes. A pesquisa sobre *middleware*s baseados em eventos para suportar a implementação de tal sistema mostraram a ausência de um *middleware* apropriado, motivando a proposta de um novo tipo de *middleware* baseado em eventos. A abordagem de tomada de decisão descentralizada com alocação distribuída de *brokers* com atualização do estado das unidades móveis de forma proativa foi implementada e sua validação descrita neste artigo. As abordagens de tomada de decisão centralizada, descentralizada com alocação de *brokers* em cluster e descentralizada com alocação distribuída de *brokers* de forma reativa serão implementadas em uma nova etapa do desenvolvimento do trabalho. Assim, as vantagens e desvantagens esperadas das abordagens serão avaliadas. Em especial, será possível verificar se o custo (em termos de mensagens e processamento), decorrente da atualização dinâmica das redes overlay, é compensado pela melhor adequação das unidades móveis acionadas em situações de emergência. A abordagem com líder, para a escolha das unidades móveis mais apropriadas, poderá ser implementada em trabalhos futuros. Com isso, a relação custo-benefício entre a abordagem com líder e as abordagens apresentadas poderá ser avaliada.

O *middleware* também poderá ser utilizado por outras aplicações como serviço de táxi e serviços móveis em geral, como por exemplo, de vendas (a rota da unidade móvel poderia ser definida em tempo real), de fiscalização (pela prefeitura e outras entidades), de entrega (alimentos, documentos, produtos em geral).

AGRADECIMENTOS

Este artigo é uma versão estendida e revisada do artigo intitulado "*Acionamento Inteligente de Unidades Móveis em Situações de Emergência em Cidades*", o qual foi publicado nos Anais do I Workshop de Computação Urbana - CoUrb 2017, evento satélite do XXXV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2017). Agradecimento especial à CAPES pelo custeio de taxa escolar.

REFERÊNCIAS

ABU-ELKHEIR, M.; HASSANEIN, H. S.; OTEAFY, S. M. A. Enhancing emergency response systems through leveraging crowdsensing and heterogeneous data analytics. *IEEE International Wireless Communications and Mobile Computing Conference*, p. 188–193, 2016.

ALKANDARI, A.; ALSHEKHLY, I. F. T. Smart cities : survey. *Journal of Advanced Computer Science and Technology Research*, v. 2, n. 2, p. 79–90, 2012.

BOONMA, P.; SUZUKI, J. TinyDDS: An interoperable and configurable publish/subscribe *middleware* for wireless sensor networks. *Wireless Technologies: Concepts, Methodologies, Tools and Applications*, p. 819–846, 2011.

CALSAVARA, A.; LIMA, L. A. P. Routing based on message attraction. *IEEE International Conference on Advanced Information Networking and Applications,* p. 189–194, 2010.

CARZANIGA, A.; ROSENBLUM, D. S.; WOLF, A. L. Design and evaluation of a wide-area event notification service. *ACM Transactions on Computer Systems (TOCS)*, v. 19, n. 3, p. 332–383, 2001.

CENEDESE, A. *et al.* Padova smart City: An urban Internet of Things experimentation. *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks ((WoWMoM))*, p. 1-6, 2014.

CHITUMALLA, P. K. *et al.* Emergency response applications. *IEEE Internet Computing*, v. 12, n. 1, p. 38–44, 2008.

COSTA, P. *et al.* The RUNES *middleware* for networked embedded systems and its application in a disaster management scenario. *IEEE International Conference on Pervasive Computing and Communications,* p. 69–78, 2006.

DRAGOICEA, M.; PATRASCU, M.; SEREA, G. A. Real time agent based simulation for smart city emergency protocols. *International Conference on System Theory, Control and Computing*, p. 187–192, 2014.

EUGSTER, P. T. H.; FELBER, P. A; KERMARREC, A. The many faces of publish / subscribe. Computing. *ACM Computing Surveys*, v. 35, n. 2, p. 114–131, 2003.

EL RHEDDANE, A. *et al.* Elastic message queues. *IEEE International*

Conference on Cloud Computing, p. 17–23, 2014.

LAI, S.; CAO, J.; ZHENG, Y. PSWare: A publish / subscribe *middleware* supporting composite event in wireless sensor network. *IEEE International Conference on Pervasive Computing and Communications (PerCom)*, p. 1-6, 2009.

MAGNONI, L. Modern messaging for distributed systems. *Journal of Physics: Conference Series*, v. 608, p. 1-8, 2015.

MEIER, R.; CAHILL, V. STEAM: Event-based *middleware* for wireless ad hoc networks. *IEEE International Conference on Distributed Computing Systems*, v. 2002–Janua, p. 639–644, 2002.

MILIS, G. *et al.* Integrated modelling of medical emergency response process for improved coordination and decision support. *Healthc Technol Lett*, v. 3, n. 3, p. 197–204, 2016.

MUSOLESI, M.; MASCOLO, C.; HAILES, S. EMMA: Epidemic messaging middleware for ad hoc networks. *Personal and Ubiquitous Computing*, v. 10, n. 1, p. 28–36, 2006.

OZALP, N.; TEKIN, Y. Content-Based Routing For Wireless Sensor Network Using Intelligent Agents. *IEEE International Symposium on Computational Intelligence and Informatics (CINTI)*¸p. 345–350, 2014.

PATSAKIS, C. *et al.* S-health as a driver towards better emergency response systems in urban environments. *IEEE International Symposium on Medical Measurements and Applications (MeMeA)*, p. 214–218, 2015.

PIETZUCH, P. R.; BACON, J. M. Hermes: A distributed event-based *middleware* architecture. *IEEE International Conference on Distributed Computing Systems*, p. 611–618, 2002.

RAZZAQUE, M. A. *et al.* Middleware for Internet of Things : A Survey. *IEEE Internet of Things Journal* v. 3, n. 1, p. 70–95, 2016.

RIBEIRO, A. D. R. L. *et al.* SensorBus – a policy-based middleware for wireless sensor networks. *IEEE Latin American Transactions*, v. 6, n. 7, p. 647–654, 2008.

SILVA, J. R. *et al.* PRISMA: A publish-subscribe and resource-oriented *middleware* for wireless sensor networks. *Proceedings of Advanced International Conference on Telecommunications (AICT)*, Paris, p. 87–97, July 2014.

SOUTO, E. *et al.* Mires: a publish/subscribe *middleware* for sensor networks. *Personal and Ubiquitous Computing*, v. 10, n. 1, p. 37–44, 2006.

ZANELLA, A *et al.* Internet of Things for smart cities. *IEEE Internet of Things Journal*, v. 1, n. 1, p. 22–32, 2014.

ZELKOWITZAYB, M. V.; WALLACE, D. Experimental validation in software engineering, *Information and Software Technology,* v. 39, p. 735–743, 1997.