Unindo Aplicações Críticas e Sensores IoT com QoS Individual e Adaptativo em Hospitais Inteligentes

Vinicius F. Rodrigues[1], **Lucas M. Policarpo**[1],
Rodrigo da R. Righi[1], **Cristiano A. da Costa**[1]

[1]PPG em Computação Aplicada - UNISINOS - Av. Unisinos, 950, São Leopoldo, RS

{vfrodrigues,lmpolicarpo,rrrighi,cac}@unisinos.br

Abstract. *With smart sensors in hospitals, critical decision-making can take place based on patients and equipment real-time monitoring. However, problems in generating and transmitting data can cause failures on applications that process this data. In this context, this article proposes HealthStack, a middleware for operating rooms with quality of service (QoS) strategies and support for real-time data transmission. We also present a QoS strategy based on artificial neurons to select middleware components with critical performance. We developed a prototype of the model and tested it in an actual operating room achieving jitter reduction up to 90.3%.*

Resumo. *Com a utilização de sensores inteligentes em hospitais, tomadas de decisões críticas podem ser realizadas baseadas no monitoramento em tempo real de pacientes e equipamentos. Porém, erros na geração e transmissão de dados podem causar falhas no resultado de aplicações que processam esses dados. Neste contexto, esse artigo propõem HealthStack, um middleware para salas cirúrgicas com estratégias de qualidade do serviço (QoS) e suporte a transmissão de dados em tempo real. O artigo propõe uma estratégia de QoS baseada em neurônios artificiais para seleção dos componentes do middleware com baixa performance. Foi desenvolvido e testado um protótipo do modelo em uma sala de cirurgia real possibilitando redução de jitter médio em até 90,3%.*

1. Introdução

Recentemente, estima-se que Internet das Coisas, ou *Internet of Things* (IoT), tenha atingido a marca de 50 bilhões de dispositivos conectados em 2020 [Munirathinam 2020]. O grande número de dispositivos conectados produzindo diferentes tipos de informações, foi o que possibilitou o surgimento da Indústria 4.0 (I4.0) [Oztemel and Gursev 2020, Munirathinam 2020]. Mais especificamente, um dos pilares da I4.0 é a combinação das tecnologias de IoT e Inteligência Artificial (IA) [Oztemel and Gursev 2020]. Após I4.0, atualmente já é comum o termo "Hospital 4.0" para se referir à revolução dos hospitais devido à chegada das tecnologias IoT em ambientes de saúde [Aceto et al. 2020]. Nesse novo paradigma, o foco principal está na personalização dos serviços clínicos para aumentar a qualidade do atendimento aos pacientes [Aceto et al. 2020]. Novos sistemas de informação e monitoramento capturam, armazenam e processam dados de sensores em tempo real para decisões médicas [Barbosa et al. 2020]. Neste cenário, o tempo de resposta para situações críticas é decisivo para salvar vidas em cirurgias, por exemplo. Mais importante, ações mais rápidas para tratar uma crise de saúde aumentam a probabilidade de melhores resultados.

A variedade de sensores e aplicações em salas cirúrgicas exige que, os sistemas de tempo real estejam atentos à heterogeneidade dos dados e sua relevância para a escolha das melhores estratégias em seu processamento e transmissão. Isso significa, que a solução deve adaptar suas estratégias de acordo com os dados e requisitos da aplicação (atraso, jitter, etc.) fornecendo qualidade de serviço, comumente chamado de QoS (*Quality of Service*). Embora existam várias abordagens para fornecer QoS para aplicativos de saúde, é um desafio produzir e entregar dados em tempo real a partir de sensores [Sisinni et al. 2018]. Tais sistemas consistem em infraestruturas distribuídas e complexas que podem sofrer muitos problemas, como interferência de transmissão e restrições de energia. A literatura atual apresenta muitos estudos que empregam estratégias de QoS para arquiteturas de saúde. No entanto, esses estudos são restringidos pelas seguintes limitações: (*i*) as estratégias não focam em ambientes hospitalares críticos; (*ii*) as soluções não combinam múltiplas estratégias em diferentes níveis; e (*iii*) iniciativas não colocam esforço especificamente em transmissões de dados em tempo real.

Nesse contexto, este artigo propõe o HealthStack, um middleware multicamada para salas cirúrgicas com suporte dinâmico de QoS para transmissão de dados em tempo real. HealthStack apresenta uma arquitetura com componentes distribuídos e protocolos de comunicação para fornecer dados em tempo real de sensores para aplicativos em uma forma de publicar-assinar (PubSub). A estratégia de QoS consiste em empilhar serviços de QoS para cada componente da arquitetura de middleware dinamicamente, de acordo com as violações de QoS no lado dos aplicativos. A pilha pode receber quatro serviços de QoS diferentes aplicados às amostras de dados ou aos recursos de computação: priorização de dados, adaptação de taxa de frequência de dados, compressão de dados e elasticidade de recursos. O HealthStack compreende um componente QoS Manager que está ciente dos requisitos de QoS de cada aplicativo e do status atual. Cada componente tem uma pilha de serviço flexível individual que pode aumentar ou diminuir dinamicamente à medida que as solicitações do aplicativo aumentam ou ocorrem violações de QoS. Os serviços consistem em diferentes estratégias de QoS de domínios de rede e recursos. Para avaliar o modelo, implantamos o HealthStack em uma sala de cirurgia equipada com sensores de imagem de profundidade e tags de banda ultralarga (UWB) para rastreamento de posição interna. As principais contribuições apresentadas por este artigo são: (*i*) um middleware para salas cirúrgicas com suporte QoS automático para transmissão de dados em tempo real; e (*ii*) estratégia de QoS baseada em neurônios artificiais para selecionar componentes de middleware com desempenho crítico.

