UM MECANISMO DE PROVISIONAMENTO DE SERVIÇOS COM CONTROLE DE DEMANDA PARA A INTERNET DAS COISAS

Jasiel das Graças Bahia

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia Elétrica.

Orientador: Miguel Elias Mitre Campista

Rio de Janeiro
Fevereiro de 2018

UM MECANISMO DE PROVISIONAMENTO DE SERVIÇOS COM CONTROLE DE DEMANDA PARA A INTERNET DAS COISAS

Jasiel das Graças Bahia

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Examinada por:

Prof. Miguel Elias Mitre Campista, D.Sc.

Prof. Pedro Braconnot Velloso, Dr.

Prof. Dianne Scherly Varela de Medeiros, D.Sc.

Prof. Alberto Egon Schaeffer-Filho, Ph.D.

RIO DE JANEIRO, RJ – BRASIL
FEVEREIRO DE 2018

Bahia, Jasiel das Graças

Um Mecanismo de Provisionamento de Serviços com Controle de Demanda para a Internet das Coisas/Jasiel das Graças Bahia. – Rio de Janeiro: UFRJ/COPPE, 2018.

XII, 58 p.: il.; 29, 7cm.

Orientador: Miguel Elias Mitre Campista

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2018.

Referências Bibliográficas: p. 54 – 58.

1. Internet das Coisas. 2. CoAP. 3. Serviços. 4. Servidores. I. Campista, Miguel Elias Mitre. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

A Yahuvah Elohim,
a Yeshua, o Ungido,
à minha família na emunah,
à minha esposa Rachel.

Agradecimentos

A Yahuvah Elohim, Criador da vida, que me deu saúde, segurança e toda provisão, me permitindo chegar até aqui.

A Yeshua, o Rei, por ter me dado sabedoria, força e ânimo para prosseguir.

Aos meus pais, que me trouxeram à existência e me conduziram pelo caminho reto, nunca deixando de me incentivar na busca de novas conquistas.

À minha amada esposa Rachel, por seu amor, cuidado e carinho sempre presentes.

Aos meus irmãos, pelo apoio nas orações, pela compreensão e pelos estímulos.

Ao meu estimado orientador, Miguel Campista, pela oportunidade, pelos seus preciosos conhecimentos, pela parceria e compreensão na condução das atividades. Agradeço pelo empenho, profissionalismo e dedicação, sempre focados na obtenção dos melhores resultados.

Aos prezados colegas do GTA, em especial Victor, JB e Silvério, que colaboraram ao longo da jornada, agindo sempre com a maior prestatividade para superarmos as dificuldades.

Aos professores Pedro Braconnot, Dianne Scherly e Alberto Schaeffer pela participação na banca examinadora.

Aos funcionários do Programa de Engenharia Elétrica da COPPE/UFRJ, Daniele e Maurício, pela presteza no atendimento na secretaria do Programa.

À empresa Petrobras Transporte S. A. por ter viabilizado a realização deste trabalho.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

UM MECANISMO DE PROVISIONAMENTO DE SERVIÇOS COM CONTROLE DE DEMANDA PARA A INTERNET DAS COISAS

Jasiel das Graças Bahia

Fevereiro/2018

Orientador: Miguel Elias Mitre Campista

Programa: Engenharia Elétrica

A Internet das Coisas (*Internet of Things* - IoT) desafia a escalabilidade em rede, dado o enorme número de dispositivos interconectados, e cria um cenário fértil para exploração de recursos redundantes para aumentar a robustez dos serviços. Consequentemente, novos protocolos vêm sendo propostos, sendo o CoAP (*Constrained Application Protocol*) um dos principais de IoT para a camada de aplicação. Este trabalho apresenta uma análise pioneira de escalabilidade do CoAP em uma rede composta somente por dispositivos limitados, mostrando a influência dos parâmetros de configuração da rede no desempenho do protocolo. Mais ainda, este trabalho propõe um mecanismo de provisionamento de serviços com controle de demanda, o qual é subdividido em dois componentes principais, sendo o primeiro executado no servidor e o segundo no cliente. O primeiro componente faz o controle de demanda e a seleção de observadores (clientes que se registram para receberem notificações sobre um recurso do servidor), baseando-se no ciclo de trabalho do rádio do servidor e no seu consumo de energia para definir o modo de operação no provimento dos serviços. O segundo componente, por sua vez, faz a seleção e comutação de servidores CoAP. As funções do próprio CoAP são usadas para realizar a comutação dos servidores, seguindo uma lista ordenada de endereços IP obtida a partir de uma infraestrutura central. Essa lista é construída com base nos requisitos da aplicação e fica armazenada no cliente. O mecanismo é avaliado em um simulador específico de IoT (Cooja) e os resultados mostram uma redução significativa no consumo de energia do servidor em comparação ao CoAP tradicional. O mecanismo também aumenta a confiabilidade e a robustez na obtenção de serviços de IoT, além de permitir o balanceamento do consumo de energia entre os servidores.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

A SERVICE PROVISIONING MECHANISM WITH DEMAND CONTROL FOR THE INTERNET OF THINGS

Jasiel das Graças Bahia

February/2018

Advisor: Miguel Elias Mitre Campista

Department: Electrical Engineering

The Internet of Things (IoT) challenges network scalability, given the huge number of connected devices, and also creates a fertile scenario to explore redundant resources for service robustness increasing. Consequently, new IoT protocols have been proposed, being CoAP (*Constrained Application Protocol*) one of the most important for the application layer. In this work, we present a leading scalability analysis of CoAP in a network composed only by constrained devices to show the influence of network configuration parameters on the protocol performance. Moreover, we propose a service provisioning mechanism with demand control divided into two main components. The first component runs at the CoAP server and controls the demand and the selection of observers (clients that subscribe at the server to receive notifications about a resource). This component is based on the radio duty-cycle at the server and its energy consumption for operation mode switching while delivering services. The second component, on the other hand, runs at the client and conducts CoAP servers selection and switching. This component uses CoAP capabilities to switch servers, by following a sorted list of IP addresses obtained from a central infrastructure. This list is built according to the application requirements and is stored at the client. The mechanism is evaluated in a simulator designed for IoT (Cooja) and the results obtained show a significant reduction on energy consumption at the server compared with traditional CoAP. The mechanism also increases the reliability and the robustness of the IoT service provisioning and, furthermore allows energy consumption balance among the servers.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

6LoWPAN	IPv6 over Low-Power Wireless Personal Area Networks, p. 16
AMQP	Advanced Message Queuing Protocol, p. 10
API	Application Programming Interface, p. 8
CoAP	Constrained Application Protocol, p. 4
DTLS	Datagram Transport Layer Security, p. 17
HTTP	Hypertext Transfer Protocol, p. 9
IEEE	Institute of Electrical and Electronics Engineers, p. 15
IETF	Internet Engineering Task Force, p. 15
IP	Internet Protocol, p. 4
IoT	Internet of Things, p. 1
LLN	Low-power and Lossy Networks, p. 3
M2M	Machine-to-Machine, p. 17
MAC	Medium Access Control, p. 15
MQTT	Message Queue Telemetry Transport, p. 9
MTU	Maximum Transmission Unit, p. 15
NGSI	Next Generation Services Interface, p. 8
REST	Representational State Transfer, p. 8
RPL	IPv6 Routing Protocol for Low-Power and Lossy Networks, p. 7
TCP	Transmission Control Protocol, p. 9
UDP	User Datagram Protocol, p. 9
URI	Uniform Resource Identifier, p. 17

Capítulo 1

Introdução

Bilhões de pessoas utilizam a Internet no mundo para aplicações que vão desde navegação *Web* até interação através de redes sociais [2]. Ao mesmo tempo em que o número de pessoas conectadas à Internet cresce, a tecnologia embarcada em dispositivos eletrônicos evolui rapidamente, possibilitando que objetos do cotidiano também se comuniquem e processem dados. Essa evolução leva a um novo paradigma de utilização da Internet, onde os objetos também se conectam e interagem com o ambiente, transformando-se em produtores e consumidores de dados na Internet. A Figura 1.1 ilustra como esse paradigma pode ser aplicado aos objetos de uma casa.

