HealthDash: Monitoramento remoto de pacientes utilizando programação baseada em fluxo de dados

Jordano R. Celestrini[1], Renato N. Rocha[1], Celso A. S. Santos[1], Vinícius F. S. Mota[1], José G. Pereira Filho[1], Rodrigo V. Andreão[2]

[1]Departamento de Informática – Universidade Federal do Espírito Santo (UFES)
Caixa Postal 01-9011 – 29060-970 – Vitória – ES – Brasil

[2]Departamento de Engenharia Elétrica
Instituto Federal do Espírito Santo (IFES) – Vitória, ES – Brasil

{jordanorc, rocha.nolasco}@gmail.com
{saibel, zegonc, vinicius.mota}@inf.ufes.br
rodrigova@ifes.edu.br

Abstract. *In this paper, we propose HealthDash, a framework for developing IoT solutions for health care. HealthDash employs the data-flow-oriented programming paradigm, from the cloud layer to the network edge (fog layer), unifying development technologies across layers, that is, from edge devices to decision making. We conducted an experiment to evaluate the proposal with the simulation of the transmission of data collected from home-monitored patients with chronic diseases. In the simulation, we observed the performance of the two implemented solutions to both continuous and event-based scenarios of data transmission. The results showed that HealthDash solution provides flexible infrastructure, consuming less bandwidth and spending little response time.*

Resumo. *Neste artigo, propomos o HealthDash, um arcabouço para o desenvolvimento de soluções IoT para saúde. O HealthDash emprega o paradigma de programação orientado a fluxo de dados, desde camada cloud à borda da rede (camada fog), unificando tecnologias de desenvolvimento em todas as camadas, isto é, dos dispositivos de borda à tomada de decisão. Para validar a proposta, foi conduzido um experimento que simulou a transmissão de dados de pacientes crônicos monitorados em domicílio, considerando dois modos de envio: transmissão contínua de dados e transmissão de dados baseada em eventos. O experimento mostrou que a solução proposta fornece infraestrutura flexível, consumindo menos largura de banda e menos tempo de resposta.*

1. Introdução

Melhorar a qualidade de vida de pacientes com doenças crônicas e reduzir os custos com saúde são metas perseguidas pelos sistemas de saúde em todo mundo. Uma das soluções mais conhecidas para este fim se baseia no monitoramento remoto dos pacientes, que possibilita o acompanhamento da condição de saúde do indivíduo e a conformidade com o tratamento fora do ambiente hospitalar. Todavia, os procedimentos para o monitoramento, o cuidado e a supervisão de pacientes com doenças crônicas são frequentemente executados por uma equipe de saúde especializada, mas ainda de forma pouco automatizada.

Recentemente, o conceito de Internet das Coisas (*Internet of Things* - IoT) estimulou o desenvolvimento de soluções para apoiar ao monitoramento remoto destes pacientes [Farahani et al. 2018]. Uma destas soluções propõe virtualizar as "coisas" (*things*) e usar serviços em *cloud*[1], oferecendo uma abstração bem definida de dispositivos e uma estrutura de programação comum para facilitar o processo de desenvolvimento. Entretanto, essa abordagem não é adequada para diversos aplicativos da IoT que requerem processamento mais próximo do dispositivo e um acoplamento mais estreito entre eventos e ações [Giang et al. 2015].

No cenário da computação aplicada à saúde, mais especificamente no monitoramento remoto dos pacientes em domicílio, o emprego da *cloud* traz consigo problemas como tempo de resposta, quantidade de dados que são trafegados na rede e também questões relacionadas a privacidade e segurança. No monitoramento de paciente cardíacos, por exemplo, o tempo de resposta é essencial para preservar a vida e, neste sentido, a indisponibilidade de comunicação com a *cloud* pode representar sérios danos à saúde do paciente. Do mesmo modo, o envio de dados sensíveis, como vídeos e imagens do ambiente monitorado e do paciente, pode gerar sobrecarga na rede, além de aumentar os riscos envolvendo quebra de privacidade de dados do paciente.

A computação *fog* busca preencher a lacuna entre a *cloud* e os dispositivos da IoT e resolver alguns dos problemas da integração destes dois mundos. Ao contrário do que acontece na *cloud*, a computação *fog* procura distribuir os recursos computacionais, se aproveitando do poder de processamento e de armazenamento dos dispositivos na borda da rede. Assim, é possível ampliar a capacidade computacional e o armazenamento da *cloud* para as camadas de acesso da rede, permitindo que os dados sejam analisados e transformados em informações ou em ações antes de serem simplesmente transmitidos. Embora este modelo apresente benefícios, ele tende a aumentar a complexidade no desenvolvimento das soluções, uma vez que novos elementos tecnológicos são adicionados à arquitetura, aumentando a multiplicidade das tecnologias adotadas, ao passo em que cada camada implementa suas próprias APIs, sistemas operacionais e linguagens de programação. Do ponto de vista dos desenvolvedores, quanto mais heterogêneo o ambiente tecnológico, maior será o esforço para implementação e manutenção das soluções.

