# Uma Proposta de Arquitetura para Virtualização de Sensores Multimídia na Borda da Rede

**Anselmo Battisti, Débora Christina Muchaluat-Saade, Flávia C. Delicato**

Laboratório MídiaCom, Instituto de Computação – Universidade Federal Fluminense
Niterói, RJ, Brasil

{anselmo, debora}@midiacom.uff.br, fdelicato@ic.uff.br

***Abstract.*** *With the increase of camera and microphone use in IoT, managing multimedia streams has become a challenge. Applications that consume discrete data typically virtualize their sensors in the Cloud, while multimedia applications consume continuous data and are latency sensitive. With enhancing capabilities in Edge Computing and its proximity to multimedia sensors, the edge has become a prime place for managing multimedia streams. In this work, we propose V-PRISM, an architecture for virtualizing multimedia sensors deployed at the edge network. Adopting V-PRISM can reduce device battery and IoT network resource consumption, reduce end-to-end delay, and increase ROI for infrastructure providers.*

***Resumo.*** Com a proliferação do uso de câmeras e microfones na IoT, o gerenciamento de fluxos multimídia tornou-se um desafio. Aplicações que consomem dados discretos tipicamente virtualizam seus sensores na Cloud. Já as aplicações multimídia consomem dados contínuos, são sensíveis à latência e, com o aumento das capacidades de computação na borda da rede e por sua proximidade aos sensores multimídia, a borda tornou-se um local privilegiado para o gerenciamento de fluxos multimídia. Neste trabalho propomos V-PRISM, uma arquitetura para virtualizar sensores multimídia implantada na borda. Sua adoção reduz o consumo de recursos dos dispositivos e rede IoT, reduz o atraso fim-a-fim e aumenta o ROI para provedores de infraestrutura.

## 1. Introdução

Com a proliferação da Internet das Coisas (IoT, do inglês *Internet of Things)*, e sua integração com a Computação em Nuvem, surgiu um novo paradigma chamado *Cloud of Things (*CoT*)* [Botta et al. 2016] que explora a sinergia entre IoT e *Cloud.* Neste cenário, ambientes, pessoas e objetos são fontes constantes de geração de dados, que por sua vez, são consumidos por aplicações intermediadas pela *Cloud*. A *Cloud* oferece grande capacidade de processamento e armazenamento para os dados produzidos pelos dispositivos IoT além de abstrair sua heterogeneidade.

Uma das tecnologias que possibilitam a existência da *CoT* é a virtualização. A virtualização é a abstração lógica de características de um dispositivo de hardware ou *software*, e seu uso promove o desacoplamento entre entidades consumidoras e produtoras. Além de sua adoção em *Cloud Computing*, a virtualização vem sendo empregada em diversos cenários, como redes de computadores [Pattaranantakul et al.

2018], e também em propostas de arquiteturas para virtualização de sensores tradicionais [Santos and Guerra 2015].

Dentre os diversos tipos de dispositivos que integram a IoT e, consequentemente sistema de *CoT*, os dispositivos multimídia são uma parte relevante. Em relatório produzido pela Cisco [Barnett et al. 2018], estima-se que em 2022 aproximadamente um terço de todas as conexões da Internet serão do tipo *machine to machine* (M2M) e que aproximadamente 80% de toda a banda consumida na Internet será composta por fluxos multimídia. Por conta da relevância deste tipo de dispositivo, surgiu o conceito de *Internet of Multimedia Things* (IoMT) [Alvi et al. 2015]. Na *IoMT* os sensores são câmeras e microfones com capacidade limitada de processamento e armazenamento, possuindo conectividade com dispositivos heterogêneos e aplicações diversas. Sensores tradicionais, tais como termômetros ou detectores de luminosidade, produzem fluxos de dados discretos. Por outro lado, sensores multimídia produzem fluxos de dados contínuos, volumosos, com estrutura não trivial e significância temporal. Tais características tornam o processamento dos fluxos multimídia mais complexo do que o dos fluxos tradicionais. Além disto, há grande heterogeneidade nos padrões de comunicação e formatos de dados suportados pelos dispositivos multimídia, acrescentando uma complexidade extra na aquisição, no processamento e no consumo destes dados. As particularidades dos fluxos no contexto da *IoMT* fazem com que o uso de modelos de virtualização de sensores tradicionais não seja adequado, sendo necessário conceber modelos talhados para sensores multimídia. Mecanismos que possam abstrair tal complexidade permitirão o desenvolvimento de serviços inovadores em áreas como segurança, saúde, educação e entretenimento.

Em um ambiente CoT, os dados multimídia produzidos pelos sensores físicos são processados por sensores virtualizados na *Cloud*. Entretanto, aplicações multimídia são pouco tolerantes à latência, e muitas vezes a *Cloud* pode não atender aos requisitos específicos deste tipo de aplicação. Uma estratégia para reduzir a latência é migrar o processamento do fluxo multimídia da *Cloud* para a borda da rede, posicionando o elemento de computação nas proximidades dos sensores físicos que geram os dados. Esta estratégia vem sendo explorada pelos paradigmas recentes de *Edge Computing e Fog Computing,* nos quais recursos de computação, armazenamento e comunicação são implantados também na borda da rede [Mukherjee et al. 2018]. Alguns benefícios obtidos pelo seu uso são diminuição da latência, maximização do uso dos recursos e aumento da segurança dos dados. Neste trabalho adotamos o termo *edge* e borda da rede como sinônimos.

Neste contexto, a principal contribuição deste trabalho é apresentar uma arquitetura para virtualização de sensores multimídia na *Edge* chamada V-PRISM. Ela segue uma arquitetura em três camadas composta por *Cloud, Edge* e *IoT Device*. Os fluxos multimídia produzidos pelos dispositivos físicos são processados em componentes chamados *Virtual Multimedia Sensors* (VMS) que estão implantados na *Edge*. A adoção da *V-PRISM* promove a diminuição do consumo de bateria e processamento dos sensores físicos, redução do consumo de banda da rede IoT (rede sem fio que conecta objetos inteligentes ao nó de borda), diminuição da latência de processamento dos fluxos multimídia e aumento do *ROI* (*Return on Investiment*) para os provedores de infraestrutura de sensores. Além disto, acredita-se que a V-PRISM seja uma proposta inédita pois, apesar dos esforços, não foi encontrada na literatura outra arquitetura para criação de VMS cujos componentes sejam executados na borda da rede.

