# Aplicação de Computação em Névoa na Internet das Coisas para Cidades Inteligentes: da Teoria à Prática

**Gustavo Uruguay Castilho, Carlos Alberto Kamienski**

Universidade Federal do ABC (UFABC) – Santo André, SP

{gustavo.castilho,cak}@ufabc.edu.br

***Resumo.*** *O conceito de computação em névoa tem como objetivo levar serviços da nuvem para perto dos clientes, a fim de otimizar recursos e aumentar o desempenho das redes de comunicação, uma vez que o modelo tradicional centralizado tem dificuldades de atender as demandas da Internet das Coisas. Este artigo apresenta um modelo genérico para implementação de computação em névoa, que pode ser utilizado como referência para implementações reais. Foi criada uma estrutura em laboratório para demonstrar a viabilidade do modelo de computação em névoa e resultados de avaliações de desempenho indicam situações nas quais é viável a sua utilização, baseado em fatores e peculiaridades específicas de cada caso de uso.*

***Abstract.*** *The fog computing concept aims at taking cloud services closer to the clients to optimize resources and increase network performance, since the current centralized model might not be able to handle the demands typical from the Internet of Things. In this paper, we present a generic model for the deployment of fog computing, which can be used as reference for real use cases. We have set up a testbed to demonstrate the feasibility of our model and present results from a performance analysis study that highlights when the use fog computing is viable, based on factors and peculiarities of each use case.*

## 1 Introdução

O número de dispositivos conectados à Internet está aumentando constantemente, caracterizando a Internet das Coisas (IoT), principalmente na forma de sensores e atuadores, permitindo a sensibilidade e interação com ambientes diversos, gerando oportunidades para melhorar a experiência de vida das pessoas através de casas ou cidades inteligentes, veículos autônomos e monitoramento de saúde. Previsões da Cisco indicam que em 2020 o volume de dados gerados pelos dispositivos que fazem parte da IoT deve chegar a 600ZB por ano [CISCO 2017]. Este cenário cria desafios para enviar e processar todos os dados na nuvem, devido a gerar uma grande demanda de largura de banda, além de não ser capaz de atender demandas que exigem baixas latências, como um caso de veículos autônomos, por exemplo, onde são necessárias respostas em tempo real para o bom funcionamento da estrutura.

Novas tecnologias surgem para prevenir problemas iminentes, como a Computação em Névoa (*Fog Computing*) [Bonomi et. al 2012], que com a utilização de Virtualização de Funções de Rede (NFV) [ETSI. 2013] e Redes Definidas por *Software* (SDN) [MCKEON et al. 2008], visa disponibilizar serviços da nuvem em localidades próximas dos clientes, de forma, que permite escalabilidade, redução de latência e otimização de

recursos, efetuando processamento remoto e evitando que dados desnecessários sejam enviados para a nuvem.

O principal desafio atualmente para a utilização de computação em névoa é a falta de padrões, modelos e arquiteturas para sua implementação. Este artigo trata da relação entre IoT e computação em névoa, tendo como objetivo contribuir com o estado da arte das seguintes formas: a) apresentar um modelo genérico para implementação de computação em névoa; b) utilizar recursos legados e tecnologias como virtualização de funções de rede e redes definidas por *software*; c) efetuar implementação sem utilização de tecnologias, *software*s ou *hardwares* proprietários; d) propor um modelo de controle para estruturas que utilizarão computação em névoa; e) desenvolver um orquestrador para computação em névoa com funções básicas; f) identificar possíveis gargalos nas estruturas e principalmente sugerir situações em que a utilização de computação em névoa é viável, baseado nos resultados dos experimentos efetuados em laboratório.

Os experimentos apresentados foram realizados em laboratório, utilizando simuladores de conexões de rede Wi-Fi e Internet e a partir da implementação de dois casos de uso baseados em dados reais. Um dos casos de uso é a assistência médica por sensores e outro, um cenário hipotético em que veículos acionam os freios automaticamente, controlados por nós de computação em névoa, o qual chamamos de semáforos virtuais.

Os resultados obtidos permitem observar casos em que é viável utilizar computação em névoa para aperfeiçoar certas métricas (latência e vazão de rede), como num caso de uso onde a carga de dados dos sensores é consideravelmente alta (em torno de 30000 Bytes por segundo) e o atraso gerado pela Internet é em torno de 15ms, onde os ganhos considerando apenas a latência compensam o esforço, custo e complexidade para implementação de computação em névoa. Os experimentos mostram que há um aumento na quantidade total de requisições atendidas num determinado período de tempo, o que aumenta a vazão do sistema. Além disso, os resultados corroboram com o especulado pela grande quantidade de trabalhos teóricos que existem até o momento sobre computação em névoa e permitem validar o modelo aqui proposto como passível de utilização em casos de uso reais.

Na sequência deste artigo, a seção 2 apresenta alguns conceitos, seguidos dos trabalhos relacionados. Na seção 3 é descrita a metodologia utilizada e os casos de uso escolhidos para base da implementação, que é descrita em detalhes, juntamente do modelo proposto na seção 4. Os resultados obtidos são mostrados na seção 5, seguidos pelas conclusões, trabalhos futuros e considerações finais na seção 6.

## 2 Conceitos Básicos e Trabalhos Relacionados

Nesta seção são mencionados, de forma sucinta, os principais conceitos utilizados no decorrer deste documento, considerados essenciais para o entendimento do texto.

### 2.1 Computação em Névoa

O desenvolvimento de verdadeiras cidades inteligentes através da computação urbana necessitará da utilização conjunta de várias tecnologias para viabilizar a oferta de novos serviços aos cidadãos [KAMIENSKI et. al 2016]. Mencionado pela primeira vez em 2012 Computação em Névoa [BONOMI et al. 2012], é um paradigma que se propõe a solucionar problemas relativos à grande quantidade de dados que inevitavelmente será

gerada com o aumento de campo de utilização de IoT, solucionando problemas como latência e utilização de banda, trazendo os serviços que hoje estão na nuvem para mais próximo dos usuários finais.