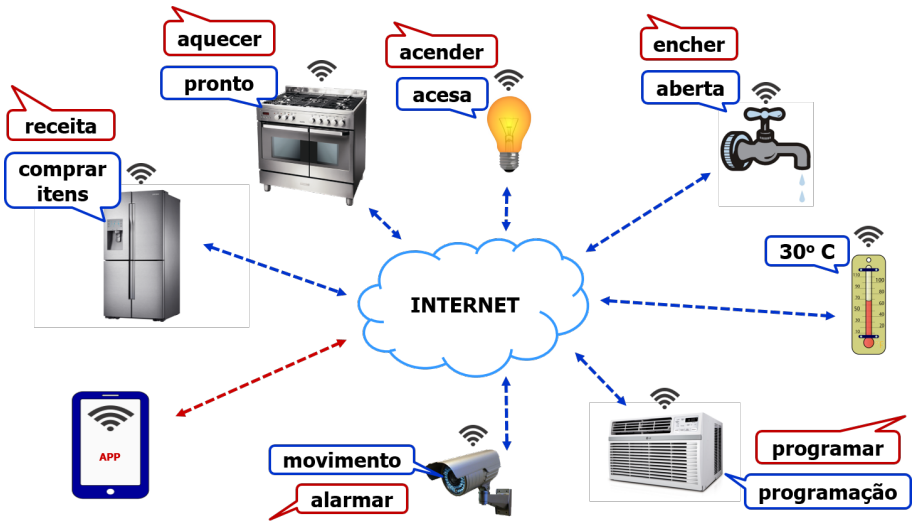


Figura 1.1: Conceito da Internet das Coisas.

O novo padigma da Internet, conhecido como Internet das Coisas (IoT - *Internet of Things*) [3], já é uma realidade na indústria, onde se busca constantemente a maximização da eficiência no uso dos recursos e a permanente disponibilidade dos serviços. A aplicação de IoT na indústria vem sendo apresentada como a quarta revolução industrial sob nomenclaturas como Indústria 4.0, Indústria Conectada e Fábrica Inteligente [4]. A ideia da Indústria 4.0 é integrar e automatizar todos os

processos da cadeia produtiva e logística, possibilitando o seu funcionamento de forma autônoma. A obtenção de informações mais precisas de ambientes e ativos industriais favorece a tomada de decisões mais efetivas, o que pode promover economia, eficiência e segurança [3, 5]. Seguindo esse mesmo conceito, diversas outras áreas já estão aplicando a Internet das Coisas [6]. Alguns exemplos de aplicações de IoT são mostrados na Figura 1.2. Na área pessoal, as pessoas produzem dados sobre si mesmas, permitindo o rastreamento de sua localização, o monitoramento de sua saúde, o acompanhamento do perfil de comportamento e a identificação biométrica. Na área de ambientes inteligentes, observa-se um foco em automação, visando o consumo eficiente de recursos, aumento da segurança e redução de custos. Na área de transporte e logística, por sua vez, o controle da logística pode ser feito através de sistemas integrados para gerenciar aquisições, estoque, transporte, qualidade, etc. Esse gerenciamento pode ser facilitado utilizando-se etiquetas inteligentes para identificação e rastreamento de objetos. O transporte de carga pode ser feito através de condução assistida, onde a localização dos veículos é monitorada. O uso de mapas inteligentes auxilia permite traçar o melhor percurso, melhorando a eficiência e a segurança dos serviços.

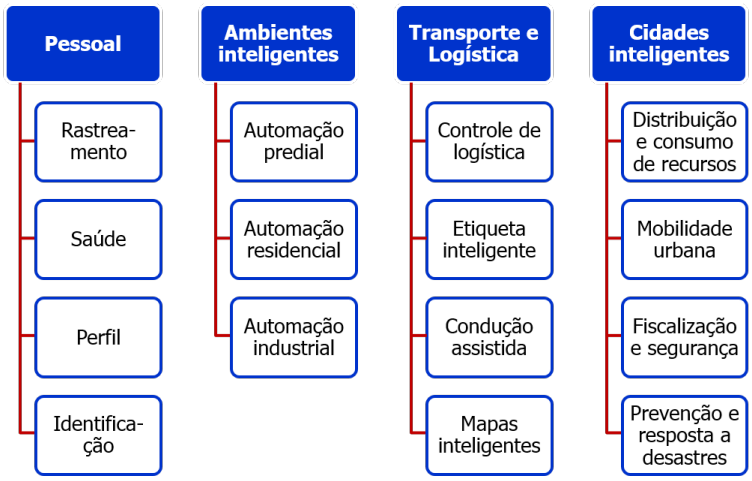


Figura 1.2: Áreas de aplicações para a Internet das Coisas.

Os exemplos de aplicações citados já são uma realidade atualmente e mostram como a Internet das Coisas é uma forte tendência. Porém, isso tudo implica numa imensa quantidade de dispositivos conectados, o que nos leva ao grande desafio de conciliar a grande quantidade de dados gerados e as limitações dos dispositivos em redes de IoT. Dessa forma, para desenvolver soluções adequadas para superar esse desafio, é necessário atentar para alguns aspectos, tais como: escalabilidade da rede, consumo eficiente de energia e disponibilidade dos serviços. Com relação ao consumo de energia, uma medida importante é controlar o ciclo de trabalho (*duty-cycle*) do sistema de comunicação, pois esse é o que mais demanda energia do dispositivo. Isso

faz com que os protocolos empregados na Internet convencional não sejam adequados para uso em dispositivos de IoT [3], já que o consumo de energia não foi um pré-requisito de projeto. Como parte do desenvolvimento de novos mecanismos, é importante que a análise de desempenho dos mesmos considere as limitações dos dispositivos de IoT em termos de processamento, armazenamento e energia. Além dessas restrições, a imensa quantidade de dispositivos conectados [7] provoca um aumento substancial no fluxo de dados e desafia o potencial de escalabilidade das redes. Os mecanismos que não consideram essas restrições podem provocar o rápido esgotamento dos recursos da rede e da energia dos dispositivos, comprometendo a disponibilidade dos serviços. Por outro lado, a multiplicidade de dispositivos conectados à Internet contendo diversos sensores oferece grande potencial de redundância de recursos. Essas redundâncias podem ser usadas para aumentar a robustez das redes de IoT, garantindo a disponibilidade dos serviços mesmo diante de instabilidades na rede, bastante comuns em redes LLN (*Low-power and Lossy Networks*) [8].

