# CSVM: Uma Plataforma para CrowdSensing Móvel Dirigida por Modelos em Tempo de Execução*

**Paulo César F. Melo, Fábio M. Costa, Ricardo C. A. da Rocha**

[1]Instituto de Informática – Universidade Federal de Goiás (UFG)
Caixa Postal 131 – 74.001-970 – Goiânia – GO – Brasil

{paulomelo,fmc,ricardo}@inf.ufg.br

***Abstract.*** *The complexity of applications in the mobile CrowdSensing domain is associated with factors such as interoperability among heterogeneous devices, identification and collection of data from these devices and adaptation of their use in dynamic environments. This paper introduces a platform based on models at runtime (M@RT) for the construction of mobile crowdsensing applications. The platform supports model-based creation and processing of queries directed to a distributed and dynamic set of sensor-capable devices. The paper presents the results of a performance evalutation that shows the impact of model processing in scenarios of mobile crowdsensing.*

***Resumo.*** *A complexidade de aplicações no domínio de CrowdSensing móvel está associada a fatores como interoperabilidade entre dispositivos heterogêneos, identificação e captação de dados desses dispositivos e adaptação do uso dessas aplicações em ambientes dinâmicos. Este artigo apresenta uma plataforma dirigida por modelos em tempo de execução (M@RT) para construção de aplicações de crowdsensing móvel por meio do uso de modelos para criação e processamento de consultas dirigidas a um conjunto distribuído e dinâmico de dispositivos com capacidade de sensoriamento. O artigo também apresenta os resultados de uma avaliação de desempenho realizada com o objetivo de mensurar o impacto do processamento de modelos em cenários de crowdsensing móvel.*

## 1. Introdução

*Crowdsensing* refere-se ao uso oportunista da vasta gama de sensores embarcados nos dispositivos de propósito geral atuais para medir e mapear fenômenos de interesse comum [Ganti et al. 2011] por meio do compartilhamento colaborativo dos sensores. O desenvolvimento de aplicações de *crowdsensing* móvel apresenta diversos desafios, como prover interoperabilidade entre diferentes dispositivos móveis, identificação dos dispositivos que podem ser fontes de dados, captação de dados provenientes desses dispositivos e adaptação dessas aplicações para o funcionamento adequado em ambientes dinâmicos [Ganti et al. 2011].

Este artigo apresenta uma plataforma dirigida por modelos em tempo de execução [Blair et al. 2009] (models@rt) denominada CSVM (*CrowdSensing Virtual Machine*) [Melo 2014], que permite criar e executar consultas de *crowdsensing* móvel por

*A pesquisa que deu origem a este trabalho foi desenvolvida com financiamento parcial da CAPES, CNPq (Processo: 473939/20126) e FAPEG

meio da especificação e interpretação de modelos descritos em uma linguagem de modelagem específica para esse domínio, denominada CSML (*CrowdSensing Modeling Language*). A abordagem dirigida por modelos em tempo de execução permite descrever dinamicamente o comportamento de *crowdsensing* de uma aplicação a partir da interpretação de modelos construídos em CSML.

Uma abordagem baseada em modelos permite diminuir a distância semântica entre o problema a ser resolvido e a plataforma utilizada, promovendo o uso de abstrações mais próximas do domínio do problema e, portanto, mais amenas ao uso por parte de usuários finais. Na abordagem de modelos em tempo de execução há uma representação causalmente conectada do ambiente e das consultas de *crowdsensing* móvel que estão sendo processadas. Desta forma, esta abordagem permite a adaptação a ambientes dinâmicos de *crowdsensing*, caracterizados pela entrada e saída de dispositivos do espaço monitorado e/ou pela modificação das consultas em andamento.

O restante deste artigo está organizado da seguinte forma. A Seção 2 descreve a abordagem para o uso de modelos em tempo de execução no domínio de *crowdsensing* móvel. A Seção 3 apresenta a linguagem CSML, enquanto a Seção 4 descreve a arquitetura da plataforma CSVM. A Seção 5 apresenta e discute os resultados da avaliação de desempenho do processamento de modelos pela plataforma. A Seção 6 discute os principais trabalhos relacionados e, por fim, a Seção 7 apresenta considerações finais e discute as principais contribuições.