O conceito de Computação em Nuvem (*Cloud Computing*) centraliza os recursos, de modo que podem ser acessados virtualmente de qualquer local que disponibilize acesso à Internet. Esta tecnologia atualmente é imprescindível para os serviços que utilizamos no nosso cotidiano, porém, a computação em nuvem pode perder parte de sua eficiência devida ao advento de novas tecnologias (IoT) que geram um grande volume de dados (*Big Data*), que precisam de atendimento prioritário, inclusive para não sobrecarregar as próprias nuvens e os meios de comunicação.

Pode-se dizer então, que a névoa (*Fog*) é alocada no ponto mais conveniente entre o usuário final e a nuvem (*Cloud*), de modo a otimizar o processamento, armazenamento e transmissão da informação, proporcionando melhoria em todo o processo de comunicação de dados. Estes recursos podem ser alocados em qualquer dispositivo capaz de realizar um processamento básico, ainda que com poucos recursos, dependendo do serviço que se deseja disponibilizar no nó névoa, tornando equipamentos como televisores, pontos de acesso sem fio, equipamentos de rede e semáforos inteligentes, por exemplo, como possíveis pontos de alocação e disponibilização de recursos.

## 2.2 Protocolo MQTT

O protocolo de aplicação MQTT[1] foi criado pela IBM e é muito usado para aplicações IoT, sendo otimizado para algumas características como baixo consumo de energia, baixa sobrecarga (*overhead*) e funcionamento em redes instáveis. O MQTT opera no modelo cliente servidor sobre TCP, sendo que o servidor é chamado de *broker*, e recebe e envia as mensagens para os clientes. No lado do cliente há dois módulos: o *publisher* e o *subscriber*. O *publisher* é o dispositivo que gera a informação e envia para o *broker*, enquanto o *subscriber* é um dispositivo que recebe as mensagens do *broker* conforme o interesse, num contexto de tópicos. Portanto o *publisher* envia as mensagens para o *broker*, e este envia as mensagens para os *subscribers* que estiverem inscritos no tópico que lhes interessa. O MQTT permite três níveis (0, 1 e 2) de qualidade de serviço (QoS), sendo que para este trabalho foi utilizado o nível 2, que proporciona garantia de entrega das mensagens e evita alertas duplicados.

## 2.3 Trabalhos Relacionados

O trabalho de BONOMI et al. [2012] foi um dos primeiros na área de computação em névoa, e trata de conceitos importantes sobre o tema, além de definir as principais características do serviço, abrindo espaço para futuros estudos sobre o tema. Alguns poucos trabalhos demonstram cenários mais específicos de aplicação, como [ZHU et al. 2013] onde é demonstrado um caso de utilização de computação em névoa para otimização de servidores web, através da organização de páginas de estilo, minimização do tamanho e número de requisições HTTP e maximizando a utilização do cache através de especificação estratégica do tempo de expiração de cada objeto.

Conceitos básicos de Computação em Névoa são aplicados em [SOUZA et al. 2016] para tentar solucionar o problema de alocação dos serviços da nuvem em uma estrutura

[1] http://mqtt.org

estática localizada próxima dos clientes, porém ainda sem utilizar alocação dinâmica de nós de computação em névoa e redes definidas por *software*, já que o objetivo principal do trabalho não é medir o desempenho, mas mostrar como tomar a decisão sobre quais serviços serão alocados nos nós névoa.

Em [YIGITOGLU et al. 2017] é proposta uma arquitetura para computação em névoa que leva em consideração os principais aspectos que deve tratar um cenário real (mobilidade, alocação sob demanda, diminuição da latência, entre outros) conforme mencionado na seção referente à teoria sobre computação em névoa. A arquitetura proposta é baseada em *containers Linux* (LXC[2]) e abrange diversos componentes de uma estrutura de computação em névoa. Neste trabalho não é descrito como foi feita a implementação e nem experimentos baseados em dados reais.

Um dos poucos trabalhos a apresentar uma estrutura e resultados para cenários de computação em névoa voltados para IoT, o trabalho de [CHENG et al. 2017] é direcionado para o gerenciamento de contexto, ou seja, identificação das situações que ocorrem no momento e determinação das medidas a serem tomadas pelo orquestrador de computação em névoa. Focado na parte de gerenciamento de contextos, não especifica o modo do qual uma estrutura de computação em névoa deve ser montada num cenário real.

Nos trabalhos relacionados mencionados nesta seção, são apresentadas várias ideias sobre como montar uma estrutura de computação em névoa, sugestões de arquiteturas e gerenciamento de contexto. Porém, em geral são voltadas a serviços específicos e muitas vezes sem apresentar resultados e modelos de utilização baseados em cenários e dados reais. Este se diferencia por criar um cenário baseado em dados reais em laboratório, que sirva de referência para qualquer cenário, efetuando experimentos e observando os resultados do cenário mostrado nas próximas seções.

## 3 Modelo para Implantação de Computação em Névoa

Embora exista uma arquitetura de referência lançada recentemente pelo *OpenFog Consortium*[3], não há no momento um consenso sobre um modelo efetivo para implementação real de uma estrutura para computação em névoa, tampouco um modelo genérico ou definido para determinados casos de uso. A Figura 1 ilustra o modelo proposto e utilizado na implementação do cenário de computação em névoa.

Os dados são gerados na camada de IoT, com destino à nuvem ou ao nó névoa, dependendo do caso de uso. Os equipamentos desde a rede de acesso até o *datacenter* que possuem compatibilidade com SDN podem ser utilizados para colaborar com a estrutura, tanto modificando a estrutura lógica da rede quanto recebendo novas funções de rede virtualizadas (VNF) (por exemplo, um ponto de acesso Wi-Fi que pode receber uma função de rede de NAT ou cache Web). Na camada névoa (*Fog Layer*) estão todos os equipamentos que podem atuar como nó névoa. Estes equipamentos devem possuir um *software* que permita a comunicação com o orquestrador de névoa (*Fog Orchestrator*), além de um *hypervisor*, que permitirá múltiplas funções de rede, independente da aplicação, funcionarem no mesmo *hardware*.