Este trabalho está estruturado da seguinte forma. Na Seção 2 é apresentada uma visão geral sobre as principais abordagens para virtualização de sensores existentes na literatura, e destacado o diferencial em relação a presente proposta. Na Seção 3, é apresentada a arquitetura V-PRISM. Na Seção 4, é apresentada uma implementação da V-PRISM. Na Seção 5, são apresentados os resultados da avaliação da proposta. Na Seção 6, são feitas as considerações finais e apontadas possíveis extensões futuras.

## 2. Trabalhos Relacionados

Como comentado anteriormente, na C*loud of Things* (*CoT*), a *Cloud* age como uma camada intermediária entre os sensores e as aplicações consumidoras dos dados produzidos. Para reduzir a complexidade existente devido à heterogeneidade dos sensores, passou-se a adotar a estratégia de virtualização de sensores na *Cloud*. A virtualização de sensores vem sendo estudada por diversos pesquisadores [Khansari et al. 2018] [Santos and Guerra 2015] [Santos et al. 2015] [Zaslavsky et al. 2013], pois promove a interoperabilidade entre as partes provedoras e consumidoras de recursos, já que abstrai as especificidades da infraestrutura física – no caso, dos dispositivos sensores. A virtualização também contribui para a diminuição dos custos de manutenção da infraestrutura de sensores e para o aumento da disponibilidade de serviços. Esta seção trata de algumas das abordagens para a virtualização de sensores tradicionais.

Uma das primeiras abordagens para virtualização de sensores foi proposta em [Alam et al. 2010]. Nela, cada sensor virtual funciona como um *serviço web* cujos dados são consumidos por múltiplas aplicações. O objetivo é criar uma arquitetura em três camadas, a saber: camada de acesso ao dispositivo, camada semântica e a camada para prover o serviço. Cada uma abstrai a complexidade da camada inferior.

Outra estratégia para definição de sensores virtuais é a criação das redes virtuais de sensores (VSN – *Virtual Sensor Networks*). No trabalho [Islam et al. 2012], os autores definem as VSNs como especialização das redes de sensores sem fio (WSN – *Wireless Sensor Networks*). Essa especialização consiste em duas partes, sendo a primeira a infraestrutura física de sensores (SInP) e a segunda a camada de abstração e agregação de dados dos sensores. Um exemplo de aplicação VSN foi proposto por [Kulkarni et al. 2005], onde um conjunto de câmeras heterogêneas foi utilizado na composição de serviços como detecção, reconhecimento e rastreamento de objetos. Outro trabalho que adota esta estratégia é proposto em [Santos and Guerra 2015] com o *framework* Osiris, onde cada sensor físico é mapeado em uma camada chamada *SensorNet* e as aplicações consomem dados na camada *VirtualSensorNet*.

Um recurso fundamental para a virtualização de sensores é a categorização e recuperação dos dados. Uma forma de realizar essa tarefa é pela criação de ontologias. Tal estratégia foi proposta em [Calbimonte et al. 2011]. Nela, cada sensor publica em um repositório os tipos de dados que ele fornece e as aplicações fazem requisições sobre o tipo de dados que elas necessitam ao repositório central. A descrição dos dados de cada sensor é feita no padrão OGC[1] especificado pelo W3C.

A virtualização de sensores pode adotar uma abordagem híbrida para o local de processamento do dado. Nesse cenário, algumas solicitações são processadas na *Cloud*, enquanto outras são processadas no próprio sensor virtual. Uma implementação dessa

[1] Open Geospatial Consortium: http://www.opengeospatial.org/

abordagem foi proposta em [Santos et al. 2015] onde, de acordo com o tipo e restrições da demanda, o dado pode ser entregue pela *Cloud* ou pelo sensor virtual.

A proposta apresentada em [Alves et al. 2020] traz um modelo de virtualização onde a *Edge* pode também ser utilizada como local para a implantação dos sensores virtuais. Os sensores virtuais são divididos em três categorias: *SensingVN*, *DatahandlingVN* e *ActuationVN,* dependendo do serviço que eles fornecem. Além disso, foi implementado um processo de colaboração para criação de nós virtuais em nós *Edge* vizinhos ao nó para a qual o serviço foi solicitado. Esse recurso permite a distribuição do processamento pela *Edge.*

Os sensores multimídia, diferentemente dos sensores tradicionais, geram dados de forma contínua, que são volumosos (consumindo assim grandes quantidades de banda da rede e armazenamento), além disso, os dados trafegados no fluxo multimídia possuem uma sintaxe mais complexa em relação aos sensores tradicionais. Como exemplo, pode-se citar uma câmera, que gera dados a uma taxa de milissegundos, enquanto os sensores tradicionais de temperatura geram dados em uma taxa na ordem de segundos. Além disso, o dado gerado por um sensor de temperatura tem seu tamanho medido em bits, enquanto um fluxo de vídeo tem facilmente vários megabits. Por estas razões, a virtualização de sensores multimídia tem requisitos diferentes da virtualização de sensores tradicionais e exige soluções que considerem aspectos específicos da comunicação multimídia, como grandes volumes de dados e restrições temporais. O uso de *Edge Computing* tem bastante potencial neste cenário, como proposto neste artigo.

Nesta seção, foram apresentadas algumas propostas já realizadas para a virtualização de sensores. Elas foram desenvolvidas para um ambiente tipicamente de duas camadas (camada dos sensores e camada da *Cloud*). Na proposta deste artigo, o objetivo é virtualizar sensores multimídia na *Edge*, estratégia ainda não abordada na literatura. Os sensores multimídia geram dados complexos, portanto é necessário estabelecer uma nova arquitetura para a criação e o gerenciamento deste tipo específico de sensor virtual.

## 3. Arquitetura V-PRISM

Nesta seção, é apresentada a arquitetura V-PRISM (*Virtual Programmable IoT Sensor Multimedia*). Ela segue uma arquitetura em três camadas: *Cloud*, *Edge* e *IoT Device,* similar a proposta em [Li et al. 2017]. Os componentes de software da arquitetura, apresentados na Figura 1, são responsáveis pelo processamento dos fluxos multimídia produzidos por sensores físicos (MS – *Multimedia Sensor*) que são consumidos pelas aplicações (APP). As APPs tipicamente estão implantadas na *Cloud* e os sensores multimídia físicos estão no ambiente IoT *(*camada *IoT Devices)*. O processamento do fluxo é realizado pelos VMS, que assim como os demais componentes da V-PRISM estão implantados na *Edge*.