Uma arquitetura básica da Internet das Coisas (Figura 1.3) prevê redes compostas por dispositivos limitados (LLN) que assumem o papel de servidor ou cliente de acordo com a aplicação e a complexidade do sistema. No papel de servidor, os

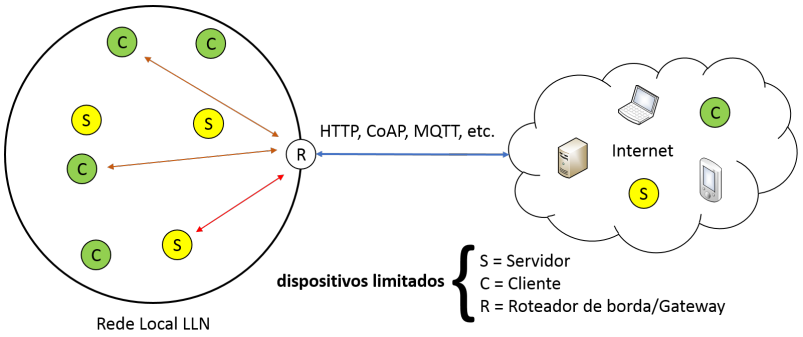


Figura 1.3: Arquitetura típica de IoT.

dispositivos disponibilizam seus recursos para os dispositivos clientes na rede local e na Internet. No papel de cliente, o dispositivo consulta os servidores para obter as informações necessárias para a sua aplicação. Considera-se que, para cada rede local, exista a figura do roteador de borda, responsável por conectar os dispositivos de sua rede entre si e à Internet, divulgando os recursos disponíveis dos servidores e fazendo a tradução entre os protocolos. Esse é o cenário básico que deve ser considerado na proposta de novas soluções para redes de IoT que visem conciliar o fluxo de dados e as limitações dos dispositivos.

Na literatura, observa-se uma grande busca pelo consumo eficiente de energia, por conta da limitação de energia dos dispositivos de IoT. Algumas soluções propostas buscam reduzir o consumo de energia através do aumento da eficiência no

roteamento [9, 10], outras por meio da redução de pacotes transmitidos [11] e do controle adaptativo do ciclo de trabalho do dispositivo [12]. Os trabalhos mencionados, porém, não aliam a eficiência energética à garantia de confiabilidade e disponibilidade dos serviços de IoT na rede. Com relação ao provisionamento de serviços, comumente, os trabalhos consideram uma arquitetura centralizada, onde o gerenciamento dos serviços fica à cargo de um dispositivo concentrador que atua como um intermediário na comunicação entre os dispositivos [13–15]. Uma desvantagem dessa arquitetura é que uma falha no dispositivo concentrador pode comprometer a disponibilidade dos serviços. Dessa forma, uma saída é habilitar o cliente a se comunicar diretamente com os servidores e a realizar a comutação entre eles, permitindo a seleção dos servidores e a obtenção dos serviços de acordo com os requisitos da aplicação. O conceito de comutação é comum na camada de rede, para fazer o roteamento dos pacotes da origem até o destino, mas neste trabalho o conceito de comutação foi usado na camada de aplicação para aumentar a robustez na obtenção dos serviços. Com isso, os critérios de comutação de servidores consideram os requisitos de cada aplicação, sendo possível aproveitar a diversidade de servidores para aumentar a disponibilidade dos serviços. Dentre os principais protocolos desenvolvidos para redes LLN, o CoAP (*Constrained Application Protocol*) [16], da camada de aplicação, destaca-se por permitir a comunicação por IP (*Internet Protocol*) entre cliente e servidor via Internet e por possuir funcionalidades que favorecem a eficiência energética no provisionamento de serviços de IoT. Por isso, este trabalho utilizou o protocolo CoAP como base para o desenvolvimento do mecanismo proposto. Na análise de desempenho de novas propostas para IoT, é importante considerar cenários com redes compostas por dispositivos limitados. De modo geral, os protocolos são avaliados qualitativamente ou por meio de experimentos que não consideram as limitações dos dispositivos [17–19].

Este trabalho visa melhorar o desempenho e a escalabilidade de servidores CoAP no provisionamento de serviços de IoT. Além disso, este trabalho visa aumentar a confiabilidade e a robustez na obtenção de serviços de IoT, através do aumento da disponibilidade de servidores. O primeiro objetivo deste trabalho é mostrar a influência dos parâmetros de configuração da rede no desempenho do servidor CoAP, considerando como métricas carga da rede, taxa de perdas, quantidade de pacotes CoAP gerados e consumo de energia. O segundo e principal objetivo é propor um mecanismo de provisionamento de serviços de IoT com controle de demanda. Esse mecanismo é constituído por dois componentes, sendo o primeiro executado no servidor e o segundo no cliente. O primeiro componente faz o controle de demanda e a seleção de observadores, baseando-se no ciclo de trabalho do rádio do servidor e no seu consumo de energia para estabelecer dinamicamente o modo de provimento dos serviços. Os observadores são clientes que se registram no servidor para receberem

as notificações a respeito de um recurso, sem a necessidade de realizar novos pedidos de atualização da informação. O controle de demanda regula o consumo de energia no servidor e garante a disponibilidade dos serviços para os clientes observadores. O segundo componente faz a seleção e comutação de servidores CoAP, baseando-se numa lista ordenada de endereços IP construída e fornecida por uma infraestrutura central de acordo com os requisitos da aplicação do cliente. A lista de servidores fica armazenada no cliente e a comutação segue a ordem de prioridade definida, de modo a permitir a obtenção dos serviços necessários à aplicação. O mecanismo proposto aumenta a confiabilidade e a robustez na obtenção de serviços de IoT, além de permitir o balanceamento do consumo de energia entre os servidores CoAP.

O mecanismo proposto é avaliado através de experimentos realizados no Cooja [20], que é o simulador da plataforma de desenvolvimento do Contiki OS, desenvolvido para o teste de redes LLN. Os experimentos consideraram um cenário no qual os dispositivos possuem recursos limitados e interagem entre si para realizar funções de controle, podendo haver recursos redundantes. O mecanismo proposto é aplicável a qualquer cenário de IoT onde se queira garantir a disponibilidade de um dado servidor para um grupo específico de clientes, controlando a demanda e, por conseguinte, o consumo de energia. Além disso, o mecanismo é adequado para cenários onde a oferta de diversos recursos permite o aumento da disponibilidade dos serviços necessários às aplicações dos clientes. Os resultados dos experimentos mostram que o uso do CoAP com o mecanismo proposto reduz significativamente o consumo de energia do servidor quando comparado ao uso tradicional do CoAP, sendo esse um requisito fundamental para o aumento do tempo de vida do servidor. Com o uso do mecanismo, o servidor consegue garantir a disponibilidade dos serviços para os clientes observadores de acordo com as especificações. Além disso, os resultados mostram também que o mecanismo promove o aumento da robustez dos serviços e permite o balanceamento do consumo de energia entre os servidores, mediante a seleção e a comutação inteligente dos servidores.