## 2. Abordagem para *CrowdSensing Móvel* baseada em M@RT

Ambientes de *crowdsensing* móvel compreendem uma vasta quantidade de aplicações que precisam comunicar e trocar dados entre si. Sua essência, assim como os principais desafios, está na diversidade e quantidade de dispositivos associados, nas condições dinâmicas dos cenários e no processo de identificação de dispositivos para atenderem a uma determinada requisição [Ganti et al. 2011]. Para transpor essas dificuldades, propomos uma plataforma dirigida por modelos para o domínio de *crowdsensing* móvel, chamada CSVM (*CrowdSensing Virtual Machine*), que atende aos seguintes objetivos:

- permite que usuários ou desenvolvedores de aplicações de *crowdsensing* especifiquem seus requisitos de coleta de dados distribuídos usando uma linguagem de alto nível;
- identifica e seleciona automaticamente os dispositivos que podem prover os dados desejados;
- adapta-se a mudanças dinâmicas no ambiente de *crowdsensing*, selecionando o conjunto de dispositivos e sensores de forma a garantir a qualidade dos dados desejados;
- permite que o usuário altere uma consulta em execução, refletindo essa alteração no comportamento da plataforma; e
- abstrai, de forma independente de fabricante, a forma de acesso a diferentes dispositivos físicos e aos sensores neles embarcados.

A abordagem proposta neste trabalho não exige que usuários e desenvolvedores de aplicações sejam especialistas no domínio, uma vez que ela facilita a especificação das solicitações dos usuários por meio de modelos de alto nível, definidos na linguagem CSML.

## 3. A Linguagem de Modelagem CSML

CSML é uma linguagem de modelagem específica para o domínio de *crowdsensing* móvel e permite descrever e modelar em tempo de execução consultas e outros elementos pertinentes a esse domínio. Ela pode ser caracterizada como uma linguagem declarativa segundo a qual usuários e, principalmente, desenvolvedores devem se preocupar apenas em especificar as características das tarefas de *crowdsensing* a serem realizadas, isentando-os dos detalhes de como o executor da CSML interpretará e executará essas tarefas. Na implementação atual, a sintaxe da linguagem CSML é baseada em XML.

Dois aspectos comuns a aplicações de *crowdsensing* são modelados em CSML: *registro de dispositivo*, por meio do qual um dispositivo é incorporado em um ambiente de *crowdsensing*, e *especificação de consultas*, que permite ao usuário criar consultas sobre os dados de sensoriamento requeridos pela aplicação. Essas duas funcionalidades são especificadas em submodelos classificados em dois tipos de esquemas: *esquemas de controle* (CS) e *esquema de dados* (DS). Esquemas de controle são modelos que representam configurações lógicas de *crowdsensing* e podem ser subdivididos em esquema de controle do ambiente (ECS) e esquema de controle da consulta (QCS). As construções usadas para especificar os esquemas são definidas no metamodelo de CSML segundo a arquitetura de metamodelagem da OMG, definida na *Meta-Object Facility* (MOF) [Omg 2008].

| Componente | Descrição | Exemplo |
|---|---|---|
| Dispositivo | modelo do dispositivo | um certo dispositivo Android |
| Sensor | sensores associados aos dispositivos | localização, temperatura |
| Operação | operações de coleta de dados dos sensores | cálculo de média de temperatura |
| Combinação | formas de combinação dos dados | agregação de temperaturas com localização |
| Notificação | formas de notificação dos dados | baseada em eventos ou periódica |

**Tabela 1. Componentes de um modelo ECS**

O **Esquema de Controle do Ambiente** (ECS) descreve o ambiente de *crowdsensing* por meio dos componentes descritos na Tabela 1. O **Esquema de Controle da Consulta** é um modelo em tempo de execução que permite especificar uma ou mais consultas, informando os tipos de sensor desejados e relacionando a cada um a quantidade, a localização, a operação (que será realizada sobre o conjunto de dados coletados pelo sensor – por exemplo, realizar o cálculo da média da umidade do ar em um determinado ambiente), o tipo de notificação (refere-se à forma de coleta e notificação dos dados). Desta forma, por exemplo, um QCS poderia descrever uma consulta para medir a umidade do ar em um ambiente fechado por meio de dados coletados de 10 dispositivos localizados no ambiente.