### 3.1 VMS e SRC

Um VMS é o componente responsável por realizar o processamento de um fluxo multimídia, por exemplo, converter um áudio *WAV* em *MP3*, ou realizar o reconhecimento facial a partir de um vídeo. Os VMS enviam os fluxos de dados processados diretamente para uma aplicação multimídia ou para outros VMSs. Para cada tipo diferente de processamento de fluxo multimídia, existirá um tipo de VMS.

Cada VMS possui múltiplas portas de entrada (conectadas a VMSs ou a SRCs, explicados a seguir) e múltiplas portas de saída, enviando fluxos para uma ou mais aplicações. Observa-se que na Figura 1 a seta existente entre o VMS e a aplicação é unidirecional, implicando assim que a aplicação não possui um canal de comunicação direto com o VMS. Caso a aplicação queira enviar alguma mensagem de controle, isso deverá ser realizado utilizando os recursos de comunicação do IoT *Broker* (também explicado mais a frente).

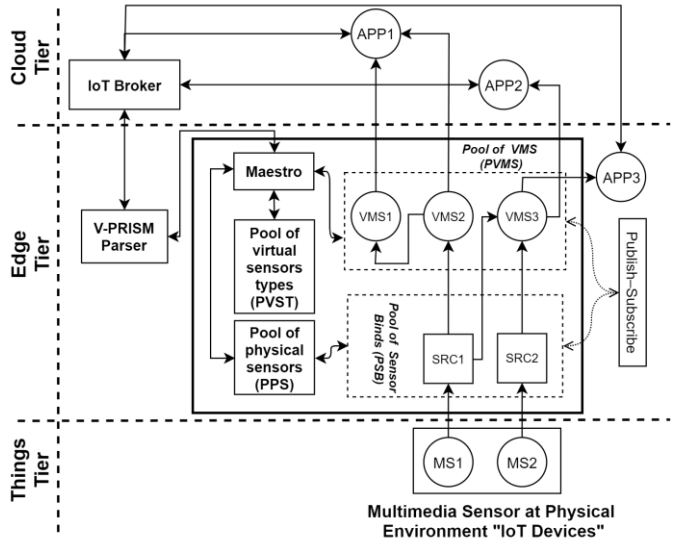


**Figura 1 – Visão Macro da arquitetura V-PRISM**

Um SRC *(Source)* é responsável por encaminhar o fluxo produzido pelos *IoT Devices* para os VMS. Cada SRC está conectado a um sensor de um *IoT Device*. Caso o *IoT Device* tenha mais de um sensor (câmera e microfone), e o fluxo produzido for multiplexado, então existirá apenas um SRC conectado ao *IoT Device*.

Cada SRC é programado para conectar-se a um tipo de dispositivo (e.g. câmera USB) e encaminhar um tipo de fluxo multimídia (e.g. vídeo H.264). Essa abordagem torna a arquitetura flexível para que dispositivos heterogêneos possam ser utilizados em conjunto. O SRC é agnóstico quanto ao fluxo transportado, pois ele não possui nenhum conhecimento sobre seu conteúdo. Essa característica diminui a complexidade para a criação de novos SRC. Um SRC pode encaminhar os dados para vários VMSs.

Um dispositivo físico pode possuir diversos sensores e ter a capacidade de fornecer dados em diversos protocolos/formatos. Cada sensor deve estará conectado com um único SRC e esta relação deve ser configurada durante o processo de implantação da V-PRISM. O tipo do SRC vinculado ao VMS deve ser compatível com o tipo de fluxo de dados e o padrão de comunicação oferecido pelo sensor físico. O SRC opera como um *virtual device* ou ainda como um *driver*, interpretando formatos de dados e protocolos específicos.

Um SRC e um VMS são fracamente acoplados, pois pode haver a troca do SRC que fornece o fluxo para o VMS em tempo de execução. Por exemplo, caso a câmera que forneça dados para um VMS deixe de funcionar, outro SRC vinculado a outra câmera pode ser acoplado ao VMS sem que ele tenha que ser reinicializado. A troca de mensagens de controle entre SRC e VMS é via mecanismos *publish-subscribe*.

Um VMS é definido como uma tupla VMS={P, O, S, D, C} onde, **P** é um conjunto de parâmetros de configuração, onde cada tipo de VMS possui parâmetros distintos, por exemplo: nível de saturação, nível de ruído, qualidade do vídeo, etc.; **O** é um conjunto formado por SRCs ou VMSs que geram o fluxo multimídia. **S** é o *software* que realiza o processamento do fluxo multimídia; **D** é uma tupla {IP, PORTA} que define o endereço onde a aplicação que receberá o fluxo está implantada; **C** é a categoria ao qual o VMS está associado.

A arquitetura V-PRISM é flexível pois permite a criação de VMS de diversas categorias. As categorias descritas abaixo já foram identificadas, mas a arquitetura é extensível para acomodar novas, conforme as necessidades das aplicações ou evolução da tecnologia de sensores.

a) **Replicador**: Todas as características e comportamentos de um sensor físico são replicadas em uma entidade virtual. VMS nesta categoria são utilizados para reduzir o consumo de recursos no dispositivo físico e na rede IoT, já que atuam como um "substituto" virtual do sensor físico, diminuindo a necessidade de acessá-lo diretamente;

b) **Selecionador**: O fluxo de entrada do VMS é gerado por um conjunto de sensores físico ou de VMSs. O fluxo de saída desse VMS é fluxo de apenas uma das fontes de dados. A escolha do fluxo de saída dependerá de critérios definidos no VMS, os quais podem refletir requisitos específicos da aplicação ou parâmetros de desempenho/qualidade. Alguns critérios são a qualidade do dado (acurácia) e seu frescor (há quanto tempo o fluxo foi produzido);

c) **Agregador**: Nesta categoria encontram-se os VMSs que recebem vários fluxos de entrada de diferentes sensores físicos ou VMSs e os combinam em um único fluxo de saída. Um vídeo mosaico ou um VMS que combina o fluxo de um microfone e de uma câmera faz parte desta categoria;