O restante deste artigo está organizado da seguinte forma. A Seção 2 apresenta uma revisão da literatura no escopo de estratégias de QoS para soluções de saúde. A Seção 3 apresenta o middleware proposto e o modelo de QoS em detalhes. Posteriormente, a Seção 4 apresenta a metodologia de avaliação, e a Seção 5 mostra os resultados da avaliação. Finalmente, a Seção 6 conclui o artigo com as considerações finais, limitações e orientações de pesquisas futuras.

2. Trabalhos Relacionados

A seleção de trabalhos relacionados seguiu diretrizes baseadas na metodologia de revisões sistemáticas [Kitchenham and Charters 2007] selecionando artigos da seguinte forma: (“QoS” no título do artigo) E (pelo menos uma dessas palavras no metadados: “health”, “healthcare”, or “hospital”). A seleção de artigos resultou em 15 artigos,

os quais foram analisados quanto a suas estratégias de QoS. A Tabela 1 apresenta as principais características dos trabalhos analisados. Considerando as estratégias de QoS empregadas pelos autores, podemos destacar quatro em comum, independentemente do foco da pesquisa: (*i*) protocolos de escalonamento [Samanta and Misra 2018, Samanta et al. 2018, Iranmanesh and Rizi 2018, Venkatesh et al. 2019]; (*ii*) priorização de dados [Samanta et al. 2018, Guezguez et al. 2018, Iranmanesh and Rizi 2018, Wang, Q. et al. 2019, Al-Tarawneh 2019, Venkatesh et al. 2019, Khalil et al. 2019, Goyal et al. 2020]; (*iii*) protocolos de roteamento [Guezguez et al. 2018, Zitta et al. 2018, Iranmanesh and Rizi 2018]; e (*iv*) gerenciamento de recursos [Celdrán et al. 2018, Wang et al. 2018, Goyal et al. 2020]. Vários estudos empregam a priorização de dados para melhorar o atraso de tempo para a transmissão de dados. A estratégia consiste em definir classes diferentes para cada tipo de dados e priorizar a transmissão de pacotes de dados com níveis de alta prioridade. Algumas abordagens empregam algoritmos de controle de admissão para garantir que novas transmissões de dados não diminuam a qualidade da rede. Portanto, o sistema só admite novas conexões após aplicar um algoritmo de avaliação do estado atual da rede e estimar o novo estado.

Tabela 1. Artigos focados em estratégias de QoS para soluções na área da saúde.

Artigo	Tempo real	Foco	Método QoS
[Samanta and Misra 2018]	✓	WBAN	Protocolo de escalonamento (priorização de dados)
[Samanta et al. 2018]		WBAN	Protocolo de escalonamento com priorização de dados, controle de admissão
[Celdrán et al. 2018]	✓	ICE	Balanceamento
[Guezguez et al. 2018]	✓	WSN, WBAN	Priorização de dados, controle de admissão, protocolo de roteamento
[Zitta et al. 2018]	✓	IoT	Transmissão multicanal
[Iranmanesh and Rizi 2018]		WBAN	Priorização de dados, controle de admissão, protocolo de roteamento, alocação de espaços de tempo
[Wang et al. 2018]		WBAN	Controle de energia
[Gatouillat et al. 2018]		IoT	Feedback e loop de controle
[Sodhro et al. 2019]	✓	Telemedicina	Suavização de vídeo
[Wang, Q. et al. 2019]		Telemedicina	Priorização de dados
[Al-Tarawneh 2019]		Medical network	Diferenciação de serviço, categorização e priorização de dados
[Venkatesh et al. 2019]	✓	SDN	Diferenciação de serviço, protocolo de roteamento
[Bai et al. 2019]		WBAN	Sincronização de relógio
[Khalil et al. 2019]	✓	IoT	Diferenciação de tráfego
[Goyal et al. 2020]	✓	WBAN	Priorização de dados e adaptação de taxa de dados

A elasticidade dos recursos é uma estratégia comum em ambientes de nuvem, e algumas pesquisas empregam essa ideia para fornecer QoS. Nessas estratégias, as soluções empregam técnicas de alocação e migração de máquinas virtuais para aumentar o desempenho de alguns módulos da arquitetura em situações de sobrecarga. São também empregados protocolos de roteamento e escalonamento para melhorar o desempenho da rede e diminuir o atraso na transmissão. O primeiro consiste em algoritmos que definem a melhor rota para transmitir os dados de acordo com o estado atual da rede. Já o segundo, consiste em algoritmos que controlam a alocação de faixas de tempo para cada transmissão de dados e sua ordem de acordo com a quantidade e o tipo de dados. Com base na análise da literatura, três lacunas de pesquisa principais são aparentes: (*i*) as estratégias não se concentram em ambientes hospitalares críticos; (*ii*) as soluções não combinam estratégias múltiplas em níveis diferentes; e (*iii*) iniciativas não colocam esforço especificamente em transmissões de dados em tempo real.

Primeiro, a maioria das estratégias concentra-se em saúde remota, como lares de idosos, e apenas WBANs para monitoramento da saúde do paciente. Em segundo lugar,

um sistema de saúde tem dois atores principais: os sensores no nível do *hardware*, gerando dados, e os usuários no nível de *software*, que consomem e processam os dados do sistema. As estratégias se concentram principalmente no primeiro nível para fornecer QoS na transmissão de dados do sensor para a rede. Embora algumas iniciativas apresentem preocupações em relação ao tempo real, elas não se concentram intensamente nesta questão. Em geral, essas soluções se consideram apenas melhorar o atraso de tempo de pacotes prioritários ou fornecer uma arquitetura que suporte transmissões de dados em tempo real. No caso específico da sala cirúrgica, o tempo real é o atributo mais importante para permitir a previsão de situações críticas que podem afetar a saúde dos pacientes. Em outras palavras, ter informações em tempo real pode representar a diferença entre morte ou vida. Com base nas lacunas apresentadas, este trabalho busca fornecer um modelo de middleware focado em ambientes de saúde críticos que será detalhado na próxima seção.