Um **Esquema de Dados** é construído de forma automática como resultado do processamento de uma consulta previamente especificada por uma aplicação (ou usuário). Assim, o esquema de dados representa um *form* a ser preenchido, contendo atributos como valor, tipo de dado, unidade de medida, tipo de sensor, precisão e horário da coleta do dado. Um esquema de dados descreve também dados referentes à requisição, como tipo da requisição e da notificação e o identificador (*id*) da consulta que o gerou. Exemplos de esquemas de dados e de controle podem ser encontrados em [Melo 2014], os quais, por falta de espaço, não foram incluídos neste artigo.

Um modelo descrito em CSML pode ser modificado dinamicamente, em tempo

de execução, por exemplo, para alterar a estrutura de uma consulta em execução ou para alterar o registro de dispositivos no ambiente. Devido a essa característica e a outros aspectos, como adaptação e reflexão, modelos CSML podem ser considerados como modelos de tempo de execução [Wang et al. 2008].

## 4. Arquitetura da Plataforma CSVM

A arquitetura de plataformas de *crowdsensing* móvel geralmente compreende dois tipos de componentes, um representando os dispositivos de coleta e propagação de dados sensoriados e outro correspondente ao serviço que analisa e processa os dados sensoriados [Ganti et al. 2011]. Seguindo essa abordagem geral, este trabalho propõe uma arquitetura estruturada na forma de um componente central (CSVMProvider), que compreende uma única instância e representa o provedor do serviço, e um componente distribuído (CSVM4Dev), que compreende várias instâncias localizadas em diferentes dispositivos móveis.

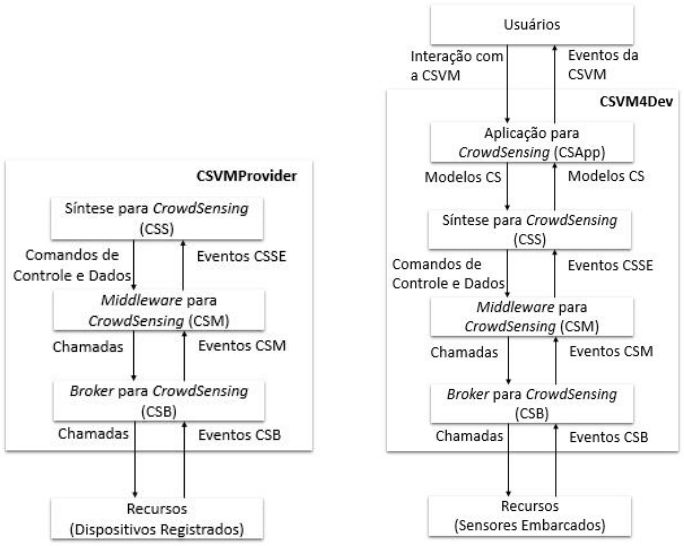
### 4.1. CSVMProvider e CSVM4Dev

O componente CSVMProvider é responsável por interpretar modelos (esquemas), construídos pelo usuário em CSML, a fim de determinar as ações a serem executadas, em termos das funcionalidades de consulta e de gerência dos dispositivos participantes do ambiente de *crowdsensing*. A CSVMProvider apresenta uma arquitetura em três camadas, ilustradas na Figura 1(a).

A **camada de síntese** (CSS - *CrowdSensing Synthesis*) é responsável por interpretar modelos recebidos em X-CSML e sintetizá-los, resultando em comandos a serem executados pelas camadas subjacentes. Esta camada também é responsável por manter um modelo em tempo de execução (m@rt) que reflete o estado atual das consulta em processamento. Durante a execução de uma consulta, o usuário pode fazer alterações em seu modelo, as quais são processadas pela CSS por meio do cálculo da diferença entre o novo modelo do usuário e o m@rt, resultando em um novo m@rt.