A principal contribuição deste trabalho é a proposta de um novo mecanismo de provisionamento de serviços com controle de demanda. O mecanismo proposto melhora o desempenho e a escalabilidade dos servidores CoAP, mantendo sob controle o consumo de energia e garantindo a disponibilidade para clientes observadores. Além disso, o mecanismo aumenta a confiabilidade no provisionamento dos serviços através da comutação de servidores. Finalmente, ressalta-se que este trabalho apresenta uma proposta inédita para economia de energia em redes com servidores CoAP e a proposta de controle de demanda, juntamente com a análise de servidores CoAP, rendeu a publicação de um artigo em congresso nacional [21]. Ademais, este é o primeiro trabalho a propor um mecanismo que permita a obtenção de serviços pelo próprio cliente CoAP através da comutação de servidores.

O restante deste trabalho está organizado da seguinte forma. No Capítulo 2, os trabalhos relacionados são listados com ênfase nos protocolos de comunicação para IoT. O Capítulo 3 apresenta uma arquitetura típica da Internet das Coisas e o funcionamento do CoAP. No Capítulo 4, o funcionamento do mecanismo de provisionamento de serviços com controle de demanda é apresentado e, a seguir, no Capítulo 5, as condições de análise do mecanismo são descritas e os resultados obtidos nos experimentos são discutidos. Finalmente, o Capítulo 6 conclui este trabalho com uma visão panorâmica dos destaques e sugestões de trabalhos futuros.

Capítulo 2

Revisão Bibliográfica

O desenvolvimento de protocolos adaptados a dispositivos com recursos limitados tem sido objeto de grande interesse no meio científico. Neste capítulo, são apresentadas algumas propostas e análises de desempenho de protocolos de comunicação para IoT.

2.1 Consumo eficiente de energia

Diversas propostas de mecanismos e protocolos para a Internet das Coisas visam promover o consumo eficiente de energia, por conta da limitação desse recurso em dispositivos de IoT. Alam et al. [22] utilizam o simulador QualNet para comparar o desempenho de alguns protocolos de roteamento, sendo uma das métricas o consumo de energia. Eles chegam à conclusão de que os protocolos analisados consomem uma quantidade de energia tal que os torna inadequados para uso em dispositivos limitados. Já Iova et al. [10] focam na identificação de dispositivos que possam representar gargalos no caminho de roteamento. Eles definem uma nova métrica de roteamento associada ao tempo de vida restante do dispositivo para um protocolo de roteamento desenvolvido para redes LLN, o RPL (*IPv6 Routing Protocol for Low-Power and Lossy Networks*) [23]. A nova métrica é, então, usada para construir o caminho de roteamento. O objetivo principal é construir uma rota que evite os dispositivos com menos energia, permitindo o balanceamento do consumo de energia entre as rotas. Embora a ideia seja interessante, a sobrecarga para estimar o tempo de vida e recalcular as rotas pode degradar o desempenho das redes LLN. Semelhantemente, Barbato et al. [9] buscam reduzir o consumo de energia através do aumento da eficiência no roteamento. Eles também propõem uma nova métrica baseada no consumo de energia para o RPL. Enquanto isso, outros autores [12, 24] buscam o consumo eficiente através do controle adaptativo do ciclo de trabalho do dispositivo para regular o tráfego na rede. A análise das propostas, porém, é feita ferramentas de simulação que não foram desenvolvidas para reproduzir as condições

encontradas em redes LLN. Na camada de aplicação, Jan et al. [11] propõem uma extensão do CoAP para promover a redução de pacotes transmitidos e, consequentemente, do consumo de energia pelos dispositivos. Embora os trabalhos mencionados busquem a economia de energia, nota-se que não há uma preocupação em aliar o consumo eficiente à garantia de confiabilidade e disponibilidade dos serviços de IoT na rede.

2.2 Provisionamento de serviços de IoT

Em termos de soluções para provisionamento de serviços de IoT, comumente, os trabalhos consideram uma arquitetura centralizada (Figura 2.1), onde o gerenciamento dos serviços fica à cargo de um dispositivo concentrador (*broker*) que atua como um intermediário na comunicação entre os dispositivos [13–15].

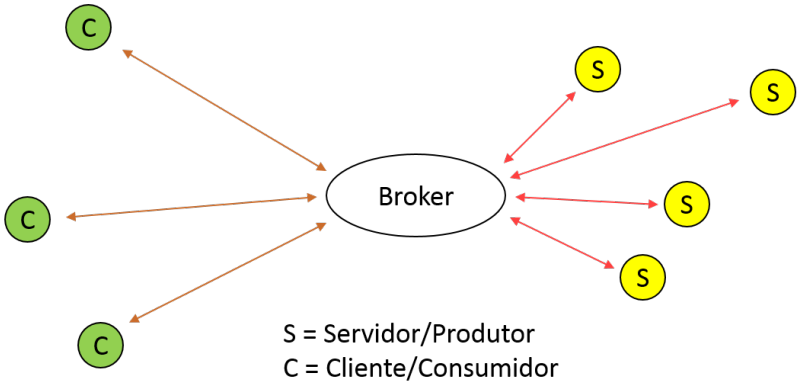


Figura 2.1: Provisionamento de serviços com arquitetura centralizada.

Há arquiteturas que focam na mineração de dados, abstraindo os protocolos de comunicação utilizados pelos dispositivos. Por exemplo, o Orion [13] não é um protocolo de comunicação, mas sim a implementação de um conjunto de APIs REST que funciona como um *content broker*, ou seja, como um concentrador de conteúdo que coleta, gerencia e combina informações de contexto para compor serviços. O Orion é um dos componentes da arquitetura FIWARE [25], a qual em sua camada mais baixa faz a coleta dos dados através da tradução dos protocolos de comunicação (HTTP, MQTT) para uma linguagem compatível com NGSI [26] (que utiliza métodos e códigos do HTTP). Nessa camada, cada protocolo é traduzido por um IoT-Agent (um bloco tradutor), e a ferramenta permite que sejam implementados novos IoT-Agent para outros protocolos (p. ex., CoAP). O modelo de arquitetura centralizada não permite o contato direto entre consumidor e produtor. Os clientes entram em contato apenas com o *broker* e não conseguem se comunicar diretamente com os servidores (produtores de conteúdo), ficando restritos à oferta de serviços do servidor central. Uma desvantagem dessa arquitetura é que uma falha no dispositivo

concentrador pode comprometer a disponibilidade dos serviços. Dessa forma, uma saída é habilitar o cliente a se comunicar diretamente com os servidores e a realizar a comutação entre eles, permitindo a seleção dos servidores e a obtenção dos serviços de acordo com os requisitos da aplicação. O conceito de comutação é comum na camada de rede, para fazer o roteamento dos pacotes da origem até o destino. Neste trabalho, porém, o conceito de comutação foi usado na camada de aplicação para aumentar a robustez na obtenção dos serviços. Com isso, os critérios de comutação de servidores consideram os requisitos de cada aplicação, sendo possível aproveitar a diversidade de servidores para aumentar a disponibilidade dos serviços. Dentre os principais protocolos desenvolvidos para redes LLN, o CoAP [16], da camada de aplicação, destaca-se por permitir a comunicação por IP entre cliente e servidor via Internet e por possuir funcionalidades que favorecem a eficiência energética no provisionamento de serviços de IoT. Por isso, este trabalho utilizou o protocolo CoAP como base para o desenvolvimento do mecanismo proposto. A capacidade de comunicação direta com os servidores permite o cliente CoAP ter acessos aos serviços sem a intermediação de um *broker*. Isso possibilita maior flexibilidade no atendimento dos requisitos do cliente e também ajuda a manter a continuidade dos serviços. Além do CoAP, existem ainda outros protocolos de camada de aplicação que estão sendo propostos para a Internet das Coisas. Alguns trabalhos que comparam estes protocolos com o CoAP são apresentados a seguir.