3. HealthStack: Um Modelo de Elasticidade de Qualidade de Serviço

Esta seção apresenta o modelo HealthStack. O modelo se concentra em integrar dados de muitos dispositivos em ambientes hospitalares e fornecê-los em um formato padrão aos aplicativos. Portanto, as aplicações de usuário não precisam implementar diferentes APIs (*Application Programming Interface*) para adquirir dados de diferentes dispositivos. Elas podem se conectar facilmente ao middleware e solicitar dados de diferentes sensores em tempo real. A Figura 1 mostra as principais ideias de HealthStack. A figura à esquerda (a) demonstra as situações atuais em que aplicações devem ter acesso através de uma rede cabeada aos diferentes dispositivos para acessá-los diretamente. Por outro lado, nossa estratégia em (b) introduz o middleware entre as aplicações e dispositivos apresentando dois componentes principais.

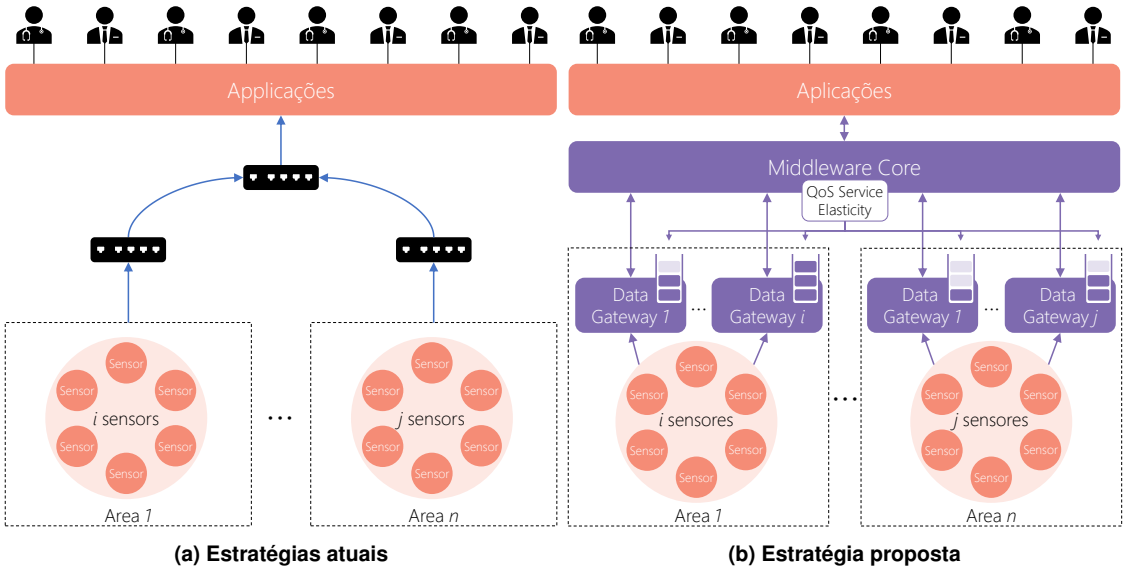


Figura 1. Visão global de estratégias atuais (a) e a solução proposta (b).

3.1. Arquitetura

HealthStack é composto por Data Gateways, que são componentes com acesso direto aos dados de dispositivos oferecendo serviços de QoS individualmente. O Middleware Core é o principal componente que coleta os dados de cada Gateway e os disponibiliza

para aplicações. Portanto, o Core pode coletar dados de muitos locais diferentes, desde que tenham instâncias de Gateways distribuídas. O Core fornece dados aos aplicativos por meio de uma interface de *publish/subscribe* [Behnel et al. 2006]. As aplicações se conectam ao Core solicitando dados de um sensor específico. O Core oferece suporte a parâmetros adicionais nas solicitações além do sensor de destino. Os aplicativos podem fornecer um limite desejável para atraso (delay) e jitter. Caso a aplicação não forneça esses limites, o Core assume os valores padrão de acordo com o tipo de sensor alvo. Por exemplo, para sensores de imagem, o delay máximo padrão aceito é de 50 milissegundos e jitter é de 25 milissegundos.

A Figura 2 ilustra a arquitetura do modelo em mais detalhes. O Middleware Core é responsável por quatro tarefas principais: (*i*) Data Provisioning; (*ii*) Performance Tracking; (*iii*) QoS Service Elasticity; e (*iv*) Data Persistance. Data Provisioning compreende o processo responsável por coletar dados de instâncias individuais de Data Gateways e o processo encarregado de servir as conexões de aplicações. Performance Tracking monitora o desempenho individual de cada instância de Gateway e cada conexão de aplicação. A tarefa produz informações usadas para a tomada de decisões pelo QoS Service Elasticity. Analisando essas informações, QoS Service Elasticity estima quais serviços de QoS são necessários para garantir QoS para aplicações. Este processo gerencia as pilhas de serviço de QoS das instâncias de Gateway correspondentes. Essa estratégia visa melhorar o desempenho dos Gateways que produzem dados para as aplicações que têm seu QoS violado. Portanto, o middleware pode lidar com a deterioração do desempenho causada por, por exemplo, várias conexões e sobrecarga de rede.