[2] https://linuxcontainers.org
[3] https://www.openfogconsortium.org

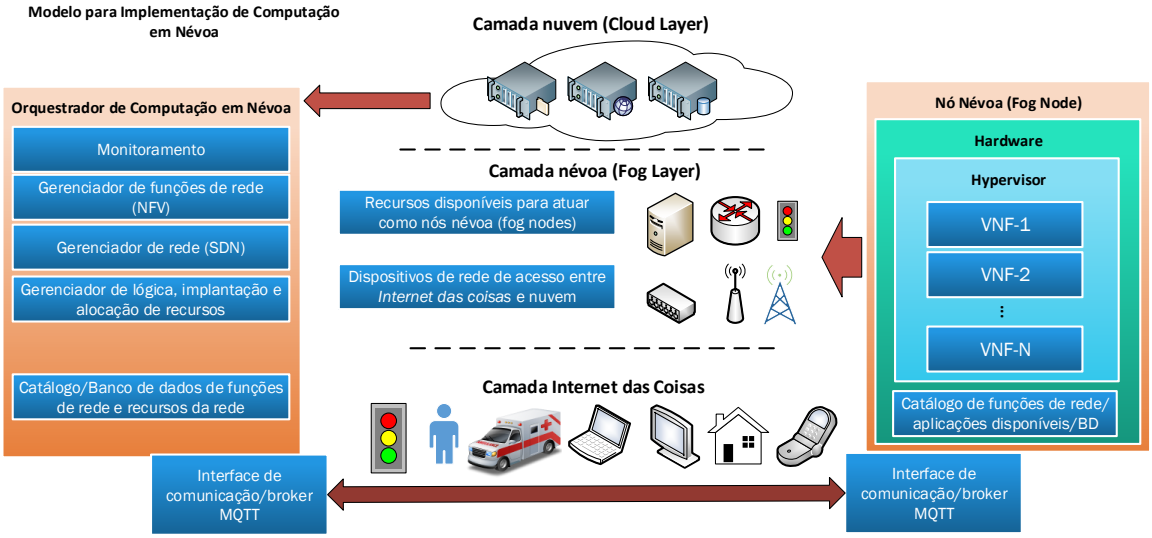



**Figura 1. Modelo de computação em névoa na Internet das Coisas**

A Figura 2 mostra exemplos de serviços em um nó névoa, em que duas aplicações distintas devem ser adicionadas ao catálogo de funções de rede e aplicações do nó névoa. O catálogo de funções de rede e aplicações é uma base de dados ou espaço de armazenamento que mantém funções de rede que podem ser alocadas naquele nó névoa. Note que uma aplicação em Java, por exemplo, necessita da *Java Virtual Machine* (JVM) para funcionar, porém não é nada prático nem eficiente disponibilizar todos pré-requisitos dos clientes no nó névoa. Assim espera-se que a imagem da função de rede ou aplicação já contenha todos os requisitos necessários para o funcionamento da aplicação. Outro exemplo, o *firewall* IPTables, utiliza funções do *kernel* do Linux. Mesmo sendo compatível com o sistema operacional do nó névoa, não é interessante que um serviço acesse diretamente o *kernel* do sistema hospedeiro do nó névoa e ainda utilize seus recursos compartilhados, portanto novamente este processo é isolado na máquina virtual. Deste modo, o fornecedor da função de rede não precisa e nem deve saber o tipo de *hardware* e sistema operacional executado no nó névoa, assim como não importa para o nó névoa o que há dentro da imagem de função de rede ou aplicação, desde que se saiba o mínimo necessário para alocação do serviço.

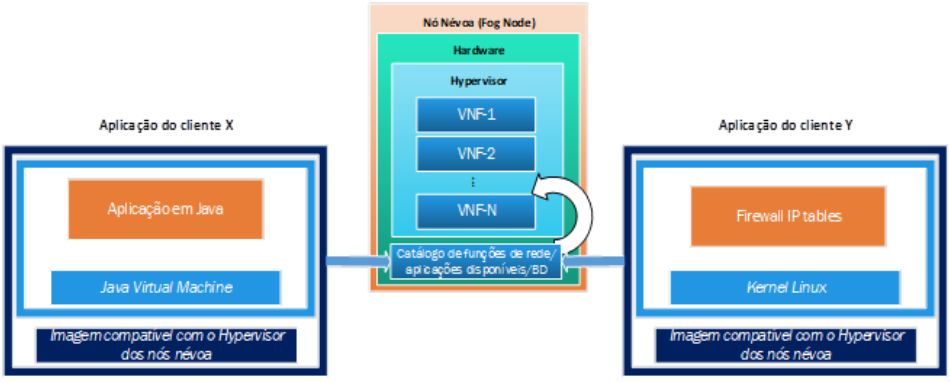



**Figura 2. Exemplo de serviços em um nó névoa.**

Na camada nuvem (*cloud layer*) o orquestrador de névoa controla toda a estrutura, em que é responsável pela alocação de recursos e redirecionamento de tráfego.

Adota-se então uma divisão por módulos do orquestrador de Computação em Névoa, de forma a permitir maior programabilidade e independência entre as partes do sistema, atuando de forma independentemente, conforme sugerido na Figura 3:

- Catálogo de funções de rede: presente também nos nós névoa quando viável, é o local ou banco de dados que contém as funções de rede disponíveis para utilização. Inserir ou não uma função de rede previamente no catálogo do nó névoa vai depender de alguns fatores como se é necessário a ativação rápida da função ou se a imagem da função de rede é pequena o suficiente para permitir seu transporte de forma eficiente toda vez que for necessária sua ativação.