Neste sentido, utilizar uma tecnologia que se propague desde a camada *cloud* até a camada *fog*, ajuda a simplificar o processo de desenvolvimento, propiciando a reusabilidade e extensibilidade dos componentes. Assim, este artigo propõe uma solução para o desenvolvimento de aplicações IoT para saúde denominada HealthDash, baseada no paradigma de programação orientado a fluxo de dados, permitindo indicar em qual camada da arquitetura (*fog* ou *cloud*) os fluxos de dados, que processam dados dos pacientes monitorados, serão executados.

Este artigo está estruturado da seguinte forma: A seção 2 introduz a computação *fog* e *cloud* no contexto da saúde. A seção 3 apresenta alguns trabalhos relacionados à pesquisa desenvolvida neste trabalho, e a seção 4 retrata o paradigma de programação orientada a fluxo de dados. A seção 5 descreve a arquitetura HealthDash e a seção 6 traz a avaliação da solução e a análise dos resultados obtidos. A seção 7 conclui o artigo e oferece direções futuras para a continuidade do trabalho.

[1] Ao longo do texto, iremos nos referir ao modelo de computação em nuvem apenas como *cloud*.

2. Computação *cloud* e *fog* no contexto da saúde

O modelo de computação em nuvem (*cloud computing*), foi concebido para viabilizar serviços de baixo custo, fácil acesso e com garantia de disponibilidade e escalabilidade. Devido ao seu vasto poder computacional e ampla capacidade de armazenamento, o modelo em nuvem é capaz de lidar com grande quantidade de dados gerados pelos sistemas IoT. Por outro lado, a *cloud* possui algumas limitações conhecidas, tais como a necessidade de conectividade de banda larga no cliente, a dificuldade de garantir baixa latência para as aplicações, além de questões relacionadas à segurança e privacidade das informações manipuladas no modelo.

Soluções IoT para computação aplicada à saúde podem ser bastante impactadas pelas limitações anteriores. Uma das estratégias para superar tais limitações é utilizar uma camada denominada "*fog*", composta por dispositivos situados entre a *cloud* e o usuário. Estes dispositivos estão fisicamente mais próximos da fonte de dados e são responsáveis para processar as solicitações de serviço na borda da rede. A computação *fog* possibilita processar requisições da rede para próximo às redes subjacentes, funcionando como uma ponte entre os dispositivos e a *cloud* [Aazam and Huh 2015].

Diversos serviços de saúde, principalmente àqueles de missão crítica e sensíveis à latência, demandam o menor tempo de resposta e de processamento possível, inviabilizando a comunicação com a *cloud*. Neste caso, a computação *fog* desempenha um papel muito importante, uma vez que permite que os dados sejam processados na borda da rede. Um exemplo é a detecção de arritmias cardíacas, que devem ser notificadas imediatamente para que uma ação seja tomada. Assim, aguardar o processamento dos dados na *cloud* pode representar um sério risco à saúde do paciente, uma vez que podem ocorrer atrasos na comunicação.

A camada *fog* também pode ser empregada para pré-processar dados brutos, transferindo para a *cloud* apenas as informações relevantes, aprimorando a qualidade dos dados transmitidos e diminuindo o uso de banda. Como exemplo, um serviço para detecção de quedas de pacientes baseado em sensores inerciais, como giroscópio e acelerômetro pode processar os dados na camada *fog*, enviando para a *cloud* somente a informação relevante (neste caso, a notificação do evento de queda).

Outra vantagem da abordagem *fog* é a garantia de privacidade dos dados. Enquanto hospedar os dados coletados do paciente na *cloud* significa que as informações estarão, em algum momento, disponíveis para o gestor da *cloud*, na abordagem *fog* os dados podem ser processados nas camadas controladas pelos usuários.

Com diferentes tipos de dados gerados por dispositivos heterogêneos, surgem também problemas de interoperabilidade. Neste contexto, é importante que a camada *fog* seja capaz de atender a uma variedade de dispositivos de diferentes fabricantes, modelos, sistemas operacionais e protocolos de comunicação. Para tal, faz-se necessário estabelecer uma estrutura de dados comum, a fim de que as informações sejam anotadas e validadas, o que ajuda a aprimorar os serviços oferecidos pela *cloud*.

Cenários tais como o da saúde poderiam se beneficiar da existência de uma tecnologia que promova a integração entre as diferentes camadas das soluções IoT, facilitando o gerenciamento de diferentes perspectivas de abstração, além de lidar com a heterogeneidade de tecnologias empregadas, o que promoveria facilidades para o ambiente de

desenvolvimento.

3. Trabalhos relacionados

O trabalho de [Ahmad et al. 2016] traz um *framework* para compartilhar e processar informações relacionadas à saúde com dados provenientes de diversas fontes. Os autores empregam uma camada *fog* entre a *cloud* e o usuário para evitar o fluxo desnecessário de informações e aprimorar o controle de questões relacionadas à privacidade, segurança e compartilhamento das informações. Um aplicativo para celular, que captura os dados em intervalos predefinidos de tempo, foi desenvolvido para avaliar a proposta. Depois de coletados, os dados são pré-processados e as informações enviadas para a *cloud*. Todavia, não há flexibilidade para indicar em que camada cada operação será realizada, e todas as funcionalidades foram implementadas para uma plataforma específica.