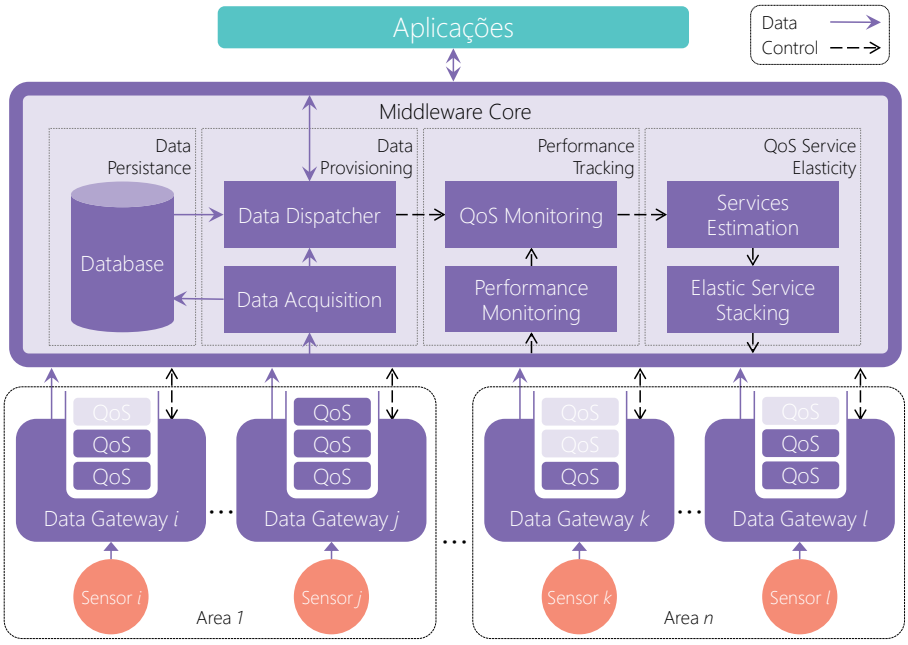


Figura 2. Arquitetura do modelo HealthStack e seus componentes.

3.2. Modelo de Elasticidade de Serviços QoS

Além da arquitetura, o middleware fornece uma estratégia de elasticidade de serviços QoS, chamada QoS Service Elasticity, para melhorar o desempenho das aplicações. A elasticidade de serviços QoS consiste em pilhas de serviço de QoS elásticas individuais para cada instância de Gateway. O Core gerencia essas pilhas de acordo com os requisitos

das aplicações e o desempenho atual do middleware. O Core monitora periodicamente se os limites estabelecidos de QoS (delay ou jitter) são respeitados ou não. Assim que ocorre uma violação de limite, o Core empilha diferentes serviços de QoS para os dispositivos sensores para resolver o problema. Portanto, as pilhas de serviço são elásticas e o Core muda quando necessário. O middleware compreende quatro serviços de QoS diferentes: (*i*) Priorização de dados; (*ii*) Compressão de dados; (*iii*) Adaptação de taxa de dados; e (*iv*) Elasticidade de recursos. Cada serviço aborda um problema diferente.

A priorização de dados define os dados de um sensor como de alta prioridade. Dessa forma, o Core fornece mais ciclos de CPU para processar esses dados. O serviço de compactação de dados ativa compressão nos Gateways a fim de reduzir a quantidade de bytes transmitidos na rede, melhorando assim seu desempenho. Por sua vez, a adaptação da taxa de dados tem dois objetivos principais: diminuir a quantidade de bytes transmitidos e reduzir o consumo de recursos de computação. Finalmente, o último serviço, elasticidade de recursos, aumenta ou diminui os recursos computacionais disponíveis para o Gateway. Este serviço específico se concentra na redução do consumo de energia em situações em que não há aplicações consumindo dados de um sensor específico.

Para analisar, planejar e executar adaptações, o QoS Service Elasticity monitora periodicamente várias variáveis de desempenho e status dos componentes e aplicativos de middleware. No lado da aplicação de usuário, duas variáveis medem os valores atuais de delay ($delay_a$) e jitter ($jitter_a$) que o middleware entrega para uma dada aplicação a. Os Gateways têm sete variáveis diferentes, uma vez que, essas instâncias são as principais responsáveis por adquirir e empacotar as amostras dos sensores: CPU (cpu_s), memória (mem_s), rede (net_s), delay ($delay_s$), jitter ($jitter_s$), conexões ($conn_s$), e prioridade (pri_s). As duas últimas variáveis representam, respectivamente, quantas aplicações estão consumindo dados do Gateway s, e o parâmetro de nível de prioridade do sensor para este Gateway s. Dadas todas as variáveis de entrada, o modelo emprega um processo de orquestração de serviços para identificar violações de QoS e realizar adaptações no middleware. O objetivo principal deste processo é selecionar uma instância de Gateway em cada ciclo de monitoramento e empilhar serviços de QoS nela se houver pelo menos uma violação de QoS no lado das aplicações. Para resolver esse problema, o modelo propõe a métrica Potencial de Adaptação (PA) para calcular a probabilidade de selecionar um determinado Gateway. Mais especificamente, ele emprega transformação linear que usa a função sigmoide e pesos adaptativos [Nielsen 2015]. Sua saída é produzida por $\sigma\left(W \cdot X + b\right)$, em que σ é a função sigmoide logística $\sigma(x) = \frac{1}{1+e^{-x}}$.

A Equação 1 define PA, a qual recebe todas as variáveis de entrada ($i_0,\ i_1,\ \ldots,\ i_j$) e pesos correspondentes ($w_0,\ w_1,\ \ldots,\ w_j$) os combinando em uma soma ponderada $\sum_0^j w_j i_j$. O resultado final é obtido aplicando a função de sigmoide de ativação no valor resultante. No final o Gateway com o maior PA possui a maior probabilidade de adaptação, sendo o escolhido para elasticidade de serviços. A maior contribuição dessa estratégia está nos pesos aplicados às variáveis de entrada delay e jitter. PA emprega pesos dinâmicos calculados em tempo de execução conforme a variação dos limites de QoS das aplicações e suas possíveis violações. Isso torna possível aumentar pesos para Gateways que produzem dados para aplicações que têm seus limites de QoS violados.