d) **Aprimorador**: VMSs nesta categoria oferecem funcionalidades extras que não existem no sensor físico. Alguns tipos de aprimoramentos são controle de confiabilidade, mecanismos de segurança, desativação e ativação periódica;

e) **Conversor**: São VMSs utilizados quando aplicações e sensores físicos são incompatíveis quanto ao fluxo multimídia. Por exemplo, a aplicação espera um fluxo MP3 ao passo que o sensor produz um fluxo WAV;

f) **Transformador**: Executam processamentos em um fluxo multimídia de tal forma que a resultante seja diferente do fluxo original. A aplicação espera um vídeo H.264 em tons de cinza, mas o sensor físico produz vídeo H.264 colorido;

g) **Detector**: Esta categoria possui VMS mais complexos do que as anteriores pois é realizado no fluxo multimídia algum tipo de processamento para detecção de padrões ou eventos. Técnicas de inteligência artificial podem ser empregadas nesta categoria. VMSs que disparam um alerta ao detectar um foco de incêndio ou ao identificar um veículo roubado fazem parte desta categoria.

### 3.2 Módulos *PPS, PSB e PVST*

O módulo *Pool of Physical Sensors* (PPS) é responsável por armazenar os metadados dos sensores físicos conectados a V-PRISM. Entre os dados armazenados podemos destacar o formato do fluxo gerado, a localização física do sensor dentro da

organização, a taxa de amostragem de dados etc. Os atributos armazenados sobre cada sensor são distintos e variam de acordo com a seu tipo e característica física. A Figura 2 ilustra a distribuição dos sensores físicos e virtuais dentro de uma organização.

O PPS deve ser atualizado sempre que um sensor físico for implantado, removido ou modificado. Os dados do PPS não são enviados para as aplicações. A aplicação receberá o fluxo multimídia, mas não saberá qual sensor físico o produziu, mantendo assim o desacoplamento provido pela virtualização. Na Figura 2(a), observa-se a distribuição física dos sensores dentro de uma organização. A relação de sensor físico por local não é exposta para a aplicação. A aplicação saberá que no ambiente denominado "externo fundos" (Figura 2), existe um VMS de detecção de intrusos, porém ela não saberá se o dado recebido foi gerado pela "câmera 360" ou "Webcam".

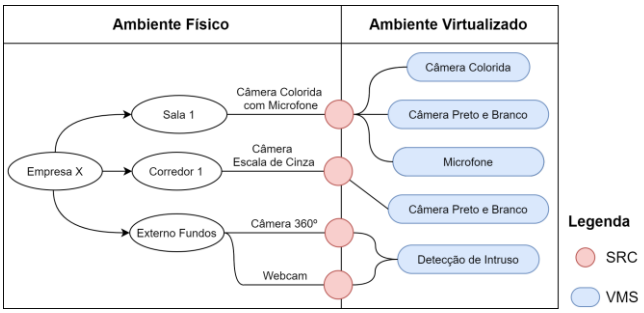


**Figura 2 - Exemplo de lista dos sensores físicos e virtuais de uma organização**

O módulo *Pool of Sensor Binds* (PSB) é o responsável por armazenar as conexões (*binds*) existentes entre um SRC e os VMS. É importante observar que não existe uma definição estática sobre qual SRC será vinculado com qual VMS. Tal vinculação é realizada em tempo de execução de acordo com critérios e restrições existentes quando o VMS é instanciado. Além disso, caso um sensor físico falhe, é possível alterar o SRC que fornece dados para o VMS em tempo de execução, tornando assim a V-PRISM flexível e com mecanismos de tolerância a falhas.

Uma possível estratégia de alocação de recursos entre SRC e VMS é a seguinte. Caso dois ou mais SRCs estejam conectados a microfones na mesma sala, e eles possam entregar fluxos com *bitrate* de 96 kbps, então será escolhido o SRC que tem a menor taxa de perda de pacotes. Vale lembrar que a estratégia de alocação de recursos ficará sob responsabilidade da implementação da V-PRISM.

O módulo *Pool of Virtual Sensor Types* (PVST) é responsável por armazenar todos os possíveis tipos de VMS cuja criação pode ser solicitada pelas aplicações. Este componente envia os dados para o Maestro, apresentado na próxima seção, que por sua vez envia ao IoT *Broker* a fim de que os dados possam ser acessados pelas APPs.

Um *Tipo de VMS* refere-se a um *software* que realiza processamento específico em um fluxo multimídia. Alguns possíveis exemplos são: detecção de ruído, conversão de vídeo colorido em tons de cinza, encaminhador de fluxo com buffer. Os tipos de VMS são utilizados para a criação de um VMS. Cada VMS será um processo ou instância de seu respectivo tipo e terá seu ciclo de vida controlado pela V-PRISM. Cada *Tipo de VMS* pode ser associado em uma ou mais categorias de VMS. Durante a implantação da V-PRISM é necessário definir quais tipos de VMS poderão ser utilizados pelas aplicações para a criação de VMS. Isso permite que sejam oferecidos apenas VMS compatíveis com os recursos físicos existentes na organização e de acordo com suas regras. Em um ambiente IoT que possui apenas microfones, não faz sentido existirem

tipos que necessitem câmeras, ou ainda, o fato de existir uma câmera não implica que o tipo de VMS que conta pessoas a partir de vídeo possa ser inicializado. Esse tipo de personalização torna a V-PRISM flexível.

O módulo *Pool of VMS* (PVMS) é o componente que armazena dados sobre os VMS criados. Ele também executará um monitoramento proativo e caso haja alguma falha então o VMS será desativado e outro será instanciado e vinculado ao SRC.

### 3.3 Maestro

O módulo Maestro realiza a orquestração dos demais componentes da V-PRISM. Ele possui um conjunto de APIs que são consumidas pela *V-PRISM Parser*, apresentado na próxima seção, para que os dados internos possam ser acessados pelas aplicações. Caberá ao maestro intermediar as requisições para a instanciação de novos VMS.