2.3 Comparação do CoAP com outros protocolos de camada de Aplicação para IoT

O HTTP (*Hyper Text Transfer Protocol*) é um protocolo de aplicações *Web* já estabelecido, porém ele não foi desenvolvido para ser usado nos dispositivos limitados da Internet das Coisas. O HTTP utiliza o TCP (*Transmission Control Protocol*) para prover comunicação confiável, já o CoAP oferece confiabilidade na própria camada de aplicação, fazendo uso do UDP (*User Datagram Protocol*) na camada de transporte. A capacidade de processamento exigida, o consumo de energia elevado e o tamanho do cabeçalho, tornam o HTTP um protocolo não recomendado para redes LLN [27, 28]. O modelo de interação do HTTP é do tipo cliente/servidor, orientado a conexão, permitindo o endereçamento por IP entre cliente e servidor. A Tabela 2.1 mostra o resumo comparativo entre CoAP e HTTP. Mais detalhes sobre as características do CoAP são apresentados no Capítulo 3.

Os dois protocolos de camada de aplicação mais difundidos para dispositivos limitados são o CoAP e o MQTT (*Message Queue Telemetry Transport*). O MQTT [14], desenvolvido pela IBM, é um protocolo do tipo produtor/assinante que roda sobre

Tabela 2.1: Resumo comparativo entre CoAP e HTTP.

Característica	**HTTP**	**CoAP**
Arquitetura	cliente/servidor	cliente/servidor
Cabeçalho	indefinido	4 bytes
Transporte	TCP	UDP
Redes LLN	não recomendado	recomendado
Endereçam. IP	disponível	disponível
Confiabilidade	TCP	CoAP

TCP, diferentemente do CoAP, que faz uso do UDP. O uso do TCP não é indicado para redes LLN e dispositivos mais limitados, visto que exige mais memória e processamento [8]. Além disso, o MQTT precisa manter uma conexão aberta com o *broker* (concentrador), o que pode ser bastante custoso num ambiente com alta taxa de perdas. O MQTT não suporta diretamente serviços *Web*, pois, diferentemente do CoAP, não se consegue rastrear as transações de pedido/resposta entre cliente e servidor fim-a-fim. O MQTT foi proposto para aplicações onde o principal objetivo é coletar e concentrar os dados num servidor central para disponibilizá-los aos assinantes. Para publicação dos serviços e acesso aos mesmos, o MQTT utiliza o *broker*, um dispositivo concentrador que coleta os dados dos recursos e os envia para a infraestrutura do servidor. Dessa forma, o MQTT não permite a comunicação fim-a-fim entre cliente e servidor. Observa-se ainda que os endereços utilizados pelo MQTT são definidos por longas cadeias de caracteres, o que não é compatível com a tecnologia de rede proposta para dispositivos limitados IEEE 802.15.4 [29], onde o tamanho máximo dos pacotes é de 128 bytes. Uma alternativa ao MQTT original é o MQTT-SN [30], que utiliza o UDP e tenta prover uma abstração para comunicação assíncrona.

Outro protocolo sugerido para aplicações de IoT é o AMQP (*Advanced Message Queuing Protocol*) [15]. O AMQP é um protocolo desenvolvido pela JPMorgan Chase que suporta interações do tipo pedido/resposta e do tipo produtor/assinante. O protocolo foi desenvolvido para comunicação corporativa confiável entre servidores, tendo nascido no âmbito das aplicações bancárias. O protocolo exige a criação de um tópico para permitir a interação entre produtores e consumidores. A partir daí, as transações dos produtores são enfileiradas por tópico para serem processadas pela infraestrutura central e encaminhadas aos consumidores dos respectivos tópicos. O AMQP possui um cabeçalho de tamanho fixo igual a 8 bytes e utiliza o TCP na camada de transporte. Sendo assim, o AMQP é orientado a conexão, assim como o MQTT e o HTTP. A Tabela 2.2 mostra o resumo comparativo entre CoAP, MQTT e AMQP.

A comparação entre os protocolos para IoT deve considerar o cenário de aplicação, visto que um protocolo pode se sair melhor em um cenário específico

Tabela 2.2: Resumo comparativo entre protocolos para IoT.

Característica	**CoAP**	**MQTT**	**MQTT-SN**	**AMQP**
Arquitetura	cliente/servidor	produtor/assinante	produtor/assinante	produtor/assinante
Cabeçalho	4 bytes	2 bytes	2 bytes	8 bytes
Transporte	UDP	TCP	UDP	TCP
Endereçam. IP	disponível	indisponível	indisponível	indisponível

e pior em outros cenários. Yassein et al. [31] mostram que um dos pontos fortes do MQTT é a redução de uso de banda, minimizando o tráfego de dados na rede. No caso de tráfego intenso na rede, o MQTT consegue maior vazão e menor latência do que o CoAP. Do mesmo modo, Naik [32] sugere que o MQTT é mais apropriado para redes com grande número de dispositivos sendo monitorados e controlados por um servidor central. Sendo assim, o MQTT não é a opção preferencial para comunicação direta entre dispositivos limitados. No mesmo trabalho evidencia-se que o HTTP é um protocolo orientado à conexão e é apropriado para aplicações mais complexas, exigindo maior capacidade de processamento. Naik mostra que, embora o cabeçalho do MQTT seja pequeno em relação ao do CoAP, a necessidade de conexão TCP resulta em maior sobrecarga. Enquanto isso, o HTTP é o protocolo com maior sobrecarga, tendo em vista que não foi desenvolvido com IoT em mente. O trabalho conclui também que o CoAP consome, em média, menos energia que o MQTT quando submetido às mesmas circunstâncias. Hedi et al. [33] comparam as principais características do MQTT e do CoAP. Enquanto no MQTT cada dispositivo que se conecta ao servidor central deve decidir se age como produtor ou consumidor, no CoAP, o dispositivo pode se comportar como cliente e servidor simultaneamente. O MQTT é um protocolo para comunicação de muitos produtores para muitos clientes através de um servidor central. Já o CoAP é fundamentalmente para comunicação de um para um, embora o protocolo permita a comunicação de um para muitos e de muitos para muitos via multicast. Finalmente, Hedi et al. sugerem que o CoAP é recomendado para aplicações onde a resposta em tempo real não é um requisito mandatório.