$$PA(s) = \sigma \begin{pmatrix} & w_0 & \times & cpu_s \\ + & w_1 & \times & mem_s \\ + & w_2 & \times & net_s \\ + & w_3 & \times & conn_s \\ + & w_4 & \times & pl(s) \\ + & w_5(s) & \times & delay_s \\ + & w_6(s) & \times & jitter_s \end{pmatrix} \quad (1)$$

De acordo com a Equação 1, $PA(s)$ calcula PA para cada Gateway s. Os pesos w_0, w_1, w_2, w_3, e w_4 são valores fixados por parâmetro. Por outro lado, as funções $w_5(s)$ e $w_6(s)$ calculam os pesos correspondentes para serem aplicados a $delay_s$ e $jitter_s$. Diferente de outras métricas que são multiplicadas por pesos, para a variável pri_s, a função $pl(s)$ calcula o nível de prioridade baseando nas prioridades de todos os sensores. A Tabela 2 demonstra todas as equações para o cálculo das funções empregadas em PA e suas respectivas descrições. Em especial, α é um valor real definido como parâmetro que define a taxa de importância de uma violação de QoS.

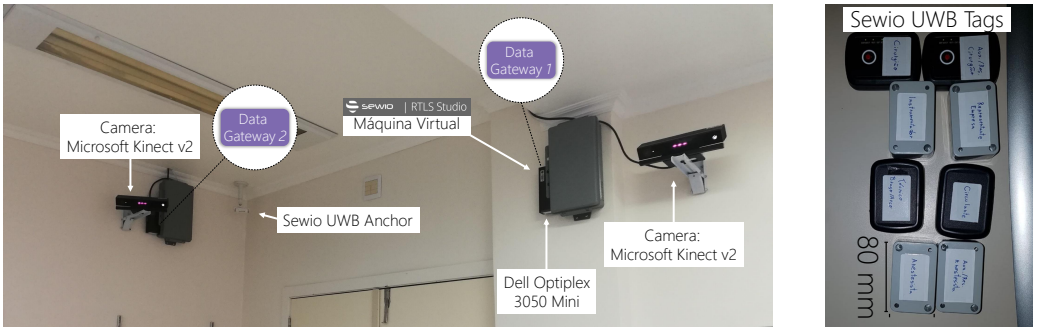
Tabela 2. Equações usadas no cálculo de PA.

Equação	Descrição
$pl(s) = \frac{pri_s}{\sum_{i=1}^{d} pri_i}$	$pl(s)$ do Gateway s é o resultado da divisão de sua prioridade pri_s pela soma das prioridades de todos os d sensores dos Gateways. Essa equação resulta em um valor no intervalo (0,1].
$w_5(s) = iu(D_s) \times \left(1 + \alpha \times \sum_{i=1}^{a} dv_{is}\right)$	D_s representa um vetor unidimensional com o QoS $delay$ requerido por cada uma das aplicações referente ao sensor do Gateway s. $dv_i s$ representa um valor binário para relação entre Gateway s e aplicação i, em que $dv_i s = 1$ significa que a aplicação tem seu QoS (delay) violado em relação ao sensor do Gateway.
$w_6(s) = iu(J_s) \times \left(1 + \alpha \times \sum_{i=1}^{a} jv_{is}\right)$	J_s representa um vetor unidimensional com o QoS $jitter$ requerido por cada uma das aplicações referente ao sensor do Gateway s. $jv_i s$ representa um valor binário para relação entre Gateway s e aplicação i, em que $dv_i s = 1$ significa que a aplicação tem seu QoS (jitter) violado em relação ao sensor do Gateway.
$iu(V) = \begin{cases} 0 & if \ \sum v_i = 0 \\ \frac{1}{\min V^*} & if \ \sum v_i > 0 \end{cases}$	iu calcula um valor de importância baseando-se em um vetor unidimensional de entrada V, em que $V^* = V - \{0\}$. Se o vetor possui pelo menos um valor menor que 0, a função retorna o quanto uma unidade (1) representa para o menor valor do vetor.

4. Metodologia de Avaliação

Foi desenvolvido um protótipo de todos os módulos da arquitetura em C++ contemplando três serviços QoS: (*i*) Priorização de dados; (*ii*) Compressão de dados; e (*iii*) Adaptação de taxa de dados. O código fonte do modelo de elasticidade de serviços está disponível em um repositório público do Github (`https://github.com/viniciusfacco/healthstack`). Para a avaliação do modelo, foi instalado um protótipo de HealthStack em um hospital real parceiro do projeto, o Instituto de Cardiologia - Fundação Universitária de Cardiologia, (Porto Alegre, Rio Grande do Sul, Brasil). O hospital deu acesso ao projeto a uma sala de cirurgia híbrida utilizada diariamente no bloco cirúrgico, a qual acomoda principalmente procedimentos cardíacos. A Figura 3 demonstra o $hardware$ empregado e como foi instalado na sala cirúrgica. Foi instalado um sistema de localização em tempo real da Sewio (`https://www.sewio.net/`), o qual emprega tecnologia

UWB (*Ultra-Wideband*). Cada pessoa envolvida nos procedimentos carrega uma tag com identificação prévia a qual o sistema provê sua localização. Para adquirir dados dos dispositivos, foram empregados 2 computadores Dell Optiplex 3050 Mini, processador Intel i7-6560U 2.20 GHz, 8GB de memória, placa de rede de 1 Gbps.