A **camada de middleware** (CSM - *CrowdSensing Middleware*) é responsável pela seleção lógica de dispositivos com base em informações extraídas do m@rt, notadamente informações sobre as capacidades dos dispositivos e sua localização. A CSM comunica diretamente com as demais camadas, processando comandos gerados pela camada de síntese e gerando eventos para ela, assim como comandos para serem executados na camada de *broker*. Por fim, a **camada de broker** (CSB - *CrowdSensing Broker*) permite à CSVMProvider interagir com os dispositivos físicos que oferecem capacidade de sensoriamento. Dentre suas principais responsabilidades estão: abstrair a diversidade de dispositivos presentes em ambientes de *crowdsensing*; acessar os recursos físicos (dispositivos e seus sensores) apropriados para satisfazer uma consulta; e monitorar os dispositivos registrados no ambiente.

O componente CSVM4Dev tem como finalidade permitir acesso aos dispositivos e aos sensores neles embarcados, além de possibilitar que o usuário especifique consultas. Ele apresenta características similares à CSVMProvider, como sua arquitetura em camadas (Figura 1(b)). A arquitetura da CSVM4Dev foi projetada de forma que ela possa ser implementada para diferentes tecnologias de dispositivos móveis. A CSVM4Dev é subdividida em quatro camadas, diferenciando-se da CSVMProvider por apresentar uma camada adicional, que possibilita a interação do usuário com a plataforma.




(a) CSVMProvider (b) CSVM4Dev

**Figura 1. Arquitetura em Camadas da Solução**

A **camada de aplicação** (CSApp - *CrowdSensing Application layer*) provê uma interface para utilização da plataforma, a qual permite ao usuário especificar e manter modelos de *crowdsensing* em CSML. A CSS é responsável por validar e converter os modelos criados pelo usuário (na camada CSApp) em modelos textuais baseadas em XML (X-CSML) antes de enviá-los à CSVMProvider. A CSM se restringe ao tratamento de requisitos não-funcionais (segurança e privacidade), em colaboração com a camada equivalente na CSVMProvider. A CSB, por sua vez, é responsável pelo gerenciamento dos recursos dos dispositivos (sensores) e pela comunicação com a CSVMProvider.

### 4.2. Registro de Dispositivos

O registro do dispositivo antecede as demais funcionalidades e é apresentado na Figura 2 por meio da interação entre os componentes CSVM4Dev e CSVMProvider.

A etapa inicial compreende o registro do dispositivo no serviço de comunicação para habilitá-lo a receber consultas geradas pela CSVMProvider (passos $r1$ e $r2$ na Figura 2). Em seguida, o usuário deve especificar as configurações do dispositivo (informações do dispositivo, como proprietário, marca, modelo e sensores que deseja compartilhar) preparando-o para registro. A CSVM4Dev procede então ao envio da requisição de registro (passo $r3$). Esse envio ocorre na forma de modelo, mais especificamente um esquema de controle, por meio do qual o dispositivo publica suas capacidades. A CSVMProvider valida o modelo (verificando se suas informações estão completas) e, em seguida, registra o dispositivo no repositório (passo $r4$), efetivamente acrescentando sua informação ao esquema de controle do ambiente.

A conclusão do registro do dispositivo no ambiente de *crowdsensing* é notificada ao dispositivo (passo $r5$). Uma vez registrado, o dispositivo está habilitado a especificar consultas utilizando a plataforma, bem como a participar de forma colaborativa na construção de respostas a consultas feitas por usuários de outros dispositivos registrados.