O *Mobile Fog* é um modelo de programação proposto por [Hong et al. 2013], que permite que os desenvolvedores especifiquem como o aplicativo pode ser executado em múltiplos dispositivos de computação *fog*, além de suportar a distribuição dinâmica dos recursos computacionais. Apesar de utilizar uma linguagem comum para executar as operações em todas as camadas, o modelo proposto exige conhecimento em programação para desenvolver os algoritmos. Além disso, a solução foi concebida para uma aplicação específica, não sendo genérica o suficiente para ser estendida para outros contextos.

Um *gateway* inteligente, denominado o UT-GATE, foi desenvolvido por [Rahmani et al. 2018], em uma arquitetura baseada em computação *fog*. O UT-GATE é capaz de oferecer vários recursos, como armazenamento local, processamento de dados em tempo real e mineração de dados. Todavia, as funcionalidades são implementadas isoladamente em cada uma das camadas (*fog* e *cloud)*, não sendo possível explicitar em que camada uma determinada função deve ser executada. No estudo de caso apresentado, as funções para redução de ruído em sinais ECG são implementadas diretamente no *gateway* utilizando a linguagem de programação Python. Neste caso, o local de processamento dos dados é fixo de acordo com o que foi previamente programado.

A arquitetura *iFogSim* proposta por [Gupta et al. 2017], é composta múltiplas camadas e também baseada em computação *fog*. Ela inclui uma camada de aplicação, na qual os aplicativos são modelados como uma coleção de módulos DDF (*Distributed Data Flow*). Os aplicativos podem ser implantados tanto na camada *cloud* quanto na *fog*, todavia a estratégia de implantação *fog* considera apenas o poder de processamento dos dispositivos *fog*, redirecionando para a *cloud* os dados coletados caso os dispositivos próximos à borda da rede não sejam capazes de processar a requisição. No entanto, esta estratégia não contempla as situações em que o processamento deve ser obrigatoriamente realizado na camada *fog*, seja por questões de privacidade ou de segurança, uma vez que o processo de escolha da camada é automatizado, definido de acordo com um algoritmo *edge-ward placement*.

Outros trabalhos integram os conceitos de IoT e de computação *fog* para tratar problemas ligados ao monitoramento e o acompanhamento de pacientes em domicílio. Apesar disso, alguns problemas ligados à implantação e ao desenvolvimento dos sistemas IoT para a área de saúde ainda precisam ser solucionados, em especial, a privacidade, o tempo de resposta, a segurança dos dados dos pacientes monitorados [Cerina et al. 2017, Malik et al. 2018, Gia et al. 2015]. Além disso, outras questões em

aberto se relacionam à implantação do sistema no ambiente domiciliar do paciente, que na maioria das vezes é gerenciado por um operador de saúde ou por algum familiar, ou seja, por pessoas que não habituadas ou treinadas para gerir os sistemas robustos propostos. A complexidade de desenvolvimento e de manutenção destes sistemas são questões que também são negligenciadas pela maioria dos trabalhos relacionados ao tema.

4. Programação orientada a fluxo de dados

A programação orientada a fluxo de dados é um modelo para desenvolvimento de aplicações no qual a lógica é expressa por meio de um gráfico direcionado em que os dados fluem entre as funções e passam por uma série de operações à medida que o código é executado. Cada nó (ou bloco) do fluxo é responsável por executar, de forma independente e sem interferir na execução de outros nós, uma funcionalidade própria [Sousa 2012]. Assim, um nó recebe uma informação, realiza uma transformação sobre ela e encaminha para o próximo bloco.

A Figura 1 ilustra parte do aplicativo para análise de dados de saúde, que foi utilizado nos experimentos desenvolvidos neste trabalho. O diagrama representa um fluxo para monitorar parâmetros de saúde de um paciente. Neste caso, o nó de entrada recebe as informações de pressão arterial e de glicemia via MQTT (*Message Queuing Telemetry Transport*) transforma o texto em formato JSON, classifica a leitura recebida de acordo com o tipo e realiza uma notificação para o usuário caso a leitura esteja anormal.

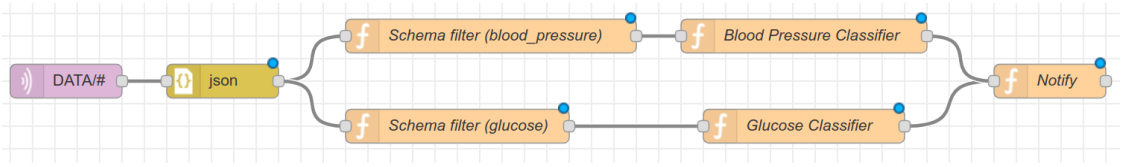

Figura 1. Exemplo de fluxo de dados para análise de dados de saúde

Uma das vantagens desta abordagem é a abstração das preocupações em desenvolver código de programação, já que toda a lógica necessária para executar determinada função está encapsulada, permitindo que o usuário se concentre em questões de mais alto nível, apesar de suportar o desenvolvimento de funções específicas, se necessário. Com o uso da abordagem baseada em fluxos, a solução do problema é quebrada numa sequência lógica de etapas, que podem ser melhor descritas.