(a) Equipamento instalado nas paredes **(b) Tags UWB**

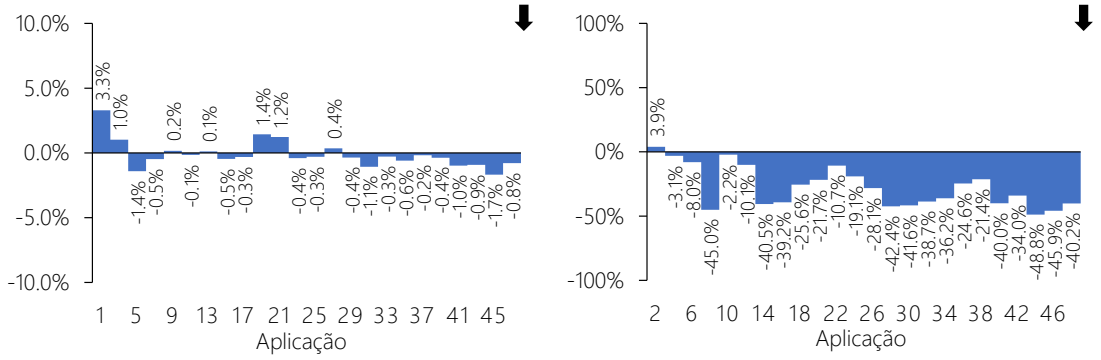
Figura 3. Implantação de HealthStack em uma sala cirúrgica híbrida real.

No momento de realização desta pesquisa, não havia disponível na literatura benchmarks de carga de trabalho para aplicações médicas em tempo real. Portanto, foi modelado um cenário composto por 48 aplicações que se conectam ao HeathStack sequencialmente, um a cada seis segundos. Cada nova aplicação (App) solicita um tipo de dados diferente do anterior. Especificamente, App 1 solicita dados de imagem do Kinect, App 2 dados de localização, App 3 dados de imagem, e assim por diante. Além disso, as aplicações definem seu limite de QoS de destino na seguinte sequência: delay=50ms, delay=50ms, jitter=25ms, jitter=25ms, e assim sucessivamente. Esses valores foram extraídos de alguns estudos em telemedicina presentes na literatura [Mukhopadhyay 2017, Nanda and Fernandes 2007]. A avaliação consistiu em observar a execução do middleware e das aplicações em uma janela de dez minutos em dois cenários: modo da elasticidade de serviços QoS (*i*) habilitado e (*ii*) desabilitado.

5. Análise dos Resultados

A Figura 4 ilustra a variação no delay médio para cada aplicação ao habilitar a elasticidade de serviços QoS. Os valores negativos representam um delay médio inferior, o que é uma melhoria. Como a aplicação solicita diferentes tipos de informações, os resultados estão segmentados de acordo com o tipo de dado para analisar a relevância.

A Figura 4 (a) demonstra que a maioria das aplicações que consomem dados de imagem experimentaram melhorias, mas nem todos eles. De um total de 24, 18 aplicações (75%) apresentaram melhorias no delay médio. Calculando a média do delay médio de cada uma das 24 aplicações (que consomem dados de imagem), obtem-se 84ms quando a elasticidade de serviços QoS está habilitada e 84,2ms quando desabilitada. Por outro lado, olhando a Figura 4 (b), as aplicações que consomem dados de localização alcançaram resultados mais significativos. Apenas uma entre 24 aplicações alcançou um delay médio mais alto ao habilitar a elasticidade de serviços QoS. A média do delay médio para essas aplicações ao habilitar a pilha de serviço foi de 3,6ms contra 5ms quando a elasticidade de serviços QoS foi desabilitada. Comparando os dois diagramas (a) e (b), o tamanho dos dados transmitidos às aplicações do primeiro conjunto é consideravelmente maior do

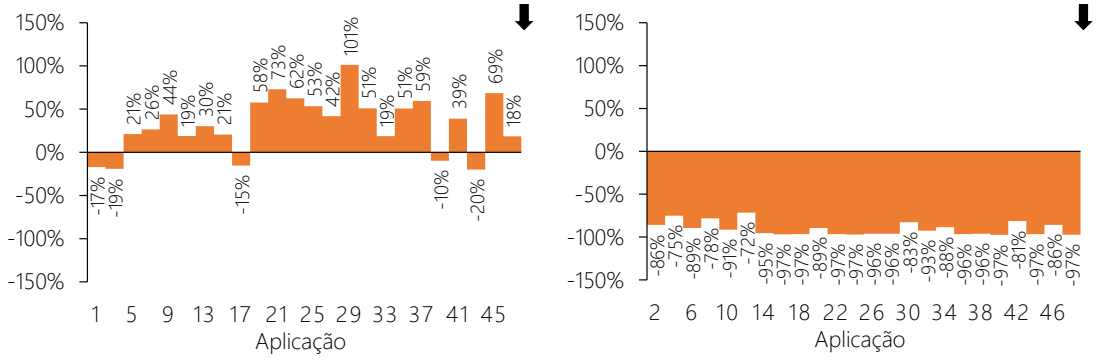


(a) Imagem **(b) Localização**

Figura 4. Variação no delay das aplicações com elasticidade de serviços QoS.

que os dados do segundo conjunto. Os dados de imagem requerem mais processamento e tempo de transmissão do que os dados de localização.

Agora, focando a métrica jitter, a Figura 5 descreve a variação no jitter médio para cada aplicação ao habilitar a elasticidade dos serviços QoS. Aqui, as variações são mais expressivas do que as de delay. No entanto, diferentes tipos de dados resultaram em valores contrastantes. Por um lado, a Figura 5 (a) demonstra os resultados para aplicações que consomem dados de imagem. Apenas cinco das 24 aplicações tiveram melhorias em seu jitter. Considerando todas as 24 aplicações, a média de jitter médio foi de 3,3ms com a elasticidade de serviços QoS habilitada contra 2,6ms se desabilitada. Embora não demonstrando melhorias, o jitter resultante ainda é baixo. Isso demonstra que habilitar a elasticidade de serviços QoS resulta em um melhor atraso não impondo maiores penalidades de jitter, já que um jitter menor que 25ms é totalmente aceitável [Nanda and Fernandes 2007].