Uma das funções do Maestro é enviar para o IoT *Broker* informações referentes aos tipos de VMS e em quais locais da organização eles podem ser instanciados. Estas informações são fornecidas pelo módulo PPS. Um exemplo de como funciona esta exportação pode ser visto na Figura 2(b), onde são apresentados os tipos de VMS possíveis de serem instanciadas bem como a sua localização dentro de uma organização. O Maestro também recebe das aplicações, intermediado pelo *IoT Broker*, solicitações para a criação de novos VMSs. A criação do VMS só ocorrerá se existirem recursos suficientes no *Edge Node* onde a V-PRISM está sendo executada. A destruição do VMS deverá ser solicitada pela aplicação que demandou sua criação. Essa estratégia é a mesma adotada em ambientes *Cloud* onde o responsável pela destruição da máquina virtual é o agente que solicitou sua criação.

### 3.4 IoT *Broker* (IB) e *V-PRISM Parser*

Com o objetivo de prover o isolamento entre a V-PRISM e as aplicações que demandam por um VMS, a comunicação entre eles (exceto o fluxo de dados multimídia) é realizada por intermédio de um IoT *Broker* (IB). Um IoT *Broker* é um caso particular de um *Message Broker* [Kovacs et al. 2016]. O uso de um IoT *Broker* permite que sistemas heterogêneos se comuniquem. Em ambientes IoT onde a interoperabilidade entre sistemas heterogêneos é um requisito crítico, essa abordagem é largamente adotada.

Na V-PRISM, existe um componente intermediário entre o IoT *Broker* (*IB*) e a V-PRISM, o *V-PRISM Parser* (VP), que é um *proxy* entre a V-PRISM e o IoT *Broker*. O objetivo do VP é promover a independência entre o V-PRISM e o IoT *Broker*, ou seja, a V-PRISM não terá uma interação direta com o IoT *Broker* adotado na implementação da arquitetura, mas sim com o VP e o VP terá os mecanismos necessários para traduzir as requisições recebidas pelo IoT Broker em chamadas internas ao Maestro.

Na Figura 3 pode ser visto o digrama de sequência do processo de solicitação de criação de um VMS por uma aplicação. Inicialmente a aplicação envia ao IB uma mensagem perguntando quais os possíveis tipos de VMS que podem ser instanciados. Como retorno, o IB envia os dados sobre o ambiente onde a V-PRISM está implantada. De posse dos dados, a aplicação pode então realizar a solicitação para a criação de um VMS, por exemplo, “criação de um VMS do tipo câmera no corredor 1 cujos dados de retorno sejam em escala de cinza e o formato seja H.264”. O processo de criação é assíncrono, ou seja, a aplicação somente ficará sabendo sobre o andamento do processo

ou realizando *polling* junto ao IB ou quando o IB realizar uma chamada de *call-back*. Vale lembrar que esse tipo de comunicação é totalmente independente da V-PRISM ficando a cargo da aplicação gerenciar essa complexidade. Após o IB receber a solicitação para a criação do VMS, ele se comunicará com o módulo Maestro utilizando para isso o VP. Após o Maestro receber a requisição de criação do VMS, ele irá executar os módulos necessários para a efetiva inicialização do VMS. Em seguida, será enviada uma mensagem sobre a situação do processo. Na próxima seção, é apresentada a implementação da arquitetura V-PRISM bem como suas especificações técnicas.

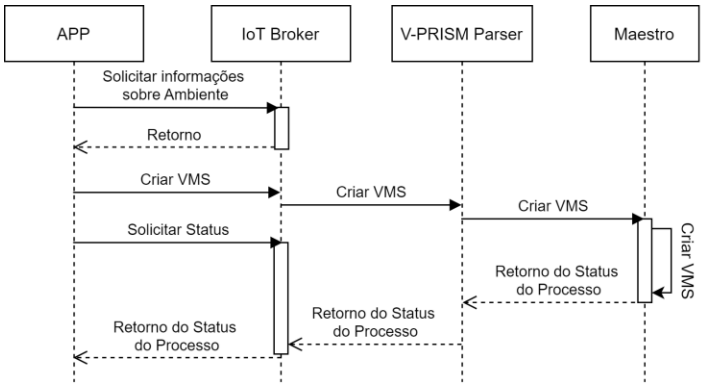


**Figura 3 - Diagrama de sequência para criação de um VMS**

## 4. ALFA - Uma Implementação da V-PRISM

A V-PRISM é independente de tecnologia. Ela foi planejada com o objetivo de padronizar a criação de sistemas para o gerenciamento de VMS. O código fonte da prova de conceito (ALFA) desenvolvida para validar a arquitetura e as instruções de instalação estão disponíveis sob licença "*MIT License*" no *github* (https://bit.ly/2B19DYa).

O ALFA foi implementado utilizando como IoT *Broker* o FIWARE [FIWARE 2019]. O FIWARE é um framework planejado para dar suporte ao desenvolvimento de *smart solutions*. Ele foi escolhido devido a sua ampla utilização na academia e indústria, por ser *open source*, orientado a APIs, e utilizar o Docker para virtualizar suas funções. O FIWARE fornece blocos de construção, chamados *Generic Enablers* (GEs), que são componentes de uso geral e estão disponíveis para uso nas camadas de *Cloud* e *Edge*. Os GEs são empacotados e distribuídos em imagens de *containers* reduzindo, portanto, o esforço para integrar os novos componentes necessários para a implementação da arquitetura proposta. A comunicação entre o FIWARE e o módulo Maestro é realizada utilizando como intermediário o módulo ALFA_FIWARE, o qual provê APIs que permitirão a comunicação entre as aplicações e os módulos internos do ALFA.

Em [Morabito 2017], os autores argumentam que a adoção da virtualização leve na *Edge* com a tecnologia de *containers* pode trazer benefícios para a IoT. Ferramentas de virtualização leve como *Docker*, *Kubernetes* e *OPENSHIFT* usam, de forma compartilhada com o sistema operacional da máquina hospedeira, bibliotecas básicas do sistema. Essa estratégia economiza processamento e memória. No ALFA, foi utilizado o Docker como sistema de virtualização leve. Ele foi escolhido pois é requisito do ALFA a orquestração (criar, destruir, monitorar e configurar) dos *containers* que executam os VMSs e SRCs via API. Além disso, é necessário que a inicialização dos *containers* não gere sobrecarga no sistema hospedeiro e o Docker possui tal característica [Xavier et al.