Estas facilidades contribuem para que usuários, especialistas ou não, possam desenvolver e estender as aplicações mais rapidamente, sem exigir conhecimentos específicos em programação. Em cenários relacionados ao monitoramento remoto do paciente, esta abordagem é muito útil, uma vez que os cuidados variam de acordo com o quadro clínico de cada indivíduo, exigindo maior dinamicidade das ferramentas empregadas para o seu monitoramento.

Existem várias ferramentas que implementam o conceito de programação orientada a fluxo de dados. Dentre elas, destaca-se o Node-RED, criado e mantido pela IBM, de código aberto e cujo foco é o desenvolvimento de aplicações voltadas para IoT. No Node-RED, os nós implementam o padrão de projetos *Publisher-Subscriber* a fim de manter uma lista de assinantes (definidos por arestas direcionais) e emitir eventos para os nós adjacentes. Durante sua instanciação, os nós de entrada podem assinar serviços

externos, escutar dados em uma porta ou processar solicitações HTTP. Uma vez que os dados são processados pelo nó, seja advindos de um serviço externo ou recebido de um nó adjacente, os dados são redirecionados por meio de uma notificação.

5. A arquitetura HealthDash

A Figura 2 detalha as três camadas da arquitetura HealthDash: (1) a camada das "coisas", da qual fazem parte os dispositivos IoT; (2) a camada de computação *fog*, composta por *gateways* inteligentes que se comunicam com os dispositivos e; (3) a camada *cloud*, que inclui os servidores para recepção e gerenciamento das informações. Nesta seção, cada camada da solução e seus componentes serão apresentados e detalharemos a proposta que permite ao desenvolvedor da solução escolher em que camada será realizada a execução de fluxos de dados.

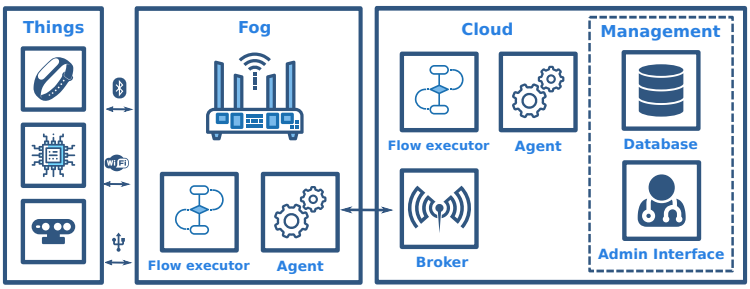

Figura 2. Visão geral da arquitetura proposta para o HealthDash.

A camada *cloud* é composta pelos componentes: (1) `Admin Interface`, responsável pelo gerenciamento de toda solução; (2) `Flow Executor`, responsável pela execução dos fluxos de dados; (3) `Database` que realiza o armazenamento dos dados coletados; (4) `Broker` encarregado pela comunicação com a camada *fog* e a recepção das informações; (5) `Agent`, que controla a alteração dos fluxos de dados.

Os componentes `Flow Executor` e `Agent` que estão camada *fog* são instâncias dos mesmos componentes utilizados na camada *cloud*. A comunicação entre as camadas *fog* e a *cloud* é bidirecional e realizada por meio de um agente que processa os comandos recebidos pela *cloud* e envia as informações capturadas nos dispositivos pela camada *fog*. Cada dispositivo na camada *fog* possui um identificador único utilizado para comunicação com a *cloud*.

O `Agent` é o responsável por controlar os fluxos de dados que são executados, tanto na camada *Fog* quanto na camada *Cloud*. Cada `Agent` possui uma identificação única, informada durante a inicialização do componente, que é utilizada como token para comunicação com o `Broker`, uma instância de servidor MQTT, protocolo utilizado para comunicação com a aplicação. O MQTT é um protocolo de mensagens leve para sensores e dispositivos pequenos, otimizado para baixo consumo de banda, comumente empregado para comunicação em soluções IoT [Farahani et al. 2018].

Após autenticar-se no `Broker`, o `Agent` inicia a escuta do tópico `$CONFIG/{{TOKEN}}`, onde `{{TOKEN}}` refere-se ao token de comunicação, utilizado como forma de autenticação. Ao receber uma nova mensagem, o `Agent` faz uma validação e, caso esteja formatada corretamente, executa uma chamada a API do `Flow Executor` que, por sua vez, atualiza o fluxo em execução na camada *fog*.

O `Flow Executor`, implementado como uma instância do Node-RED, é o componente responsável por controlar a execução do ciclo de vida dos fluxos de dados. Por meio de uma API, o `Flow Executor` comunica-se com o `Agent`, modificando o fluxo de dados sempre que há uma atualização. Cada `Flow Executor` pode executar vários fluxos, que são configurados na `Admin Interface`.

A Figura 3 exibe o diagrama de classes para o HealthDash. O pacote `data` contém as classes relacionadas a estrutura e armazenamento dos dados, incluindo o esquema (`DataSchema`), o fluxo (`DataFlow`) e os registros propriamente ditos (`Record`). As classes relacionadas ao `coaching` foram agrupadas no pacote com o mesmo nome, incluindo as informações do paciente (`Patient`) e as informações sobre o plano de cuidados (`CarePlan`) desenvolvido para cada paciente. Por fim, as classes relacionadas aos dispositivos *fog* foram atribuídas ao pacote `gateway`.