- Módulo de Monitoramento: monitora a estrutura, verificando o tráfego, latência e utilização da rede, além de obter informações dos recursos disponíveis, tanto de rede para SDN quanto de *hardware* para alocação de novos nós névoa e manter uma visão geral de toda a rede envolvida na estrutura. Todos os dados recebidos pela interface de comunicação vão para o módulo de monitoramento, e este módulo vai efetuar a entrega dos dados pertinentes a cada cliente, de acordo com as regras pré-estabelecidas. Este módulo deve monitorar também internamente se alguma das funções de rede do catálogo de funções de rede necessita ser alocada. Isto pode ocorrer por exemplo em funções de rede que operam por sazonalidade, em casos que necessitam ser ativadas em determinadas horas ou períodos específicos. Este módulo pode ainda receber informações de meios externos, como por exemplo, um cliente que necessita de Computação em Névoa para reduzir a demanda em seus servidores da nuvem, e informa ao Orquestrador de Névoa esta condição, para que sejam tomadas providências.

- Gerenciador de lógica, implantação e alocação de recursos: contém as regras do negócio, ou seja, ele vai tomar as decisões de acordo com a necessidade do cliente, baseado nas informações disponibilizadas pelo módulo de monitoramento e o catálogo de funções de rede. Neste módulo se encaixa o componente conhecido como gerenciador de contexto [CHENG et al., 2017], que automatiza a tomada de decisões. Quando o módulo de monitoramento identifica uma mudança na estrutura, ele avisa este módulo gerenciador de lógica, para que então seja feita a análise das informações e tomada de decisões, acionando então os outros módulos de comando conforme a necessidade.

- Gerenciador de funções de rede (VNFs): controla as funções de rede disponíveis e seus parâmetros, ou seja, controla a possibilidade de alocar determinada função de rede e sua prioridade, tipo de serviço e recursos necessários para execução, dados os requisitos estabelecidos e as instruções oriundas do módulo gerenciador de lógica. O gerenciador de funções de rede vai se comunicar enviando comandos e instruções para os virtualizadores existentes em cada névoa, além de enviar, alterar e remover as imagens do catálogo de funções de rede localizado na nuvem e nas névoas.

- Gerenciador de rede: controla a estrutura da rede e os equipamentos disponíveis para SDN. Neste caso foi utilizado o *Floodlight*[4] como controlador SDN, ou seja, o *software* que fará a comunicação com os dispositivos SDN através do protocolo *OpenFlow*. Este módulo recebe informações e instruções dos módulos de monitoramento e gerenciamento de lógica, para então efetuar a inclusão ou remoção de fluxos SDN, que vão alterar a estrutura lógica da rede conforme necessário e se possível. Neste módulo deve constar os equipamentos da estrutura que são compatíveis com *OpenFlow*, porém nada impede que novos equipamentos sejam cadastrados sob demanda, por isso é importante o conhecimento da topologia de rede e atualizações constantes de possíveis mudanças, conforme mencionado no módulo de monitoramento.

Foi desenvolvido o Orquestrador de Névoa utilizando linguagem Java e bibliotecas para o protocolo MQTT (biblioteca MQTT para Java *paho*[5]), baseado no modelo proposto nesta seção.

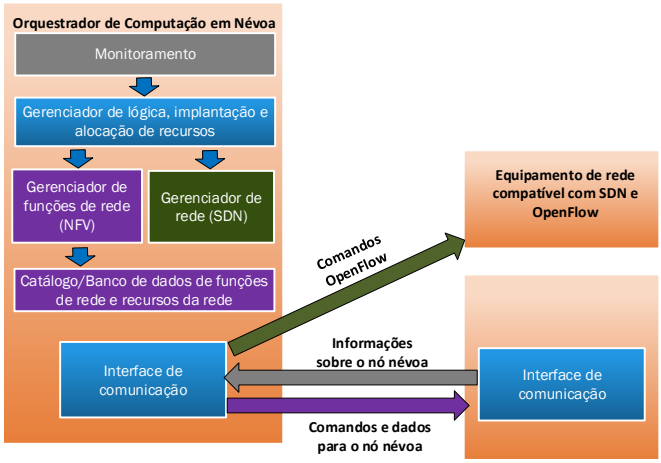



**Figura 3. Organização do Orquestrador de Computação em Névoa.**

A seguir, é descrita a metodologia e casos de uso adotados para criação do cenário de motivação em laboratório, seguindo o modelo apresentado nesta seção.

## 4 Metodologia e Casos de Uso

A metodologia utilizada neste artigo é baseada em estudos teóricos do estado da arte, uma vez que o tema conta com pouca aplicação prática até o momento. Com as questões levantadas pelos muitos trabalhos teóricos, obtivemos as bases para realização dos experimentos, coletas dos dados e estabelecimento de prioridades para as tarefas mais relevantes. O cenário aqui descrito foi implementado em um ambiente de testes em laboratório (*testbed*) e uma avaliação de desempenho foi realizada para aumentar a percepção sobre a viabilidade do uso de computação em névoa para cenários envolvendo IoT em cidades inteligentes.

[4] http://www.projectfloodlight.org/floodlight.
[5] https://www.eclipse.org/paho.

Foram definidos dois casos de uso para realização de experimentos. Um caso baseado em sensoriamento na área de assistência médica e outro caso hipotético onde automóveis em centros urbanos possam, com a ajuda de computação em névoa, realizar frenagem automática, com semáforos virtuais. Estes casos de uso serão simulados em laboratório utilizando-se de máquinas e dados reais, extraídos de outros artigos e pesquisas de órgãos do governo. Os dois casos de uso são representados simultaneamente para especificar uma situação real. A Figura 4 ilustra a situação proposta, onde é considerado um cruzamento de duas vias de grande circulação de veículos, que representa o cenário de semáforos virtuais. Próximo dali, há uma praça com pessoas que portam sensores diversos para monitoramento de saúde.