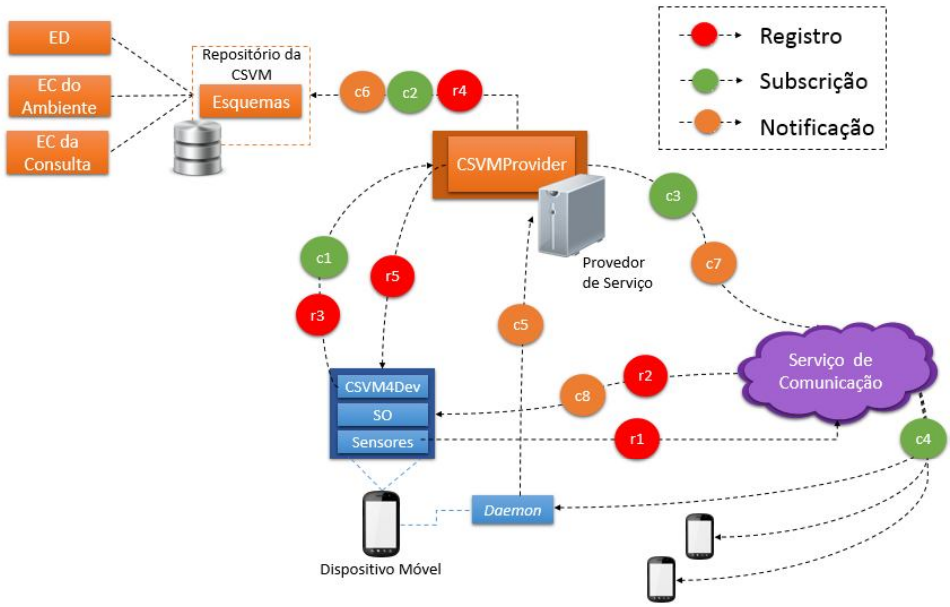



**Figura 2. Processo de Registro de Dispositivos e Processamento de Consultas.**

### 4.3. Consulta a Dados de Sensores

O principal serviço do domínio de *crowdsensing* é a especificação e processamento de consultas. Uma consulta refere-se a um conjunto de sensores e suas especificações (tais como tipo, localização e quantidade). O processamento de consultas é retratado também na Figura 2, enfatizando as etapas referentes à subscrição de consultas e à notificação dos dados requisitados, que serão detalhadas a seguir.

A **subscrição** refere-se às etapas iniciais do processamento de consultas especificadas pelos usuários, que expõe seu interesse no sensoriamento de uma determinada localidade, podendo especificar mais de um tipo de sensor por consulta e combiná-los a fim de derivar informações compostas.

Após especificada a consulta, na forma de um esquema de controle de consulta, ela é subscrita pela CSVM4Dev (Figura 2, passo $c1$) junto à CSVMProvider, onde será interpretada, gerando uma sequência de transações realizadas na base de dados (passo $c2$). Essas transações envolvem a seleção de dispositivos que satisfaçam os critérios de seleção da consulta (por exemplo, a localização e os tipos de sensores requisitados). Ao processar uma consulta e selecionar os dispositivos que dela participarão, a CSVMProvider gera uma instância do esquema de controle da consulta, que constitui o modelo em tempo de execução da consulta e compreende os dispositivos selecionados e as demais especificações da consulta. A próxima etapa envolve o envio das requisições aos dispositivos selecionados, que ocorre por meio da geração e envio de esquema de dados a cada um deles (passos $c3$ e $c4$). Cada dispositivo processa o esquema de dados recebido e, a partir dos dados sensoriados, envia notificações ao provedor de serviço. Essa etapa é descrita a seguir.

A **notificação** refere-se à etapa final do processamento de uma consulta. A CSVM4Dev processa o esquema de dados e aciona os sensores nele especificados para

capturar as informações desejadas, encapsulando-as em uma instância do esquema de dados, que é enviada ao provedor de serviço (passo $c5$). Na medida em que os dispositivos recrutados processam as requisições e geram as notificações, a CSVMProvider se encarrega de aplicar as operações relacionadas a cada tipo de dado monitorado, como por exemplo, calcular a média. Ao fim desse processamento, a CSVMProvider envia uma mensagem contendo a instância do esquema de dados (resposta à consulta) para o dispositivo que gerou a consulta (passos $c7$ e $c8$). Por fim, o resultado da consulta é apresentado pela CSVM4Dev ao usuário ou à aplicação que fez a consulta.