Dentre os protocolos de camada de Aplicação mencionados, o CoAP é o único ao qual se aplica plenamente o mecanismo proposto neste trabalho. Por essa razão, não se comparou o funcionamento do mecanismo com outros protocolos. O CoAP, comparado ao MQTT e ao AMQP, por exemplo, possui vantagens como endereçamento IP e comunicação fim-a-fim entre cliente e servidor. No caso do HTTP, além de este não ser adequado para redes LLN, não permite ao cliente se tornar observador de um recurso como no CoAP. Neste caso, o cliente observador é aquele que, depois de se registrar como observador no servidor, passa a receber notificações e não precisa gerar pedidos adicionais para obter atualizações do recurso de interesse.

2.4 Análise de Desempenho de Protocolos para IoT

Com relação à análise de desempenho de protocolos para IoT, Sutaria et al. [27] comparam o HTTP e o CoAP em termos de tamanho de pacote e consumo de energia por pedido. No artigo, mostra-se que, para um servidor HTTP implementado no Contiki OS [20], quando um pedido é feito com o CoAP, usa-se 154 bytes; quando feito com o HTTP, usa-se 1451 bytes, consumindo-se 0,774 mW e 1,333 mW, respectivamente. Já Colitti et al. [28] mostram que o CoAP gasta menos tempo na transmissão do pedido e consome menos energia que o HTTP. O experimento foi realizado utilizando um computador como cliente CoAP ou HTTP e um dispositivo com recursos limitados como servidor CoAP ou HTTP, respectivamente.

Tabela 2.3: Comparativo de desempenho entre CoAP e HTTP.

Métrica	**HTTP**	**CoAP**
Tamanho de pacote do pedido	1451 bytes	154 bytes
Consumo de energia por pacote	1,333 mW	0,774 mW
Tempo pedido/resposta	1,8 s	184 ms

Thangavel et al. [18] analisam o desempenho do MQTT e do CoAP em termos de tempo entre pedido e resposta e consumo de banda, quando usados em conjunto com o *middleware* proposto para prover interoperabilidade. Os resultados revelam que mensagens MQTT possuem menor atraso do que mensagens CoAP quando há baixa taxa de perda e que a situação se inverte quando há alta taxa de perda. Quando o tamanho da mensagem é pequeno e a taxa de perda é menor ou igual a 25%, o CoAP gera menos tráfego adicional para assegurar a confiabilidade da mensagem. Em outro trabalho, Mun et al. [17] comparam MQTT, CoAP e outros protocolos para ajudar programadores a escolherem o protocolo mais adequado para aplicações em dispositivos com recursos limitados. A avaliação é feita comparando tempo entre pedido e resposta, energia consumida, uso de memória e carga de processamento. Os resultados mostraram que, nas condições de teste, o CoAP apresentou melhor desempenho que o MQTT e o MQTT-SN, especialmente quando o tamanho da mensagem é menor que 1024 bytes. Quando a o tamanho da mensagem é maior que esse valor, o tempo de transmissão do CoAP e a energia consumida aumentam devido à fragmentação do pacote pelo protocolo. Assim, Mun et al. concluem que o MQTT é preferível quando o tamanho dos pacotes a serem transmitidos é grande. Mijovic et al. [34] mostram que CoAP é um protocolo mais eficiente que o MQTT e garante um tempo de pedido e resposta menor nos cenários de IoT considerados. Eles evidenciam ainda que o perfil de qualidade de serviço escolhido no MQTT afeta bastante o desempenho do protocolo.

Tabela 2.4: Comparativo de desempenho entre CoAP e MQTT.

Métrica	**CoAP**	**MQTT**
Tempo pedido/resposta (msg $<$ 1024 bytes)	Menor	Maior
Tempo pedido/resposta (msg $>$ 1024 bytes)	Maior	Menor
Consumo de energia (msg $<$ 1024 bytes)	Menor	Maior
Consumo de energia (msg $>$ 1024 bytes)	Maior	Menor
Sobrecarga para garantir confiabilidade	Baixa	Alta

Naik [32] compara os protocolos CoAP, HTTP, MQTT e AMQP qualitativamente em termos de tamanho de mensagem, consumo de energia, uso de banda e latência. A sua análise conclui que o CoAP é o mais adequado para consumo eficiente dos recursos dos dispositivos em redes LLN. Para Naik, um dos fatores determinantes para o melhor desempenho do CoAP é o uso do UDP no lugar do TCP, o que reduz a sobrecarga na rede e o consumo de energia do dispositivo. Kovatsch et al. [35] apresentam uma implementação do CoAP para o Contiki OS, e o ContikiMAC, um gerenciador do ciclo de trabalho do rádio do dispositivo. O mecanismo é avaliado pela variação no consumo de energia e no tempo de resposta, mostrando que o ContikiMAC reduz o consumo de energia, mas aumenta a latência da rede. Zhang et al. [36] analisam o desempenho do ContikiRPL, uma implementação do protocolo de roteamento RPL para o Contiki OS, observando o seu comportamento em diferentes arranjos da rede. Assim como neste trabalho, Zhang et al. consideram uma rede de IoT LLN contendo dispositivos com recursos limitados.

Pelo levantamento feito, não foi identificado nenhum trabalho de análise de desempenho de servidores CoAP em redes de IoT compostas por dispositivos com recursos limitados que tivesse a abrangência aqui apresentada. Além disso, não foram encontrados outros trabalhos que mencionassem mecanismos de controle de demanda para servidores CoAP ou de comutação de servidores para clientes CoAP, sendo essa a principal contribuição deste trabalho.

Capítulo 3

Internet das Coisas

Este capítulo introduz as características principais da Internet das Coisas, mostrando a pilha de protocolos que está sendo desenvolvida para dispositivos limitados. Além disso, o CoAP e suas principais funcionalidades é apresentado.

3.1 Arquitetura Típica da Internet das Coisas

Na literatura, existem diversas definições para a Internet das Coisas. Dentre elas, destaca-se a apresentada por Atzori et al. [6], cuja abrangência vai além dos aspectos básicos de endereçamento único, interconectividade e uso de protocolos padronizados. Atzori et al. veem a Internet das Coisas como um sistema que abrange o interfaceamento entre entidades físicas e virtuais, comunicação inteligente dos dispositivos com o ambiente e com as pessoas, mineração de dados e composição de serviços. A Internet das Coisas objetiva, em especial, a interoperabilidade entre os dispositivos de redes LLN [8] e as diversas aplicações da Internet convencional. O que caracteriza uma rede LLN é o fato de ela ser composta somente por dispositivos limitados em capacidade de processamento, armazenamento e energia (fornecida por uma bateria). Esses dispositivos tipicamente se comunicam numa rede sem fio, utilizando um rádio de baixa potência. Dessa forma, a rede é altamente susceptível a perdas, devido a interferências, ruídos e outros fatores.