2016]. É importante observar que o Docker não está sendo executado no sensor multimídia e sim na *Edge* onde as restrições de energia são mais frouxas. Além disso, o Docker possui mecanismos automáticos para balanceamento de carga. Por serem executados dentro de um container Docker, VMS mais complexos são criados utilizando linguagens de programação como C, Python ou Java. No ALFA, os VMSs são escritos em C, pois a biblioteca gstreamer, que fornece a API usada para o tratamento de fluxos multimídia, também é escrita nessa linguagem.

Dentro do ALFA, todas as informações são armazenadas em um banco de dados mongoDB. Ele foi escolhido por ser não relacional, adotado largamente pela indústria, possuir baixo overhead de processamento e ter uma inicialização rápida. Além disso, como o ALFA foi desenvolvido em Javascript e o mongoDB armazena dados usando a notação JSON, sua escolha facilitou o processo de integração.

## 5. Avaliação da Proposta

Nesta seção, são descritos os experimentos realizados com o objetivo de avaliar a proposta da V-PRISM. Os primeiros experimentos avaliam se um *IoT Device* pode fornecer fluxos multimídia para diversas aplicações que esperam fluxos diferentes dos produzidos pelo sensor físico. Em seguida, é apresentado um experimento que avalia o consumo de bateria e processamento no *IoT Device,* bem como o consumo de banda trafegada na rede IoT, quando múltiplas aplicações demandam fluxos de um mesmo *IoT Device*. Vale ressaltar que a aplicação proposta para este experimento não reflete a amplitude de possibilidades de uso do VMS. Ela foi escolhida por representar uma aplicação simples e assim permitindo a análise isolada das variáveis avaliadas. Por fim, foi elaborado um experimento para avaliar a diferença de latência entre um VMS implantado na *Edge* e na *Cloud.* Os experimentos foram realizados em máquinas virtualizadas no *VirtualBox*, com Ubuntu 18.04, 8GB de RAM e 4 processadores Intel(R) Core (TM) i7-8565U. Foi usado o Docker na versão 19.03 para virtualizar os componentes do ALFA.

### 5.1 Casos de Teste e Validação da Arquitetura

Os componentes do ALFA são acessados via API REST. Na Figura 4, pode ser vista a interface web desenvolvida para consumir esta API. Na figura, é executado o comando para a criação de um novo VMS do tipo *Video Merge* (categorizado como Agregador). Ao criar o VMS, é necessário informar o endereço IP e PORTA, no campo *startup parameters,* os quais definirão o destino do fluxo após processado pelo VMS. Este IP é o onde a aplicação que irá consumir o fluxo multimídia está sendo executada.

Para o primeiro caso de teste, foi criado um tipo de VMS chamado *Noise Detector* (categorizado como Detector). Ele recebe fluxos de áudio e determina o seu volume. Caso o volume seja maior do que o limiar estabelecido durante a instanciação do VMS (requisito da aplicação), então é enviado para um servidor MQTT um alerta sobre a detecção do ruído. A Figura 5 mostra os parâmetros do VMS dentro do ALFA, o servidor MQTT que recebe os alertas, e o *container* Docker que executa o VMS.

No segundo teste, foram criados dois tipos de VMS, *Video Crop* (corta uma região específica de um vídeo, categorizado como Aprimorador) e *Video Grayscale* (conversão de vídeo colorido em tons de cinza, categorizado como Transformador). Foram instanciados dois VMS do tipo *Video Crop* para demultiplexar o fluxo de uma câmera que filmava duas portas, Figura 6(a), e o segmento do vídeo de cada porta foi entregue

para a aplicação específica, como pode ser visto na Figura 6(b) e (c). A Figura 6(c) mostra que o fluxo entregue para a aplicação foi processado inicialmente pelo *Video Crop* e depois pelo *Video Grayscale* criando um encadeamento de processamento entre VMSs, e assim ilustrando uma das características da V-PRISM (Figura 1).

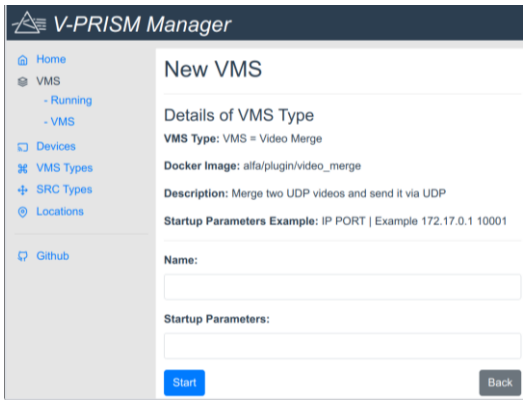


**Figura 4 - Cliente web consumindo a API do ALFA**

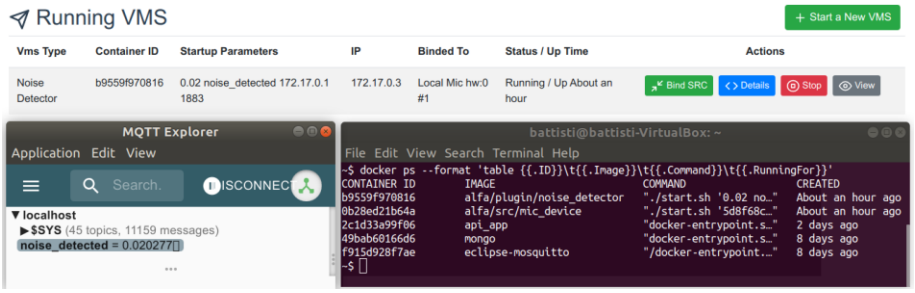


**Figura 5 – Telas de execução do VMS do tipo *Noise Detector***

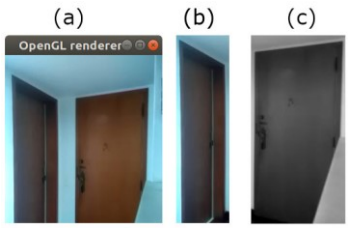


**Figura 6 - Demultiplexação de Vídeo**


**Figura 7 - Vídeo mosaico**

O terceiro caso de teste foi a criação de um tipo de VMS chamado *Video Merge* (categorizado como Agregador). Ele foi utilizado para enviar para uma única aplicação o fluxo de vídeo que é resultado da combinação de fluxos de duas câmeras diferentes. A Figura 7 mostra uma aplicação do *Video Merge,* o vídeo da esquerda é gerado por uma câmera em modo de teste e o vídeo da direta foi capturado de uma webcam e, o resultado do processamento realizado pelo VMS é a combinação dos fluxos distintos em um único fluxo que será enviado para a aplicação de visualização.