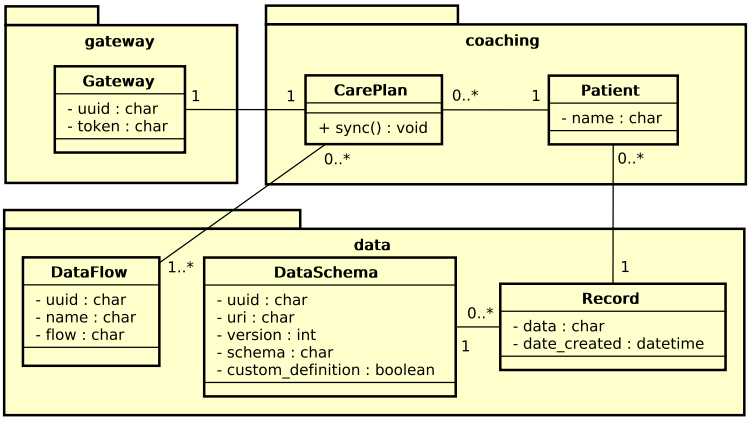

Figura 3. Diagrama de classes do `HealthDash`.

A classe `DataSchema` permite modelar os esquemas de dados que poderão ser recebidos pelo HealthDash. Cada esquema possui uma URI (*Universal Resource Identifier*) que permite identificá-la de forma única, além de um código para versionamento. Assim, sempre que houver uma atualização no esquema de dados de determinado modelo, o HealthDash poderá permitir o recebimento de dados em múltiplas versões do esquema.

Os fluxos de dados que serão utilizados nos planos de cuidados estão representados na classe `DataFlow`. Cada fluxo de dados possui um identificador único, um nome e uma representação do fluxo armazenado em string codificada em JSON. O `DataFlow` possui ainda uma propriedade indicando a camada apropriada para sua execução: `fog` ou `cloud`.

Por fim, a interface administrativa, ilustrada na Figura 4 permite o gerenciamento do HealthDash via painel de controle, que pode ser acessado pelo navegador. Este módulo é destinado tanto ao *coach* de saúde, quanto ao administrador do sistema, responsável pelas questões de ordem técnicas, mantendo as configurações para que o sistema esteja operacional. Um vídeo, disponível no link https://youtu.be/7JDywvIUGzQ, demonstra o processo de utilização do HealthDash, desde o gerenciamento dos fluxos de dados e dos esquemas, até a manutenção dos planos de cuidados para os pacientes.

Ao administrador, cabe gerenciar os esquemas de dados, bem como as diferentes versões que porventura sejam criadas, além de controlar os fluxos de dados. Os esquemas

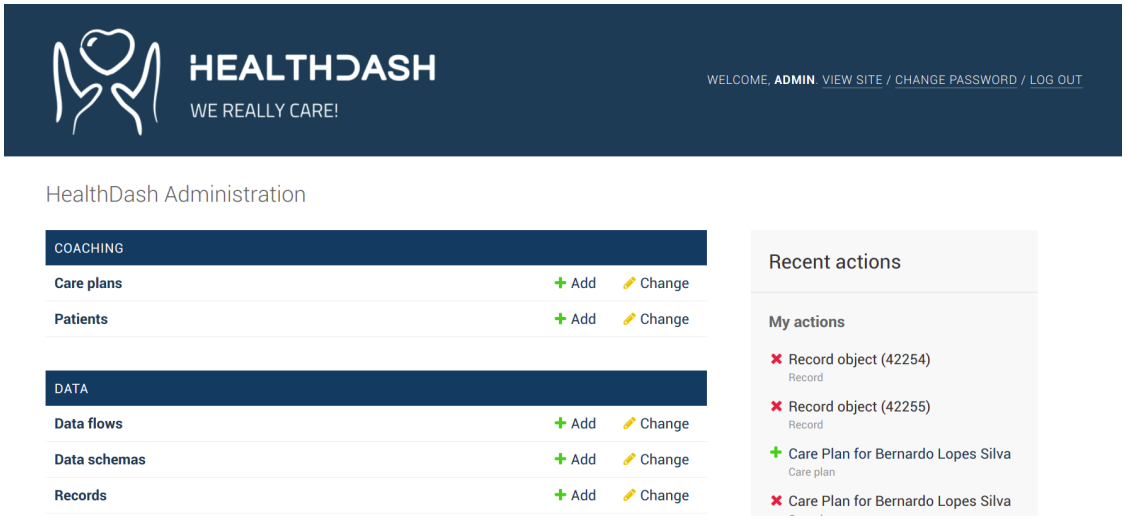

Figura 4. Interface administrativa do `HealthDash`.

de dados fornecem um meio de validar os dados recebidos pela aplicação, em conformidade com uma estrutura previamente concebida. Por fim, os *gateways* são configurados, a fim de manter a sincronia dos fluxos de dados necessários para cada paciente.