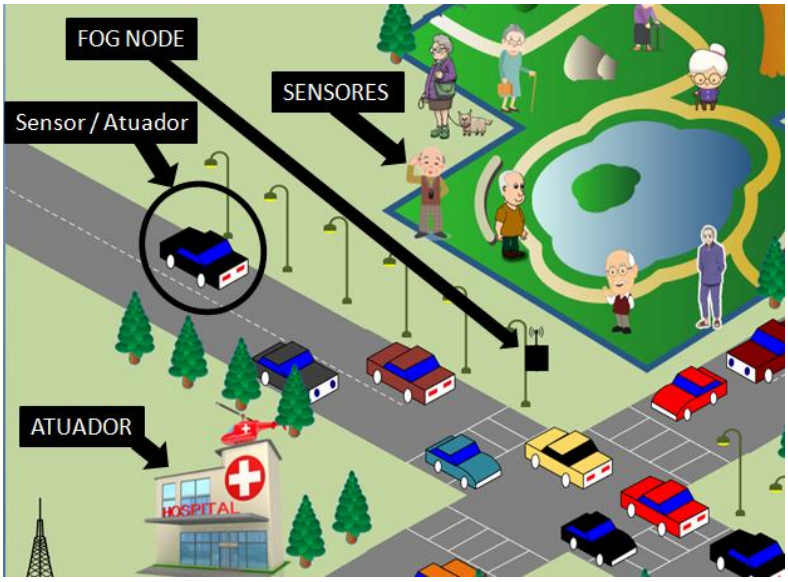



**Figura 4. Cenários de uso: assistência médica e semáforos virtuais**

Os veículos representam o caso de uso de semáforos virtuais e as pessoas na praça próxima ao cruzamento representam o caso de uso de assistência médica. No centro do cruzamento há um equipamento, o nó névoa, que é o responsável por executar o processamento das informações oriundas dos dois casos. Este nó névoa é controlado pela nuvem, mais precisamente pelo orquestrador de computação em névoa (seção 3). Todas mensagens trocadas pelos sensores, atuadores e servidores utilizam o protocolo de aplicação MQTT.

### 4.1 Cenário de Assistência médica

O cenário de assistência médica representa uma situação onde pessoas diversas são constantemente monitoradas por sensores espalhados pelo corpo, e a todo momento precisam enviar informações à central de monitoramento. Em caso de alguma anomalia ser detectada, um atuador deve ser acionado, que neste caso será um alarme em um hospital ou posto de saúde próximo ao local do evento.

Os dados utilizados para gerar o tráfego deste cenário foram obtidos a partir dos trabalhos de [GIA et al. 2015] e [MOHAPATRA et al. 2012]. Conforme estes trabalhos citados, um paciente que tem monitorados a pressão arterial, batimentos cardíacos, temperatura corporal, glicose, insulina, localização, sensores de movimento (para detectar quedas em idosos por exemplo) e eletrocardiograma, gera em média 30 KB de dados por

segundo (sendo grande parte destes dados gerados pelo eletrocardiograma [DUBEY et al. 2015]). Portanto, cada paciente, representado por um *thread* no sistema gerador de dados, transmite dados em média a uma taxa de 240 Kbps.

Neste cenário, os dados são constantemente enviados para a nuvem, onde está localizado o servidor central do sistema. Quando disponível, um nó névoa efetua o trabalho da nuvem, ou seja, recebe os dados, processa, armazena e caso necessário envia um alerta ao atuador. É um cenário onde o volume de dados é grande, porém não necessita de respostas em tempo real, embora também não possa permitir grandes atrasos nos alertas (alguns milissegundos de atraso são aceitáveis). Apesar da quantidade de dados ser significativa, o processamento é reduzido para este caso, visto que basta uma verificação simples se os valores estão acima (ou abaixo) de determinados limites, para então acionar os atuadores, pois, embora o eletrocardiograma crie uma quantidade grande de dados, o processamento destes dados não é alto. Para aproximar o cenário da realidade e devido à falta de dados reais, foi definido que a cada 50 mensagens recebidas é acionado um atuador.

### 4.2 Cenário de Semáforos Virtuais

No cenário de semáforos virtuais, em cada cruzamento nós névoa substituem semáforos convencionais, ou seja, equipamentos que podem atuar com diversas funções de rede virtualizadas. Cada veículo (representado por um *thread* no sistema gerador de tráfego) que passa pelo local está constantemente enviando dados para o nó névoa mais próximo. Este nó névoa executa então um cálculo a partir dos dados enviados e retorna um aviso para o veículo, informando este que ele precisa iniciar o processo de frenagem. Conclui-se, portanto, que neste cenário, tanto o sensor como o atuador estão no mesmo dispositivo, que é o veículo.

Cada veículo envia ao nó névoa dados de velocidade atual e coordenadas geográficas (latitude e longitude). O nó névoa recebe os dados e efetua um cálculo para obter a distância que o veículo está do cruzamento (cálculo de *Haversine*[6]), para então calcular o tempo que resta para que o veículo chegue ao cruzamento, alertando-o caso necessário para que ele inicie o processo de frenagem. A frequência de envio dos dados e quantidade de veículos foi baseada em relatório da Companhia de Engenharia de Tráfego da cidade de São Paulo[7], onde, na Avenida Paulista, passam, em média, 6884 veículos por hora no momento de pico. Um atuador é acionado cada vez que o cálculo resultante dos dados recebidos indique que o veículo está a menos de 10 segundos de chegar ao cruzamento (considerando a velocidade constante).

Este caso de uso terá a função de condicional para que o caso de uso de assistência médica seja ativado nos nós névoa, ou seja, quando a utilização de recursos pelos semáforos virtuais caírem, inicia-se a utilização destes recursos pelo caso de assistência médica.