A execução do VMS na V-PRISM é realizada na *Edge*, sendo assim o fluxo multimídia não sai da rede interna da organização, o que diminui riscos de violação da privacidade e segurança. Os casos de teste mostram que é possível utilizar o mesmo dispositivo físico como fonte de dados para diversos VMSs que podem atender a diversas aplicações ou mesmo a outros VMSs. O processamento na *Edge* combinado

com o fluxo de sensores multimídia permite a criação de serviços que otimizam os recursos físicos aumento assim o ROI da infraestrutura IoT.

### 5.2 Redução no Consumo de Recursos no Dispositivo IoT

Para avaliar como a adoção da *V-PRISM* influencia o desempenho dos *IoT Devices*, foi desenvolvido um experimento onde um *smartphone* Moto G5 (representando o dispositivo IoMT) com Android 8.1 e um servidor RTSP[2] fornecem vídeo H.264 em 640x480 para diversas aplicações. A comunicação utilizou uma rede WiFi. Foram realizados experimentos em dois cenários: na Figura 8(a) as aplicações acessam o dado diretamente no *smartphone* ao passo que na Figura 8(b) as aplicações acessam o dado mediado pela *V*-PRISM (utilizando um VMS do tipo Replicador). Cada experimento foi executado 5 vezes durante 5 minutos. Os resultados podem ser vistos nas Figuras 9, 10 e 11, com intervalo de confiança de 95%. Os dados foram coletados no sensor pelo software *Batterystats*[3]. Pode-se observar que quanto maior o número de aplicações acessando diretamente o *smartphone* para coletar o dado, maior é o consumo dos recursos. Em contrapartida, quando o acesso é feito através de um SRC, o consumo dos recursos no dispositivo IoMT sofre pouca variação. Os resultados apontam que além do menor consumo de CPU, memória, bateria e rede no dispositivo IoMT. estes valores também são constantes, trazendo assim vantagens no uso da V-PRISM. Vale ressaltar que o consumo de energia e o compartilhamento de recursos são apontados em [Mukherjee et al. 2018] como desafios no ambiente IoT.

### 5.3 Comparação entre a latência na *Edge* e na *Cloud*

Para avaliar o uso da *Cloud* e *Edge* no processamento de fluxos multimídia, foi desenvolvido um experimento no qual dois VMSs do tipo *Noise Detector* foram criados. Um foi implantado na *Edge* e o outro na *Cloud* . Um som de bipe foi reproduzido 80 vezes em um intervalo de 5 segundos e o fluxo foi enviado aos dois VMS em paralelo.

O som captado gerou fluxo com uma taxa média de 1Mb/s. O *delay* foi calculado utilizando o comando ping. Entre o SRC e a *Cloud* era de 67,5ms, já entre o SRC e a *Edge* era desprezível. A banda disponível entre o SRC e a *Cloud* era de 100Mb/s, e no momento dos testes não havia congestionamento. Foi calculado o intervalo de tempo entre os sucessivos reconhecimentos do som no VMS na *Cloud* e na *Edge*. O tempo médio para detecção do ruído na *Edge* foi de 918ms com um desvio padrão de 88ms e na *Cloud* foi de 967ms com um desvio padrão de 137ms. A detecção na *Edge* foi em média de 49ms menor.

Um aspecto digno de nota é que o desvio padrão das detecções na *Edge* foi menor do que o da *Cloud* apontando assim uma maior previsibilidade da latência na *Edge* do que na *Cloud*. Outro ponto importante de se observar é que a diferença média entre o tempo de detecção na *Edge* e na *Cloud* foi menor do que o *delay* entre o SRC e a *Cloud*, isso significa que o tempo de processamento da *Cloud* foi menor do que na *Edge*. Isso ocorre, pois, a *Cloud* utilizada no experimento tem um poder computacional maior do que o *Edge Node*. Portanto, podemos concluir que a escolha entre realizar o processamento na *Edge* ou na C*loud* deve levar em consideração não apenas a latência entre o dispositivo físico e a *Cloud,* mas também o tempo de processamento do fluxo.

[2] https://play.google.com/store/apps/details?id=com.pas.webcam&hl=pt_BR
[3] https://developer.android.com/studio/profile/battery-historian