### 4.4. Exemplo de Cenário de Uso da CSVM

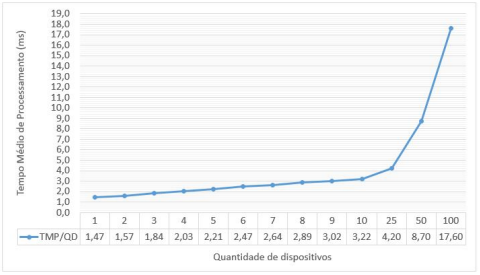
Para ilustrar o uso da CSVM adotamos o cenário de uma casa de shows, visando ao controle da temperatura e à detecção de incêndio, que podem ser realizados por meio de aplicações de *crowdsensing* utilizando a plataforma CSVM. Para modelar consultas de *crowdsensing* nesse cenário, o usuário deve registrar seu dispositivo na plataforma. Uma vez efetuado o registro, o gerente da casa de shows poderia especificar a seguinte consulta: *"Recuperar a temperatura de 20 sensores na casa de shows 'X'"*. De modo geral, a plataforma processará a consulta, selecionando, dentre os dispositivos das pessoas presentes na casa de shows (e que tenham sido registrados na plataforma) aqueles que satisfaçam a consulta, procedendo então ao monitoramento dos dados requisitados. Qualquer alteração no ambiente (por exemplo, entrada e saída de dispositivos) é refletida na plataforma, que realiza uma adaptação da consulta em execução. Ao fim do processamento, os dados são retornados para o usuário que originou a consulta.

### 4.5. Implementação da CSVM

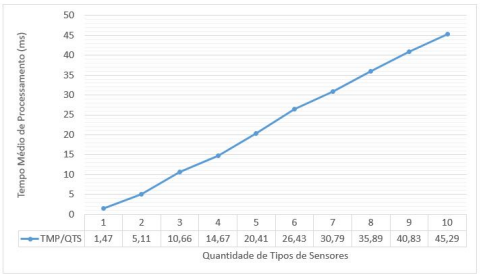
Para demonstrar a viabilidade da arquitetura proposta, foi desenvolvido um protótipo da plataforma CSVM e dos componentes CSVM4Dev e CSVMProvider. A CSVM4Dev foi implementada na plataforma Android. Para comunicação com a CSVMProvider foi implementado um cliente *RESTful* e serviços GCM (*Google Cloud Messaging*. A persistência de dados no dispositivo móvel utiliza a biblioteca *SQLite*. A CSVMProvider foi escrita em linguagem Java e sua implementação envolve a construção de um *web service* que implementa serviços *RESTful* (usando a API JAX-RS e JSON para a formatação dos dados). A CSVMProvider é executada sobre um contêiner de *servlets Apache Tomcat*. *Hibernate* é utilizado para o gerenciamento e mapeamento entre objetos Java e um banco de dados relacional *MySQL*. O projeto completo da plataforma CSVM contém 85 classes Java que implementam os serviços básicos e os processos descritos nesta seção.

## 5. Avaliação de Desempenho

A avaliação de desempenho da CSVM considerou duas tarefas relevantes: (i) o processamento de um esquema de controle de consulta e (ii) o processamento e agregação de um esquema de dados. Para essa avaliação, foi construído um experimento com um repositório contendo 100 dispositivos simulados, com 15 tipos de sensores cada. O cliente consumidor dos dados dos sensores foi executado em um *smartphone* Android. Os dados coletados correspondem à média obtida após 10 execuções, o que foi considerado suficiente para minimizar os erros no experimento. As figuras 3 e 4 exibem os resultados obtidos.

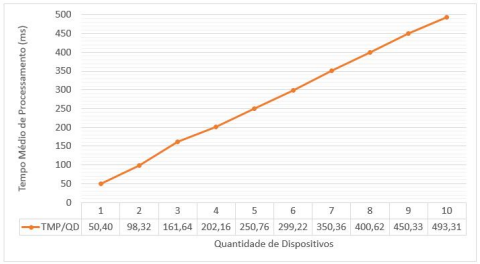



(a) Em função do número de dispositivos

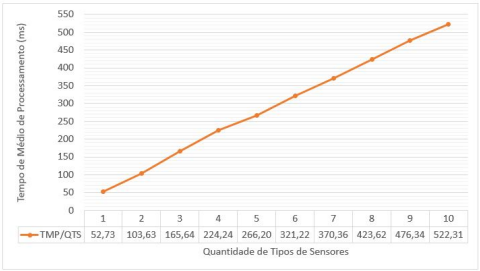



(b) Em função do número de tipos de sensores

**Figura 3. Tempo de Processamento de Consultas**




(a) Em função do número de dispositivos




(b) Em função do número de tipos de sensores

**Figura 4. Tempo de Processamento e Agregação de Dados**

Os gráficos das figuras 3(a) e 3(b) apresentam, respectivamente, o tempo médio de processamento de uma consulta em relação ao número de dispositivos (considerando um único tipo de sensor) e em relação ao número de tipos de sensores especificados em uma consulta. Em ambos os experimentos, o tempo de processamento aumenta à medida em que as variáveis testadas (quantidade de dispositivos e tipos de sensores) também aumentam. No primeiro experimento, extrapolamos a quantidade de dispositivos (25, 50 e 100) para avaliar a escalabilidade da plataforma.