A Tabela 1 apresenta um resumo das principais características de cada cenário.

[6] http://rosettacode.org/wiki/Haversine_formula
[7] http://www.cetsp.com.br/media/115063/btcetsp48.pdf

**Tabela 1. Resumo dos dados de cada cenário**

| Característica | Assistência médica | Semáforos Virtuais |
|---|---|---|
| Carga útil (*payload*) | Alta: média 30KB por paciente | Baixa: média 55 *Bytes* por veículo |
| Processamento | Baixo: apenas verifica se os valores estão abaixo ou acima de um determinado limite | Médio: cálculos de média complexidade, pesados em equipamentos com baixo poder de processamento |
| Atuador | Hospitais ou postos de saúde próximos do local do alerta | Os próprios veículos são os atuadores, recebendo dados do processamento |
| Destino dos dados | Nuvem | Nó Névoa |
| Tráfego na WAN | Alto | Baixo |
| Latência desejada | Não necessita tempo real | Precisa ser em tempo real. |

### 4.3 Implementação

Todas os servidores utilizados para implementação dos cenários executam o sistema operacional Linux Debian 8 (Jessie). A comunicação entre os aplicativos desenvolvidos em Java nas máquinas é realizada pelo protocolo MQTT com nível de QoS 2, uma vez que este nível garante a entrega das mensagens e não envia informações duplicadas, o que é importante para dados de assistência médica, evitando alertas falsos e informações inconsistentes sobre a saúde dos pacientes. Para simular as conexões de rede de acesso e a Internet, foi utilizado o simulador de redes WanEm[8].

- Gerador de tráfego: Servidor responsável por gerar o tráfego que caracteriza os cenários propostos. Os dados partem com destino à nuvem no caso de uso de assistência médica e com destino ao nó névoa no caso de uso de semáforos virtuais. O algoritmo gerador de tráfego foi construído baseado no trabalho de [ZYRIANOFF et al. 2017].
- Nuvem: Servidor que representa a nuvem e executa o orquestrador de névoa desenvolvido em Java que trabalha com o protocolo de aplicação MQTT. Nesta máquina executam também um *Broker* MQTT, que recebe as informações dos sensores (gerador de tráfego) e efetua o processamento de acordo com o caso de uso. Após o processamento, dependendo do resultado pode-se acionar ou não um atuador.
- Nó Névoa: Servidor representando o nó névoa (*fog node*), que no cenário proposto é um equipamento localizado na esquina de duas vias públicas de grande circulação de veículos e pessoas. Possui um *broker* MQTT e um virtualizador (*hypervisor* - KVM[9]). As VNFs são criadas de acordo com os comandos oriundos do orquestrador de névoa localizado na nuvem.
- WanEm Local: Servidor Intel Core 2 Duo, 4GB RAM, que simula uma conexão Wi-Fi na estrutura montada. Os parâmetros, baseados nos valores das companhias

[8] http://wanem.sourceforge.net
[9] https://www.linux-kvm.org/page/Main_Page

OpenSignal[10] e SpeedGuide.net[11], são latência de 30ms, jitter de 5ms, taxa de perda de pacotes de 5% e largura de banda de 30 Mbps.

- WanEm WAN: Servidor Intel Core 2 Duo, 4GB RAM, que simula uma conexão de Internet, com os seguintes parâmetros: latência de 15ms, jitter de 2ms, taxa de perda de pacotes de 1%. Esses valores representam aproximadamente o equivalente à distância de São Paulo ao Rio de Janeiro, baseado no simulador da companhia WintelGuy[12] e no trabalho de KOMOSNY et al. [2015].

Os experimentos de avaliação de desempenho foram executados de acordo com os fatores e níveis descritos na Tabela 2.

**Tabela 2. Fatores e níveis da avaliação de desempenho.**

| Fator | Nível |
|---|---|
| Requisições por segundo (assistência médica), média | 400 |
| Requisições por segundo (semáforos virtuais), média | 4 (variando de 2 a 4) |
| Parâmetros simulador de rede de acesso (Wi-Fi) | Latência: 30ms, Jitter: 5ms, Perda: 5%, Banda: 30Mb/s |
| Parâmetros simulador de Internet | Latência: 15ms, Jitter: 2ms, Perda: 1% |
| Carga útil de cada mensagem (ass. médica), média | 30KB |
| Carga útil de cada mensagem (sem. virtuais), média | 55 Bytes |
| Quantidade de Nós Névoa | 1,2 e 4 |

O fluxo de dados ocorre da seguinte forma:

1. Os dados são gerados em *containers* na máquina geradora de tráfego, onde para o cenário de assistência médica ocorre da seguinte forma: Cada container gera 10 *threads*, ou seja, representando 10 pacientes, totalizando 400, que enviam, cada um, em média 30KB de dados por segundo com destino ao *Broker* na nuvem. Os dados do cenário de semáforos virtuais são gerados da seguinte forma: uma aplicação Java cria *threads* que vão enviar mensagens MQTT com destino ao nó névoa. Estas mensagens possuem em média 55 Bytes de carga e são geradas seguindo uma distribuição de *Poisson* com média $\lambda = 4$, variando gradativamente após certo tempo para $\lambda = 2$, com o objetivo de diminuir a carga no nó névoa, abrindo recursos para alocação de outras funções de rede.
2. O Orquestrador de Névoa está constantemente monitorando o tráfego que chega à nuvem e ao nó névoa, e quando é detectada uma disponibilidade de recursos no nó névoa, ele envia a este instruções para alocar uma função de rede que vai atender ao serviço de assistência médica. Através do controlador de SDN, são enviados comandos para criar fluxos SDN no switch virtual, direcionando parte do tráfego de forma transparente para o nó névoa.
3. Os dados recebidos pela nuvem (no caso de uso assistência médica) e pelo nó névoa (no caso de uso semáforos virtuais) são processados e dependendo do resultado, pode-se acionar ou não um atuador que está próximo ao cliente (assistência médica) ou a própria origem da informação (semáforos virtuais).