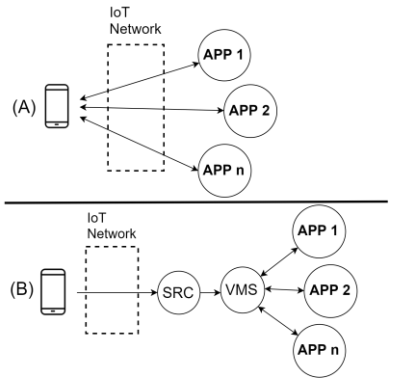


**Figura 8 - Modelo do experimento**

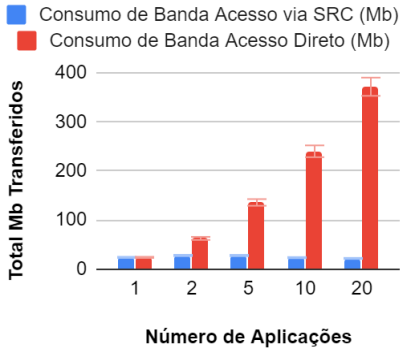


**Figura 9 - Dados trafegados na rede IoT**

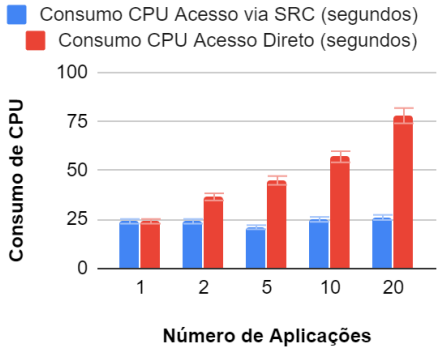


**Figura 10 - Consumo CPU dispositivo IoT**

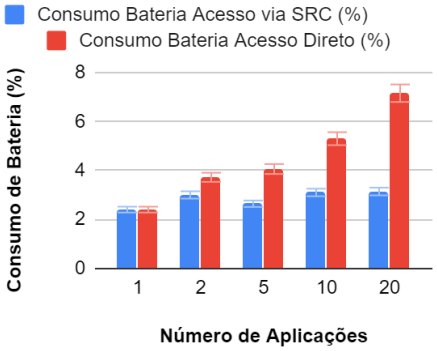


**Figura 11 - Consumo Bateria dispositivo IoT**

## 6. Considerações Finais

Neste trabalho, apresentamos a arquitetura V-PRISM, sua prova de conceito e diversos VMSs foram desenvolvidos para ilustrar os casos de uso. A criação de tipos de VMS permite que novos serviços sejam oferecidos sem a necessidade de instalar equipamentos especializados aumentando o ROI da infraestrutura IoT. Os resultados obtidos no experimento que mediu o consumo de bateria, processamento e banda da rede IoT mostram que o uso da V-PRISM traz benefícios nas três métricas. Destaca-se que quanto maior o número de aplicações que consomem diretamente o fluxo multimídia do dispositivo, maior é o consumo de banda, processamento e bateria. Quando as aplicações consomem o fluxo multimídia intermediado por um VMS, independentemente do número de aplicações, os recursos consumidos no dispositivo IoT não sofrem alteração. Esses resultados podem ser ainda mais expressivos caso a aplicação executada no dispositivo IoMT seja intensiva em uso de CPU. No experimento que mediu a influência da latência, foi constatado uma maior imprevisibilidade ao utilizar a *Cloud* como local de processamento dos VMSs e, que o maior poder computacional da *Cloud* influencia no tempo total de detecção do ruído.

Como trabalhos futuros, pretende-se desenvolver novos tipos de VMSs que utilizem CPU intensivamente, como reconhecimento facial, que serão utilizados para estudar a latência quando processamentos complexos são realizados na *Edge*.


### Agradecimentos

Este trabalho é parcialmente financiado pela Fundação de Amparo à Pesquisa do Estado de São Paulo - FAPESP, através do número da concessão 2015/24144-7 e Fundação de Amparo à Pesquisa do Estado do Rio de Janeiro - FAPERJ. Debora Saade e Flávia Delicato são Bolsistas CNPq.

## 7. Referências


Alam, S., Chowdhury, M. M. R. and Noll, J. (2010). SenaaS: An event-driven sensor virtualization approach for internet of things cloud. *IEEE NESEA*, p. 1–6.

Alves, M. P., Delicato, F. C., Santos, I. L. and Pires, P. F. (7 mar 2020). LW-CoEdge: a lightweight virtualization model and collaboration process for edge computing. *World Wide Web*, v. 23, n. 2, p. 1127–1175.

Alvi, S. A., Afzal, B., Shah, G. A., Atzori, L. and Mahmood, W. (2015). Internet of multimedia things: Vision and challenges. *Ad Hoc Networks*, v. 33, n. May.

Barnett, T., Jain, J. S., Usha, A. and Khurana, T. (2018). Cisco Visual Networking Index ( VNI ) Complete Forecast Update , 2017 – 2022.

Botta, A., De Donato, W., Persico, V. and Pescapé, A. (2016). Integration of Cloud computing and Internet of Things: A survey. *Future Generation Computer Systems*, v. 56, p. 684–700.

Calbimonte, J. P., Jeung, H., Corcho, O. and Aberer, K. (2011). Semantic sensor data search in a large scale federated sensor network. *CEUR Workshop Proceedings*, v. 839, p. 23–38.

FIWARE (2019). About us. https://www.fiware.org/about-us/, [accessed on Dec 6].

Islam, M. M., Hassan, M. M., Lee, G.-W. and Huh, E.-N. (15 feb 2012). A Survey on Virtualization of Wireless Sensor Networks. *Sensors*, v. 12, n. 2, p. 2175–2207.

Khansari, M. E., Sharifian, S. and Motamedi, S. A. (2018). Virtual sensor as a service: a new multicriteria QoS-aware cloud service composition for IoT applications. *Journal of Supercomputing*, v. 74, n. 10, p. 5485–5512.

Kovacs, E., Bauer, M., Kim, J., et al. (2016). Standards-Based Worldwide Semantic Interoperability for IoT. *IEEE Communications Magazine*, v. 54, n. 12, p. 40–46.

Kulkarni, P., Ganesan, D., Shenoy, P. and Lu, Q. (2005). SensEye: A Multi-tier Camera Sensor Network. In *ACM Multimedia*. . ACM Press.

Li, W., Santos, I., Delicato, F. C., et al. (2017). System modelling and performance evaluation of a three-tier Cloud of Things. *Future Generation Computer Systems*, v. 70, p. 104–125.

Morabito, R. (2017). Virtualization on internet of things edge devices with container technologies: A performance evaluation. *IEEE Access*, v. 5, p. 8835–8850.

Mukherjee, M., Shu, L. and Wang, D. (2018). Survey of fog computing: Fundamental, network applications, and research challenges. *IEEE Comm Surveys and Tutorials*, v. 20, n. 3, p. 1826–1857.

Pattaranantakul, M., He, R., Song, Q., Zhang, Z. and Meddahi, A. (2018). Nfv security survey: From use case driven threat analysis to state-of-the-art countermeasures. *IEEE Communications Surveys and Tutorials*, v. 20, n. 4, p. 3330–3368.

Santos, F. and Guerra, R. (2015). OSIRIS Framework : construindo sistemas de monitoramento com redes de sensores sem fio para compartilhar dados. In *XXXIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC), Vitória*.

Santos, I. L., Pirmez, L., Delicato, F. C., Khan, S. U. and Zomaya, A. Y. (2015). Olympus: The cloud of sensors. *IEEE Cloud Computing*, v. 2, n. 2, p. 48–56.

Xavier, B., Ferreto, T. and Jersak, L. (2016). Time Provisioning Evaluation of KVM, Docker and Unikernels in a Cloud Platform. *CCGrid 2016*, p. 277–280.

Zaslavsky, A., Perera, C. and Georgakopoulos, D. (2013). Sensing as a Service and Big Data. *Proceedings of the International Conference on Advances in Cloud Computing*, n. May 2014.