A função do *coach* de saúde é facilitar a identificação, por parte do paciente, de metas de saúde ou de assistência médica e facilitar a mudança de comportamento necessária para alcançar estes objetivos. O *coach* de saúde é uma espécie de mentor que apoia e monitora os pacientes, a fim de promover um estilo de vida mais saudável. A interface administrativa permite ao *coach* ter uma visão geral da saúde dos pacientes por ele monitorados, por meio do módulo de *coaching*. Neste módulo, o *coach* de saúde gerencia o cadastro dos pacientes e os planos de cuidados, que podem ser personalizados de acordo com a necessidade de cada indivíduo.

6. Avaliação

Para a avaliação da HealthDash, utilizamos o cenário de monitoramento de pacientes crônicos em domicílio, coletando dados de pressão arterial e glicemia. A partir deste cenário, são discutidas duas questões relevantes: o tempo de resposta e a transmissão de dados, a partir da comparação entre o emprego da programação com fluxo de dados utilizando a camada *fog* e a camada *cloud*.

6.1. Monitoramento remoto de doenças crônicas

O monitoramento remoto de pessoas com doenças crônicas em domicílio requer uma abordagem de gerenciamento de pacientes que combine várias tecnologias de informação a fim de monitorá-los a distância. Segundo [Pare et al. 2007], trata-se de um processo automatizado para a transmissão de dados sobre o estado de saúde de um paciente domiciliar para o respectivo ambiente de assistência médica, sem a necessidade de um profissional de saúde no local em que o indivíduo se encontra. Neste caso, apenas os pacientes ou familiares são os responsáveis por coletar e transmitir os dados monitorados.

Pesquisas apontam os efeitos positivos desta abordagem na condição de saúde do paciente e no processo geral do cuidado da saúde [Liu et al. 2016]. Do ponto de vista

tecnológico e de processamento das informações, duas abordagens são comumente empregadas: o processamento local (*fog*), com o auxílio de um *gateway* e o processamento dos dados diretamente na *cloud*. A abordagem empregada depende do tipo de dado sensoriado e a implementação de tais tecnologias implica diferentes desafios, como tempo de resposta, segurança, privacidade e transmissão de dados.

Na avaliação da arquitetura proposta, o mesmo fluxo será testado utilizando duas abordagens: a primeira detectando as anomalias na camada *cloud* e a segunda detectando as anomalias na camada *fog*, conforme ilustra a Figura 5. Uma das vantagens da proposta é exatamente a utilização de uma mesma tecnologia (programação em fluxo de dados) para especificar a lógica da aplicação em qualquer das camadas.

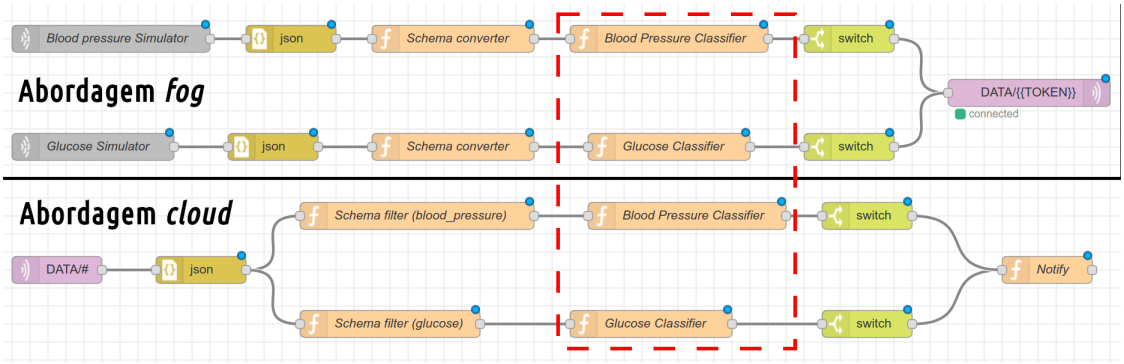

Figura 5. Fluxo de dados utilizados para detecção de anomalias na abordagem *fog* e *cloud*

O processo empregado no experimento em cada camada (*fog* e *cloud*) é ilustrado na Figura 6. O diagrama engloba as etapas de (1) recepção, (2) empacotamento e (3) transmissão dos dados dos sensores para a *cloud*, seguidas do (4) desempacotamento dos dados na *cloud*, (5) detecção de anomalias e (6) notificação. A ordem de execução e a camada em que cada etapa é implementada depende da abordagem utilizada.

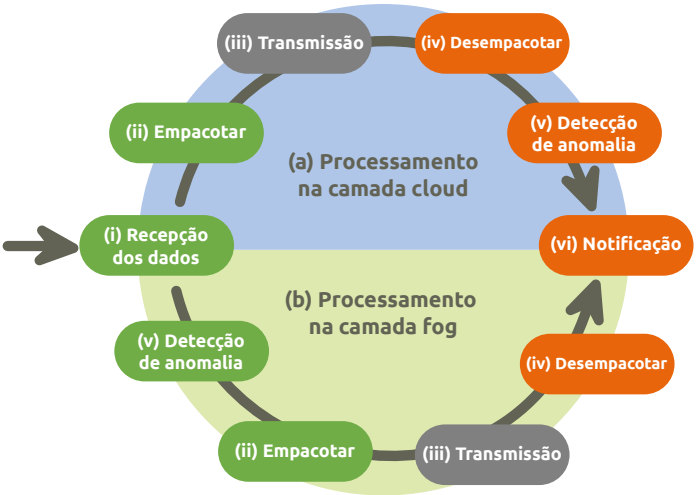

Figura 6. Diagrama de fluxo de dados utilizado no experimento

Na abordagem com processamento em *cloud*, os dados dos sensores são capturados e encapsulados na camada *fog* e, em seguida, enviados para a detecção de anomalias

na *cloud*. Na abordagem baseada em *fog*, os dados dos sensores são processados localmente e, apenas em caso de anomalias, os registros são enviados para a camada *cloud* para que as ações de notificação sejam realizadas.