[10] https://opensignal.com
[11] https://www.speedguide.net/faq/what-is-the-actual-real-life-speed-of-wireless-374
[12] http://wintelguy.com/wanlat.html

# 5 Resultados

Nesta seção serão apresentados os resultados da avaliação de desempenho. Estes resultados têm como objetivo mostrar que uma implementação seguindo o modelo proposto é viável. Foram efetuadas 30 repetições para cada experimento desta seção. Os gráficos apresentam barras de erro representando o nível de confiança em 95%.

A Figura 5 mostra a série temporal do experimento feito em laboratório, relativa a quantidade de requisições atendidas ao longo do tempo pela nuvem e por um nó névoa, para os dois casos de uso simultaneamente. Conforme mencionado na seção 4.3, quando as requisições dos semáforos virtuais diminuem, inicia-se o redirecionamento dos dados de assistência médica para a névoa, até que metade dos dados gerados nos clientes são redirecionados para a névoa, através de fluxos SDN. No momento em que as requisições dos semáforos virtuais começam a subir novamente, os fluxos SDN começam a ser removidos, liberando recursos do nó névoa e voltando o tráfego novamente direto para a nuvem, num processo transparente aos clientes. Estas decisões sobre quando direcionar o tráfego são configuradas no orquestrador de névoa, conforme mencionado na seção 3.

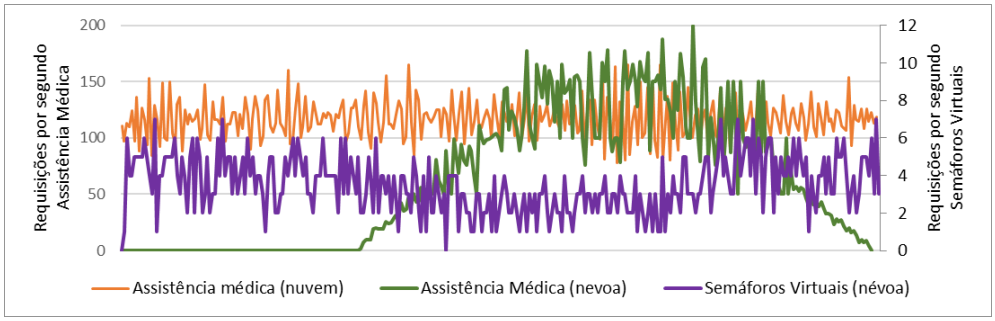



**Figura 5. Número de requisições atendidas pela nuvem e pela névoa**

No gráfico da Figura 6 é mostrada a média de tempo para as mensagens sair do cliente (paciente) e chegar ao *broker* tanto na nuvem quanto na névoa no cenário de assistência médica. Apesar de utilizar a mesma rede de acesso, apenas o fato de remover a necessidade de transmitir informação pela Internet já mostrou melhora nos números. Observa-se que conforme adicionamos mais nós névoa, e consequentemente dividimos a quantidade de requisições entre eles, ocorre melhora nos tempos de resposta, conforme era esperado.

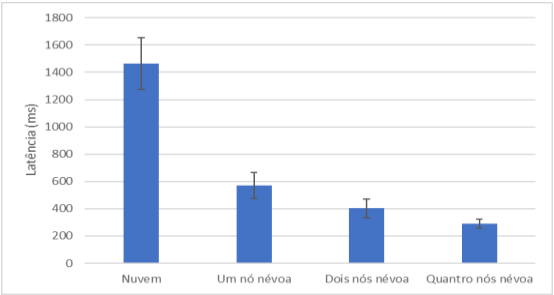
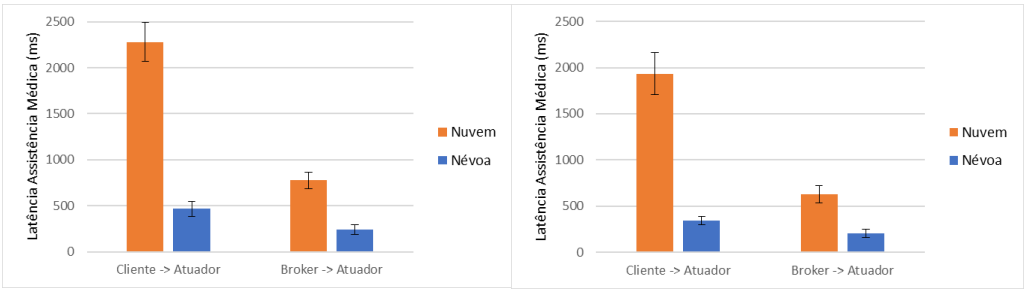



**Figura 6. Atrasos de tráfego de dados**

Na Figura 7 temos os resultados dos tempos medidos nos atuadores, para dois nós névoa (a) e para quatro nós névoa (b). Consequente da diminuição de tempo de resposta mostrado anteriormente, os tempos de acionamento aos atuadores também diminuíram.




**a) Dois nós névoa** **b) Quatro nós névoa**

**Figura 7. Tempo de acionamento dos atuadores: assistência médica**

A Figura 8 mostra a distribuição de frequência acumulada para as respostas em menos de determinados milissegundos (eixo x do gráfico). Fica evidente a melhora proporcionada pela alocação de nós névoa em relação à nuvem.