Uma arquitetura básica da Internet das Coisas (Figura 3.1) prevê redes compostas por dispositivos limitados que assumem o papel de servidor ou cliente de acordo com a aplicação e a complexidade do sistema. No papel de servidor, os dispositivos disponibilizam seus recursos para os dispositivos clientes na rede local e na Internet. No papel de cliente, o dispositivo consulta os servidores para obter as informações necessárias para a sua aplicação. Considera-se que, para cada rede local, exista a figura do roteador de borda, responsável por conectar os dispositivos de sua rede entre si e à Internet, divulgando os recursos disponíveis dos servidores e fazendo a tradução entre os protocolos (p. ex., HTTP/CoAP). Todos os dispositivos podem

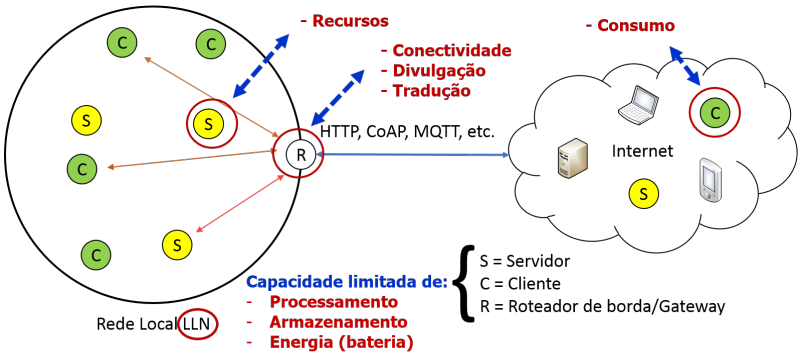


Figura 3.1: Arquitetura básica de IoT.

se comunicar entre si, porém o roteador de borda é quem define o roteamento para a mensagem da origem até o destino.

3.2 Pilha de protocolos

Em face das limitações de energia (uso de bateria), processamento e memória dos dispositivos de IoT e do nível de escalabilidade exigido, as soluções empregadas na Internet convencional não são compatíveis com os novos cenários introduzidos pelas redes LLN. Em razão disso, grupos de trabalho formados pelo IEEE (*Institute of Electrical and Electronics Engineers*) e pelo IETF (*Internet Engineering Task Force*) têm por objetivo desenvolver soluções alinhadas às características dos dispositivos de IoT, buscando, ao mesmo tempo, garantir a interoperabilidade com os padrões da Internet já estabelecidos. Sendo assim, uma nova pilha de protocolos (Tabela 3.1) vem sendo desenvolvida com o objetivo de fomentar aplicações futuras de IoT [8, 37]. Observando a Tabela 3.1 de baixo para cima, as principais características dos

Tabela 3.1: Protocolos da Internet Convencional e da Internet das Coisas.

Camadas	**Protocolos Convencionais**	**Protocolos de IoT/LLN**
Aplicação	HTTP,FTP,SMTP,etc.	CoAP
Transporte	TCP,UDP	UDP
Rede/Adaptação	IPv4,IPv6	RPL/6LoWPAN
Enlace	IEEE 802.3,802.11	IEEE 802.15.4

protocolos são apresentadas a seguir:

- Comunicação sem fio de baixa potência, com consumo eficiente de energia, nas camadas física e de enlace (MAC - *Medium Access Control*) é realizada através do protocolo IEEE 802.15.4 [29]. Nesse cenário, o dispositivo utiliza no máximo 127 bytes para transmissão de dados nas camadas superiores da pilha. Esse valor é muito menor que a máxima unidade de transmissão (MTU

- *Maximum Transmission Unit*) de 1280 bytes exigida pelo IPv6. O rádio escuta o meio continuamente, demodulando o sinal recebido e esperando pelo preambulo definido para recepção dos dados.

- A camada de adaptação 6LoWPAN (*IPv6 over Low-Power Wireless Personal Area Networks*) [38, 39] faz a adaptação do IEEE 802.15.4 para uso com IPv6, realizando fragmentação e desfragmentação de pacotes, compressão do cabeçalho e outras funções.

- O roteamento sobre 6LoWPAN é realizado pelo protocolo de roteamento para redes LLN, o RPL. É possível configurar o protocolo para melhor adequação aos requisitos da aplicação de IoT, através de perfis pré-definidos. Um protocolo de roteamento dedicado a redes LLN é necessário devido às limitações dos dispositivos e a outras características da rede, tais como topologia em malha de múltiplos saltos sujeita a constantes mudanças.

- Na camada de transporte, os protocolos têm como função prover comunicação fim-a-fim. Porém, para cumprir essa função com confiabilidade, exige-se um número de pacotes e um consumo de energia que em geral não são adequados para redes LLN. Devido à falta de padronização de novos protocolos de transporte para dispositivos limitados, o uso do UDP em conjunto com o controle de retransmissões na camada de aplicação tem-se mostrado uma boa opção. Essa abordagem visa minimizar o consumo de energia e a sobrecarga na rede.

- O CoAP é o protocolo encarregado pela comunicação na camada de aplicação, provendo interoperabilidade entre as aplicações. O uso direto de protocolos baseados na arquitetura REST, tal como o HTTP, não se adequa às limitações dos dispositivos, sendo necessária uma solução mais simples. Embora o CoAP não seja o único protocolo de IoT proposto para camada de aplicação (como visto no Capítulo 2), o CoAP é o mais amplamente divulgado como referência para a Internet das Coisas e redes LLN.

Os sistemas desenvolvidos para a Internet das Coisas tipicamente adotam uma arquitetura orientada a serviço [5], onde a camada de aplicação tem a função básica de apresentar a informação ao usuário através de uma interface amigável. De acordo com os requisitos da aplicação, o sistema deve realizar a descoberta e composição de serviços, fazendo uso da infraestrutura da rede de IoT para atender às solicitações. Os pedidos podem ser gerados por meio de protocolos de serviços *Web* adaptados às restrições dos dispositivos de IoT, como é o caso do CoAP.

O mecanismo proposto neste trabalho foi desenvolvido tendo como alvo dispositivos que utilizam a pilha de protocolos de IoT apresentada nesta seção. Dentre

esses protocolos, o CoAP é o protocolo fundamental para o pleno funcionamento do mecanismo proposto.

3.3 Constrained Application Protocol (CoAP)

O CoAP é um protocolo de comunicação desenvolvido para dispositivos com recursos limitados e redes LLN [16]. Esses dispositivos possuem baixa capacidade de processamento e armazenamento e as redes LLN, por sua vez, apresentam alta taxa de erro e vazão típica de 10 kb/s. O CoAP foi projetado para aplicações M2M (*Machine-to-Machine*), como automação industrial. O CoAP fornece ainda um modelo de interação pedido/resposta fim-a-fim que suporta descoberta de serviços e inclui conceitos da *Web* como URI (*Uniform Resource Identifier*) e tipos de dados da Internet. Além disso, o CoAP pode ser traduzido para o HTTP, oferecendo suporte a multicast e comunicação assíncrona, com baixa sobrecarga e simplicidade para ambientes com recursos limitados. O CoAP pode funcionar utilizando *proxy* e *cache*, permitindo o acesso aos recursos CoAP via HTTP de maneira uniforme. A segurança no CoAP é estabelecida por meio da camada de transporte DTLS (*Datagram Transport Layer Security*) [16].