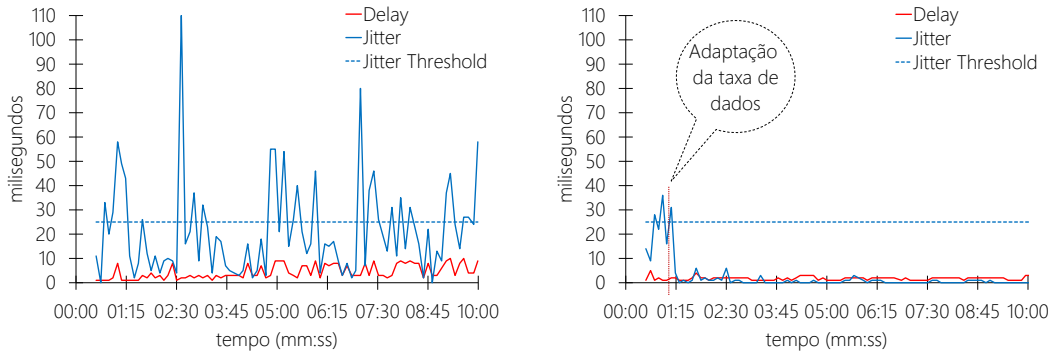
(a) Imagem **(b) Localização**

Figura 5. Variação no jitter das aplicações com elasticidade de serviços QoS.

Por outro lado, a Figura 5 (b) mostra que todas as aplicações que consomem dados de localização tiveram melhorias consideráveis. Considerando as 24 aplicações deste diagrama, a média de seu jitter médio foi de 2,1ms ao habilitar a elasticidade de serviço QoS contra 21,7ms ao desabilitá-la. Representa uma diminuição substancial em 90,3%. A principal razão para isso é que o sistema de loalização não fornece amostras de posições de tags a uma taxa estável, impactando diretamente no jitter. Nesse cenário, o serviço de adaptação da taxa de dados desempenha um papel essencial porque estabiliza a taxa de dados para um valor mais baixo. Ele também estabiliza o jitter, uma vez que o Data

Gateway adquire amostras a uma taxa mais baixa. Comparando ambas as Figuras 4 e 5, é possível notar que o tipo de dado passa a fazer diferença nos resultados. Embora os resultados demonstrem vantagens significativas para tipos de dados de tamanho pequeno, eles também demonstram resultados excelentes para tipos de dados de tamanho grande. Isso é importante no cenário da sala de cirurgia, uma vez que muitas informações críticas são tipos de dados de pequeno tamanho. Como exemplos, temos a temperatura corporal do paciente, eletrocardiograma, nível de saturação de oxigênio, etc.

A Figura 6 mostra as medidas de delay e jitter de uma das aplicações (ID 4) que conectou o middleware. Esta aplicação consome informações de localização e informa um limite de jitter de 25ms. Foram selecionados os dados de uma aplicação isolada para facilitar o entendimento, uma vez que, traçar todos os dados geraria uma visualização confusa. O diagrama (a) demonstra que o jitter viola o limite em vários momentos. Essas violações continuam ocorrendo durante toda a execução. Por outro lado, o diagrama (b) demonstra um comportamento diferente. Isso é possível porque na hora 01:08 (mm:ss), HealthStack empilha o serviço de adaptação de taxa de dados. Logo após esse momento, as métricas delay e o jitter caem e se mantêm estáveis até o fim da execução. Olhando especificamente para a linha Jitter, depois de violar o limite nos primeiros 60 segundos, a ativação do serviço resolveu o problema e não ocorreram mais violações. Isso demonstra o poder da estratégia de elasticidade de serviços QoS. Outra coisa importante a se ter em mente é que o sistema só começa a empilhar serviços após 60 segundos devido ao período de cooldown. HealthStack sempre começa em cooldown para evitar ações prematuras.



(a) Elasticidade de Serviços QoS Desabilitada **(b) Elasticidade de Serviços QoS Habilitada**

Figura 6. Medidas de delay e jitter ao longo do tempo da aplicação 4.

6. Conclusão

Os hospitais modernos avançam em direção ao futuro, no caminho do desenvolvimento tecnológico trazidos pela tecnologia IoT. Porém, esse futuro depende de um sistema distribuído com processamento centralizado de dados, o que traz problemas de escalabilidade. Neste contexto, este artigo apresentou um modelo de middleware para ambientes hospitalares 4.0 chamado HealthStack. Diferente de trabalhos relacionados que empregam estratégias isoladas de QoS, HealthStack provê elasticidade de serviços QoS, visando fornecer QoS para aplicações de acordo com a demanda. A contribuição é viabilizada através do monitoramento de desempenho de delay e jitter das aplicações e execução de ações caso limites sejam violados. O estudo incluiu o design e a implementação de um protótipo em uma sala de cirurgia híbrida real. Experimentos com sensores de localização demonstraram a capacidade de HealthStack reduzir jitter de aplicações em até 90,3%.

Além da contribuição técnico-científica, HealthStack também traz contribuições ao hospital e aos pacientes em nome da sociedade. Visando os hospitais do futuro, o modelo oferece estratégias para garantir o desempenho de aplicações em tempo real que processam dados de processos médicos. Mais importante, aplicações futuras em tempo real exigirão confiabilidade nos fluxos de dados de cirurgias para serem executadas corretamente. Por fim, podemos citar algumas limitações que devem ser exploradas em trabalhos futuros. A avaliação do modelo é restrita a dois tipos de dados e a uma quantidade limitada de aplicações. Trabalhos futuros podem derivar desta arquitetura e incluir outros serviços de QoS e tipos de dados diferentes.

Agradecimentos

Os autores gostariam de agradecer a CAPES (Código Financeiro 001), o CNPq (Números de concessão 309537/2020-7), e a FAPERGS (Código 21/2551-0000118-6).