Os gráficos das figuras 4(a) e 4(b) apresentam o tempo médio de processamento dos dados notificados pelos dispositivos em relação ao número de dispositivos e de tipos de sensores. Em relação aos experimentos anteriores, percebemos um aumento no tempo de processamento, que decorre do fato de que a quantidade de operações realizada pela CSVMProvider durante essas etapas é maior que a quantidade de operações realizadas no processamento de consultas.

Os resultados obtidos permitem demonstrar o grau de escalabilidade da plataforma CSVM em sua implementação atual. Apesar do aumento consistente no tempo de processamento tanto de consultas quanto de respostas (notificações), podemos observar que isto não inviabiliza o uso da CSVM em aplicações que toleram tempos de resposta da ordem de alguns poucos segundos, que são o caso típico de aplicações de *crowdsensing* móvel. A gráfico da Figura 3(a) apresentou uma tendência exponencial no aumento do atraso, podendo produzir atrasos mais significativos quando há mais do que 100 dispositivos envolvidos no ambiente de crowdsensing. Essa tendência foi causada pela sobrecarga na interpretação e desmembramento do esquema de consultas (no formato JSON) e, por este

motivo, a CSVM é melhor indicada em cenários de crowdsensing de média escala, como no experimento apresentado em [Mathur et al. 2010].

## 6. Trabalhos Relacionados

Nesta seção discutimos três trabalhos que adotam abordagens baseadas em modelos para implementação de aplicações de *crowdsensing*. Moo-Ryong *et al.* [Ra et al. 2012] descreve a plataforma Medusa, que propõe uma linguagem de alto nível baseada em XML, denominada MedScript. Chan *et al.* [Chan et al. 2013] descreve a plataforma de sistemas ciberfísicos Vita, que disponibiliza uma interface gráfica para auxiliar o usuário na especificação de consultas. Por fim, Hachen *et al.* [Hachem et al. 2014] apresenta o *middleware* MobIoT, que possibilita a especificação de consultas por meio de uma linguagem de consultas baseada em SQL e TinyDB.

Com relação à detecção e seleção de dispositivos, Medusa propõe o uso do serviço *Amazon Mechanical Turk*, enquanto a plataforma Vita propõe a seleção com base nos recursos computacionais disponíveis nos dispositivos. Por sua vez, no middleware MobIoT, a detecção e seleção é realizada por meio de padrões de mobilidade dos dispositivos que satisfazem uma cobertura de monitoramento, tendo a escalabilidade como principal preocupação. A CSVM permite a utilização de diferentes algoritmos para essa finalidade, sendo que atualmente utilizamos um algoritmo simples que seleciona os primeiros dispositivos encontrados que satisfaçam as exigências de uma consulta.

Com relação à técnica de interpretação de modelos em tempo de execução, dois trabalhos se destacam: a CVM (*Communication Virtual Machine*) [Deng et al. 2008], que é uma plataforma para construção e execução dirigida por modelos de aplicações que são montadas a partir de serviços de comunicação; e a MGridVM (*Microgrid Virtual Machine*) [Allison et al. 2011], uma solução dirigida por modelos inspirada na CVM para o gerenciamento de redes elétricas inteligentes de área local *microgrids*. A CVM realiza o processamento de modelos descritos em uma DSML denominada *Communication Modeling Language* (CML) [Clarke et al. 2006], enquanto a MGridVM apresenta uma DSML para *microgrids*, denominada MGridML. De modo semelhante, no nosso trabalho propomos a CSVM, como sendo uma plataforma para execução de modelos descritos por meio da CSML e específicos para o domínio de *crowdsensing*.