6.2. Ambiente de teste

Para realizar os testes, foram selecionados 3 provedores de *cloud* diferentes: Azure, Digital Ocean e Google Cloud, com configurações semelhantes de máquinas virtuais com 2 núcleos de processamento, 4GB de RAM, e armazenamento em SSD.

As trocas de mensagens com a *cloud* foram realizadas utilizando o EMQ X, um *broker* MQTT de mensagens distribuído, de código aberto, que provê mecanismos de escalabilidade e alta disponibilidade, voltados à aplicações IoT e que é capaz de gerenciar milhões de clientes simultaneamente. Foi criado também um simulador que gera dados de glicemia e pressão arterial e que permite configurar diversos parâmetros, como o número de usuários conectados simultaneamente, a quantidade de leituras geradas diariamente e a porcentagem de leituras consideradas anormais.

Foram definidos 4 fluxos de dados no HealthDash, conforme definido na Figura 6. Nesta avaliação, testamos o simulador com os seguintes parâmetros: 100, 1000 e 2000 usuários simultâneos, registrando 3 coletas de glicemia e 3 coletas de pressão arterial com 20% de registros anormais. Os dados foram disparados simulando a quantidade total de usuários conectados simultaneamente ao servidor, em intervalos de 1 segundo entre cada leitura, contabilizando 600, 6000 e 12000 registros em cada cenário de teste, respectivamente. A seguir, foram observados o tempo médio de resposta, a quantidade total de dados transmitidos e a questão da privacidade para as duas implementações propostas, com detecção de anomalias na camada *fog* e na camada *cloud*.

6.3. Latência

Uma variável crítica a ser observada ao se propor soluções para a área da saúde é a latência da resposta. Ela é composta pelo tempo de pré-processamento, o tempo transmissão de dados e o tempo de processamento. As Equação 1 detalha como é calculada a latência média do sistema para as abordagens *cloud* e fog, respectivamente:

$$Lat_{fog|cloud} = T_{preproc} + T_{trans} + T_{proc} \quad (1)$$

Nesta equação, $T_{preproc}$ é o tempo médio de pré-processamento do dado, em milissegundos (ms), na camada local *fog*, T_{trans} o tempo médio de transmissão para a camada *cloud* (ms), e T_{proc} representa o tempo médio para análise de dados e tomada de decisão na camada $cloud$ ou fog, de acordo com a abordagem adotada. Os resultados, sumarizados na Tabela 1, indicam que a latência é limitada basicamente pelo tempo de transmissão.

Tabela 1. Latências médias para *cloud* e *fog* com tráfego de dados

Usuários	**Lat$_{cloud}$(ms)**				**Lat$_{fog}$(ms)**				**KB trafegados**	
	$T_{preproc}$	T_{trans}	T_{proc}	T_{total}	$T_{preproc}$	T_{trans}	T_{proc}	T_{total}	*Cloud*	*Fog*
100	0.10	17.05	0.19	17.34	0.17	7.12	0.14	7.43	139.31	29.66
1000	0.08	172.87	0.14	173.09	0.18	169.22	0.12	169.52	1398.91	297.70
2000	0.10	279.91	0.19	280.2	0.18	159.82	0.11	160.11	3770.69	596.01

6.4. Transmissão de dados e armazenamento

Outro teste realizado buscou avaliar a quantidade de dados transmitidos do *gateway* para a camada *cloud*, além do espaço gasto para armazenamento dos servidores em *cloud*. A parte mais à direita da Tabela 1 apresenta os resultados obtidos para as abordagens *cloud* e *fog* da arquitetura HealthDash.

Os valores obtidos indicam uma redução significativa na utilização da largura de banda em um período de 1 dia de monitoramento. Essa redução de tráfego de dados é de aproximadamente 78% se for utilizada a abordagem baseada em *fog*. Essa redução pode ser ainda mais significativa dependendo do tipo de sinal a ser monitorado. Quanto maior a taxa de amostragem exigida pelo cenário, maior será a redução de uso de banda utilizando a abordagem baseada em *fog* em comparação com a abordagem baseada em *cloud*, para o cenário analisado.

7. Conclusão

Este trabalho apresentou a arquitetura HealthDash para desenvolvimento de soluções de IoT aplicadas à saúde. A característica principal da arquitetura é o uso da programação orientada a fluxo de dados desde camada *cloud* até a camada *fog*, unificando tecnologias de desenvolvimento em todos níveis da aplicação. O objetivo é reduzir a complexidade e aumentar a flexibilidade para criar e manter as soluções computacionais aplicadas à saúde, onde os cenários são bastante complexos e dinâmicos.