Referências

Aceto, G., Persico, V., and Pescapé, A. (2020). Industry 4.0 and health: Internet of things, big data, and cloud computing for healthcare 4.0. *Journal of Industrial Information Integration*, 18:100129.

Al-Tarawneh, L. A. (2019). Medical grade qos improvement using ieee802.11e wlan protocol. In Al-Masri, A. and Curran, K., editors, *Smart Technologies and Innovation for a Sustainable Future*, pages 229–235, Cham. Springer Int. Publishing.

Bai, T., Lin, J., Li, G., Wang, H., Ran, P., Li, Z., Pang, Y., Wu, W., and Jeon, G. (2019). An optimized protocol for qos and energy efficiency on wireless body area networks. *Peer-to-Peer Networking and Applications*, 12(2):326–336.

Barbosa, V., Ferreira, H., Gomes, L., Gomes, F., Barreto, I., Monteiro, O., and Oliveira, M. (2020). Smartres - uma plataforma iot para monitoramento inteligente em saúde e sua aplicação no contexto da covid-19. In *Anais do XX Simpósio Brasileiro de Computação Aplicada à Saúde*, pages 297–307, Porto Alegre, RS, Brasil. SBC.

Behnel, S., Fiege, L., and Muhl, G. (2006). On quality-of-service and publish-subscribe. In *26th IEEE Int. Conf. on Dist. Comp. Sys. Workshops (ICDCSW'06)*, pages 20–20.

Celdrán, A. H., García Clemente, F. J., Weimer, J., and Lee, I. (2018). Ice++: Improving security, qos, and high availability of medical cyber-physical systems through mobile edge comp. In *2018 IEEE 20th Int. Conf. on e-Health Networking, Applications and Services (Healthcom)*, pages 1–8.

Gatouillat, A., Badr, Y., and Massot, B. (2018). Qos-driven self-adaptation for critical iot-based systems. In Braubach, L., Murillo, J. M., Kaviani, N., Lama, M., Burgueño, L., Moha, N., and Oriol, M., editors, *Service-Oriented Comp. – ICSOC 2017 Workshops*, pages 93–105, Cham. Springer Int. Publishing.

Goyal, R., Patel, R., Bhaduria, H., and Prasad, D. (2020). An energy efficient qos supported optimized transmission rate technique in wbans. *Wireless Pers. C.*, pages 1–26.

Guezguez, M. J., Rekhis, S., and Boudriga, N. (2018). A sensor cloud for the provision of secure and qos-aware healthcare services. *Arabian Journal for Science and Eng.*, 43(12):7059–7082.

Iranmanesh, S. A. and Rizi, F. Y. (2018). Qos provisioning for multiple co-existing body sensor networks. In *Electrical Eng. (ICEE), Iranian Conf. on*, pages 1595–1600.

Khalil, A., Mbarek, N., and Togni, O. (2019). Iot service qos guarantee using qbaiot wireless access method. In Renault, É., Boumerdassi, S., and Bouzefrane, S., editors, *Mobile, Secure, and Prog. Net.*, pages 157–173, Cham. Springer Int. Publishing.

Kitchenham, B. and Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering. Technical report, Keele University.

Mukhopadhyay, A. (2017). Qos based telemedicine technologies for rural healthcare emergencies. In *2017 IEEE Global Humanitarian Tech. Conf. (GHTC)*, pages 1–7.

Munirathinam, S. (2020). Chapter six - industry 4.0: Industrial internet of things (iiot). In Raj, P. and Evangeline, P., editors, *The Digital Twin Paradigm for Smarter Systems and Environments: The Industry Use Cases*, volume 117 of *Advances in Computers*, pages 129–164. Elsevier.

Nanda, P. and Fernandes, R. C. (2007). Quality of service in telemedicine. In *First Int. Conf. on the Digital Society (ICDS'07)*, pages 2–2.

Nielsen, M. A. (2015). *Neural Networks and Deep Learning*, volume 2018. Determination Press San Francisco, CA, USA:.

Oztemel, E. and Gursev, S. (2020). Literature review of industry 4.0 and related technologies. *Journal of Intelligent Manufacturing*, 31(1):127–182.

Samanta, A., Li, Y., and Chen, S. (2018). Qos-aware heuristic scheduling with delay-constraint for wbsns. In *2018 IEEE Int. Conf. on Communications (ICC)*, pages 1–7.

Samanta, A. and Misra, S. (2018). Dynamic connectivity establishment and cooperative scheduling for qos-aware wireless body area networks. *IEEE Transactions on Mobile Comp.*, 17(12):2775–2788.

Sisinni, E., Saifullah, A., Han, S., Jennehag, U., and Gidlund, M. (2018). Industrial internet of things: Challenges, opportunities, and directions. *IEEE Transactions on Industrial Informatics*, 14(11):4724–4734.

Sodhro, A. H., Luo, Z., Sangaiah, A. K., and Baik, S. W. (2019). Mobile edge computing based qos optimization in medical healthcare applications. *Int. Journal of Information Management*, 45:308 – 318.

Venkatesh, K., Srinivas, L., Krishnan, M. M., and Shanthini, A. (2019). Qos improvisation of delay sensitive communication using sdn based multipath routing for medical applications. *Future Generation Computer Sys.*, 93:256 – 265.

Wang, J., Sun, Y., and Ji, Y. (2018). Qos-based adaptive power control scheme for co-located wbans: a cooperative bargaining game theoretic perspective. *Wireless Networks*, 24(8):3129–3139.

Wang, Q. et al. (2019). Enable advanced qos-aware network slicing in 5g networks for slice-based media use cases. *IEEE Transactions on Broadcasting*, 65(2):444–453.

Zitta, T., Neruda, M., Kozak, M., and Vojtech, L. (2018). Multi-channel access to improve qwl in health care services - infrastructure based qos ensurance in iot. In *2018 Int. Symposium ELMAR*, pages 7–10.