Na CVM, m@rt são empregados a fim de adaptar uma sessão de comunicação em andamento para atender novos requisitos especificados pelo usuário. Na MGridVM, m@rt são empregados para permitir que propriedades específicas definidas pelo usuário da *microgrid* sejam mantidas e façam parte do processo de gerenciamento dos elementos da rede. Enquanto na CVM todos os nós executam uma instância da VM, a MGridVM adota uma arquitetura centralizada. Por sua vez, a CSVM adota uma arquitetura de distribuição que apresenta duas instâncias de VM distintas: CSVM4Dev e CSVMProvider.

## 7. Conclusão

Este artigo apresentou uma arquitetura para *crowdsensing* móvel, baseada em uma DSML, denominada CSML, e uma máquina virtual, a CSVM, para gerenciamento dos modelos em tempo de execução. A linguagem CSML permite modelar o registro de dispositivos de sensoriamento e especificar as consultas requeridas pelos usuários em

cenários de *crowdsensing*. A CSVM é uma máquina de execução de modelos que permite a adaptação dinâmica das funcionalidades de *crowdsensing* previamente modeladas. A arquitetura de distribuição da CSVM provê dois componentes: o CSVMProvider, que é o provedor do serviço para o usuário e o CSVM4Dev, que é o gerenciador das capacidades de sensoriamento dos dispositivos móveis. Maiores detalhes sobre a arquitetura e implementação da CSVM podem ser obtidos em [Melo 2014].

A principal contribuição deste trabalho foi a demonstração de como uma arquitetura baseada em modelos em tempo de execução é capaz de manter uma representação atualizada de consultas que envolvem sensores distribuídos, permitindo a adaptação dinâmica em caso de consultas de longa duração.

## Referências


Allison, M., Allen, A. A., Yang, Z., and Clarke, P. J. (2011). A software engineering approach to user-driven control of the microgrid. In *SEKE*, pages 59–64.

Blair, G., Bencomo, N., and France, R. B. (2009). Models@ run. time. *Computer*, 42(10):22–27.

Chan, H., Chu, T., and Leung, V. (2013). Vita: A crowdsensing-oriented mobile cyber physical system. *IEEE Transactions on Emerging Topics in Computing*.

Clarke, P. J., Hristidis, V., Wang, Y., Prabakar, N., and Deng, Y. (2006). A declarative approach for specifying user-centric communication. In *Collaborative Technologies and Systems, 2006. CTS 2006. International Symposium on*, pages 89–98. IEEE.

Deng, Y., Masoud Sadjadi, S., Clarke, P. J., Hristidis, V., Rangaswami, R., and Wang, Y. (2008). Cvm–a communication virtual machine. *Journal of Systems and Software*, 81(10):1640–1662.

Ganti, R. K., Ye, F., and Lei, H. (2011). Mobile crowdsensing: Current state and future challenges. *Communications Magazine, IEEE*, 49(11):32–39.

Hachem, S., Pathak, A., and Issarny, V. (2014). Service-oriented middleware for large-scale mobile participatory sensing. *Pervasive and Mobile Computing*, 10:66–82.

Mathur, S., Jin, T., Kasturirangan, N., Chandrasekaran, J., Xue, W., Gruteser, M., and Trappe, W. (2010). Parknet: Drive-by sensing of road-side parking statistics. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, MobiSys '10, pages 123–136, New York, NY, USA. ACM.

Melo, P. C. F. (2014). Csvm: Uma plataforma para crowdsensing movel dirigida por modelos em tempo de execucao. Master's thesis, Mestrado em Ciencia da Computacao, Instituto de Informatica, Universidade Federal de Goias.

Omg, Q. (2008). Meta object facility (mof) 2.0 query/view/transformation specification. *Final Adopted Specification (November 2005)*.

Ra, M.-R., Liu, B., La Porta, T. F., and Govindan, R. (2012). Medusa: A programming framework for crowd-sensing applications. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pages 337–350. ACM.

Wang, Y., Clarke, P. J., Wu, Y., Allen, A., and Deng, Y. (2008). Runtime models to support user-centric communication. *Models@runtime Workshop in conjunction Models*.