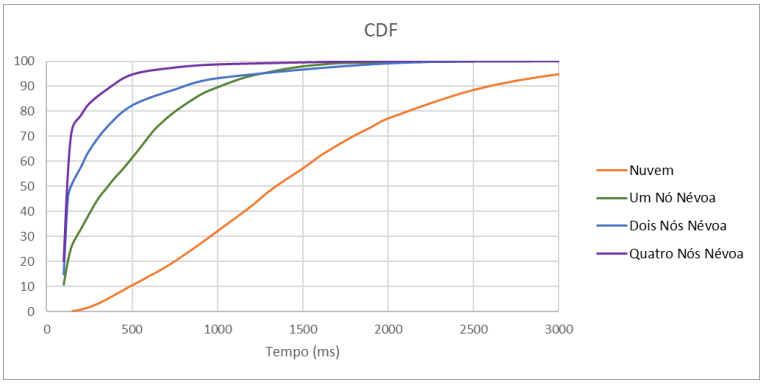



**Figura 8. CDF para o tempo de atendimento das requisições.**

Nestes resultados, observa-se que as velocidades das redes de acesso e WAN tem influência significativa no desempenho do sistema como um todo, que em muitos casos pode determinar a viabilidade da utilização de computação em névoa. A combinação de fatores diversos como carga dos dados e quantidade de requisições por segundo gera melhora nos tempos de resposta, mostrando que a aplicação do modelo proposto é viável e não causa prejuízo à estrutura existente. Para o caso de uso utilizado, de assistência médica, talvez não seja interessante alocar mais recursos (mais que dois nós névoa) para obter-se uma melhora de alguns milissegundos nos tempos de respostas. Porém em um caso de uso que exige respostas mais rápidas, como monitoramento de máquinas industriais ou jogos online por exemplo, esta diferença é significativa.

## 6 Conclusões

Neste artigo é apresentado um modelo para implantação de infraestruturas compatíveis com computação em névoa, mostrando cada etapa e descrevendo como foi aplicado o modelo em laboratório, sendo que este pode ser perfeitamente implantado em cenários reais, da forma mais genérica possível e utilizando equipamentos legados e em muitos casos já disponíveis na própria infraestrutura presente nos locais envolvidos.

Os resultados indicam situações onde se mostrou viável a implantação de computação em névoa, sendo que os casos de uso aqui apresentados podem ser adaptáveis a outros casos de uso de fatores equivalentes. Assim, espera-se contribuir para futuras

pesquisas e desenvolvimento de *softwares* voltados para atender casos de uso que necessitem de computação em névoa, além de prover uma base e um ponto de partida para trabalhos futuros neste campo, onde ainda há muito o que explorar.

Como trabalhos futuros, baseando-se no modelo apresentado aqui para implementações, muitos experimentos podem ser efetuados para medições em condições e cargas de trabalho diversas, além de campos pouco trabalhados até o momento, como consumo de energia e segurança da informação em névoa.

## Referências


Bonomi, F., Milito, R., Zhu, J. e Addepalli,S. (2012) "Fog Computing and Its Role in the Internet of Things", MCC workshop on Mobile cloud computing, Agosto 2017.

Cheng, B., Solmaz, G., Cirillo, F., Kovacs, E., Terasawa, K. e Kitazawa, A. (2017) "FogFlow: Easy Programming of IoT Services Over Cloud and Edges for Smart Cities". IEEE Internet of Things Journal Volume: PP, Issue: 99.

CISCO. (2017) "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016–2021 White Paper". https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html. Acesso em 11/2017.

Dubey, H., Yang, J., Constant, N., Amiri, A.M., Yang, Q. e Makodiya, K. (2015) "Fog Data: Enhancing Telehealth Big Data Through Fog Computing". Proceedings of the ASE BigData & SocialInformatics. P14:1-14:6.

ETSI. (2013) "Network Functions Virtualisation (NFV): Use Cases". IEEE Network. Disponível em: http://www.etsi.org/deliver/etsi_gs/NFV/001_099/001/ 01.01.01_60 /gs_NFV001v010101p.pdf. Acesso em: 11/2017.

Gia, T. N., Jiang, M., Rahmani, A., Westerlund, T., Liljeberg, P. e Tenhunem, H. (2015) "Fog Computing in Healthcare Internet of Things: A Case Study on ECG Feature Extraction" 15th IEEE International Conference on Computer and Information Technology. P356-363.

Kamienski, C., Biondi, G., Borelli, F., Heideker, A., Ratusznei, J., Kleinschmidt, J., "Computação Urbana: Tecnologias e Aplicações para Cidades Inteligentes", Minicursos SBRC 2016, Maio de 2016.

Komosny, D., Voznak, M., Ganeshan, K. e Sathu, H. (2015) "Estimation of internet node location by latency measurements – The underestimation problem". Information Technology and Control, vol. 44, p279-286.

Mckeon, N., Anderson, T., Peterson, L., Rexford, J., Shenker, S. e Louis, S. (2008) "Openflow: Enabling Innovation in Campus Networks". ACM SIGCOMM Computer Communication Review. ACM. p69-74.

Mohapatra, S. e Rekha, K.S. (2012) "Sensor-Cloud: A Hybrid Framework for Remote Patient Monitoring". International Journal of Computer Applications, p7-11.

Souza, V.B.C, Ramirez, W., Masip-Bruin, X., Marín-Tordera, E., Ren, G. e Tashakor, G., (2016) "Handling Service Allocation in Combined Fog-Cloud Scenarios". IEEE International Conference on Communications, p0-4.

Yigitoglu, E., Mohamed, M., Liu, L. e Ludwig, H. (2017) "Foggy: A Framework for Continuous Automated IoT Application Deployment in Fog Computing". IEEE International Conference on AI & Mobile Services. p38-45.

Zhu, J., Chan, D. S., Prabhu, M. S., Natarajan, P., Hu, H. e Bonomi, F. (2013) "Improving Web Sites Performance Using Edge Servers in Fog Computing Architecture". IEEE Seventh International Symposium on Service-Oriented Sys Engineering. P320-323.

Zyrianoff, I., Borelli, F. e Kamienski, C. (2017). "SenSE – Sensor Simulation Environment: Uma ferramenta para geração de tráfego IoT em larga escala". Salão de Ferramentas - Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos – SBRC.