As principais contribuições do HealthDash são: (1) Prover dinamicidade no processo de criação de aplicações IoT, tanto na camada *fog* quanto na camada *cloud*, por meio da unificação das tecnologias adotadas em ambos os níveis; (2) fornecer meios para diminuir o volume de informações trafegadas entre a camada *fog* e a camada *cloud*, possibilitando a análise e filtragem dos dados mais próximo à borda da rede.

Os testes realizados evidenciam o potencial do HealthDash para o desenvolvimento de aplicações que necessitam de maior flexibilidade, em cenários que demandam baixa latência ou quando existe a necessidade de pré-processar dados, antes de transmitilos pela rede, resultando em economia de banda. Em relação à elasticidade da solução, novos testes precisam ser realizados, uma vez que encontramos problemas com o *client* MQTT ao escalar a solução para além de 2000 conexões simultâneas. Estes testes devem incluir novos *brokers* e protocolos de comunicação, como COAP, AMQP e DDS.

Além disso, a arquitetura proposta tem potencial para contribuir para a segurança e privacidade das informações dos usuários, uma vez que é possível determinar em que nível da aplicação cada tipo de informação será processada. Por exemplo, no caso de informações sensíveis, como imagens ou streamming de vídeo, é possível processar as informações localmente, enviando à *cloud* apenas a ocorrência de eventos que se deseja monitorar. Os próximos passos incluem simulações com número maior de usuários, a fim de testar a escalabilidade da ferramenta, além do monitoramento real de pacientes crônicos em domicílio. Serão ainda implementados novos mecanismos de privacidade que permitam ao usuário final do HealthDash determinar quais fluxos podem ser executados e quais informações podem ser encaminhadas para a *cloud*.

Referências

Aazam, M. and Huh, E. N. (2015). E-HAMC: Leveraging Fog computing for emergency alert service. In *2015 IEEE International Conference on Pervasive Computing and Communication Workshops, PerCom Workshops 2015*, pages 518–523. IEEE.

Ahmad, M., Amin, M. B., Hussain, S., Kang, B. H., Cheong, T., and Lee, S. (2016). Health Fog: a novel framework for health and wellness applications. *The Journal of Supercomputing*, 72(10):3677–3695.

Cerina, L., Notargiacomo, S., Paccanit, M. G., and Santambrogio, M. D. (2017). A fog-computing architecture for preventive healthcare and assisted living in smart ambients. *Proc. Int. Forum on Research and Technologies for Society and Industry - RTSI 2017*.

Farahani, B., Firouzi, F., Chang, V., Badaroglu, M., Constant, N., and Mankodiya, K. (2018). Towards fog-driven IoT eHealth: Promises and challenges of IoT in medicine and healthcare. *Future Generation Computer Systems*, 78:659–676.

Gia, T. N., Jiang, M., Rahmani, A. M., Westerlund, T., Liljeberg, P., and Tenhunen, H. (2015). Fog computing in healthcare Internet of Things: A case study on ECG feature extraction. *Proceedings of the CIT 2015*, pages 356–363.

Giang, N. K., Blackstock, M., Lea, R., and Leung, V. C. (2015). Developing IoT applications in the Fog: A Distributed Dataflow approach. *Proceedings - 2015 5th International Conference on the Internet of Things, IoT 2015*, (January 2016):155–162.

Gupta, H., Vahid Dastjerdi, A., Ghosh, S. K., and Buyya, R. (2017). iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments. *Software: Practice and Experience*, 47(9):1275–1296.

Hong, K., Lillethun, D., Ramachandran, U., Ottenwälder, B., and Koldehofe, B. (2013). Mobile fog: a programming model for large-scale applications on the internet of things. In *Proceedings of the second ACM SIGCOMM workshop on Mobile cloud computing - MCC '13*, page 15, New York, New York, USA. ACM Press.

Liu, L., Stroulia, E., Nikolaidis, I., Miguel-Cruz, A., and Rios Rincon, A. (2016). Smart homes and home health monitoring technologies for older adults: A systematic review. *International Journal of Medical Informatics*, 91:44–59.

Malik, B. H., Cheema, S. N., Iqbal, I., Mahmood, Y., Ali, M., and Mudasser, A. (2018). From Cloud Computing to Fog Computing (C2F): The key technology provides services in health care big data. 03010.

Pare, G., Jaana, M., and Sicotte, C. (2007). Systematic Review of Home Telemonitoring for Chronic Diseases: The Evidence Base. *Journal of the American Medical Informatics Association*, 14(3):269–277.

Rahmani, A. M., Gia, T. N., Negash, B., Anzanpour, A., Azimi, I., Jiang, M., and Liljeberg, P. (2018). Exploiting smart e-Health gateways at the edge of healthcare Internet-of-Things: A fog computing approach. *Future Generation Computer Systems*, 78:641–658.

Sousa, T. B. (2012). Dataflow Programming Concept, Languages and Applications. *Doctoral Symposium on Informatics Engineering*, 7(November):13.