O modelo de interação do CoAP é similar ao cliente/servidor do HTTP, porém interações M2M tipicamente possibilitam a um dispositivo agir simultaneamente como cliente e como servidor. Uma requisição CoAP é enviada pelo cliente solicitando uma ação (de acordo com o código do método) sobre um recurso (identificado pelo URI) do servidor. O servidor então responde com o código de resposta, podendo incluir uma representação do recurso (p. ex., valor da temperatura). Essa troca de mensagens é exemplificada na Figura 3.2, onde cada mensagem carrega informações como: tipo de mensagem (p. ex., CON, ACK), tipo de pedido (p. ex., GET), URI do recurso (p. ex., /temperatura), identificador da mensagem (p. ex., 0xbc90), identificador da transação pedido/resposta (p. ex., 0x71), código de resposta (p. ex., 2.05), conteúdo solicitado (p. ex., 22,5 C), etc. A descoberta de serviços é feita com um único pedido endereçado a um URI específico do servidor, conforme definido pelo protocolo CoAP. O servidor responde a esse pedido enviando uma lista com os recursos que ele oferece. O CoAP pode ser visto como um protocolo de duas camadas (Tabela 3.2): uma camada de mensagens, usada para lidar com as interações assíncronas sobre UDP, e outra camada usada para lidar com interações de pedido/resposta, usada para determinar os métodos e os códigos de resposta. Essas duas camadas virtuais são integradas no cabeçalho das mensagens CoAP (ver Tabela 3.3). Diferentemente do HTTP, o CoAP lida com as transações assincronamente por meio de transporte orientado a datagrama, utilizando o UDP. Isso é feito logicamente usando a camada de mensagens, que suporta confiabilidade opcional

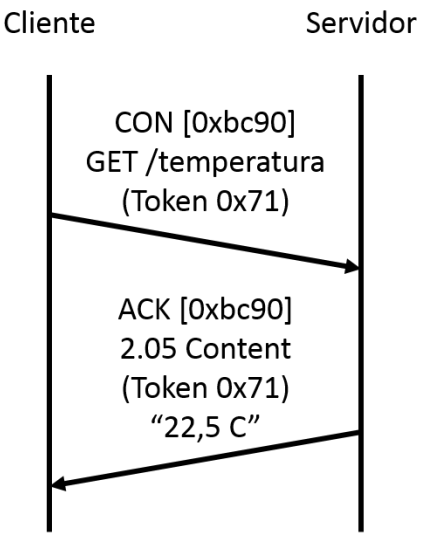


Figura 3.2: Troca de mensagens entre cliente/servidor.

(com *backoff* exponencial). Já retransmissões e reordenamento são implementados na própria camada de aplicação, excluindo a necessidade do TCP.

Tabela 3.2: Camadas lógicas do CoAP.

Protocolo	**Camada lógica**	**Descrição**
CoAP	Pedido/Resposta	interações de pedido/resposta
	Mensagem	interações assíncronas sobre UDP

Tabela 3.3: Formato da Mensagem CoAP.

Parte	**Campo**	**Tamanho**
Cabeçalho	Versão do CoAP	2 bits
	Tipo de mensagem	2 bits
	Tamanho do token	4 bits
	Código Pedido/Resposta	8 bits
	ID da mensagem	16 bits
Corpo	Token	0 a 8 bytes
	Opções	bytes restantes
	Conteúdo	

O CoAP define quatro tipos de mensagens (Tabela 3.4): CON (*Confirmable*), NON (*Non-confirmable*), ACK (*Acknowledgement*) e RST (*Reset*). Os códigos dos métodos e das respostas incluídos nas mensagens caracterizam o tipo de mensagem a ser utilizada nos pedidos e respostas. Os pedidos podem ser feitos com mensagens CON (com confirmação) e NON (sem confirmação) e as respostas são enviadas como mensagens ACK, portando o conteúdo solicitado. A mensagem RST é enviada quando o servidor não consegue processar uma mensagem.

Cada mensagem contém um identificador usado para detectar duplicatas e para

Tabela 3.4: Tipos de Mensagens CoAP.

Mensagem	Descrição
CON	Pedido com confirmação
NON	Pedido sem confirmação
ACK	Confirmação de recebimento
RST	Aviso de mensagem não processada

prover confiabilidade. A confiabilidade é alcançada pelo envio de mensagens CON. Uma mensagem CON é retransmitida usando um tempo de estouro padrão e *backoff* exponencial entre retransmissões até que o receptor envie uma mensagem ACK com o mesmo identificador da mensagem CON original. Quando o receptor não consegue processar a mensagem CON, este responde com uma mensagem RST no lugar do ACK. Uma mensagem que não exige transmissão confiável pode ser enviada por meio de uma mensagem NON. Embora não haja uma resposta com ACK para a mensagem NON, ela possui identificação para a detecção de duplicata. Quando o servidor recebe um pedido, mas não pode responder imediatamente, ele envia um ACK sem conteúdo para que o cliente não continue retransmitindo. Tão logo a resposta com o conteúdo esteja pronta, o servidor envia uma nova mensagem CON, incluindo o conteúdo solicitado, e o cliente confirma o recebimento com um ACK. Esse tipo de transação é chamado de resposta separada. O CoAP utiliza um subconjunto dos métodos do HTPP (GET, PUT, POST e DELETE), os quais funcionam de maneira similar, simplificando a tradução de um protocolo para o outro. Esses métodos são usados para definir os tipos de interações de pedido/resposta entre cliente e servidor (Tabela 3.5).

Tabela 3.5: Tipos de Interações CoAP.

Interação	Descrição
GET	Consulta recurso
POST	Cria recurso
PUT	Atualiza recurso
DELETE	Exclui recurso

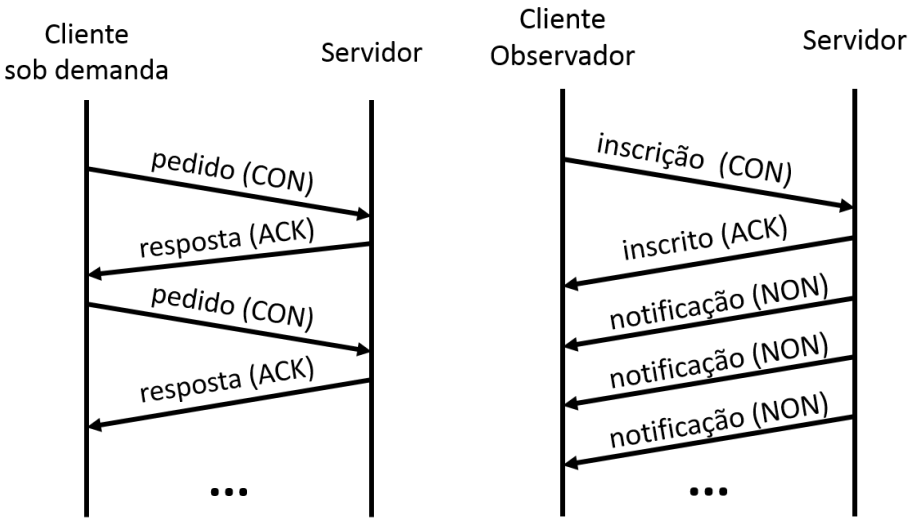
A cada interação, o servidor responde marcando a mensagem com o respectivo código de resposta (p. ex., 2.05), indicando se foi possível atender à solicitação e incluindo o conteúdo em caso positivo. O método GET solicita a informação atual correspondente a um recurso identificado pelo URI. O método POST solicita que o conteúdo da mensagem enviada seja processado pelo servidor, geralmente resultando na criação de um novo recurso ou na atualização de recurso existente. O método PUT solicita que o recurso identificado pelo URI seja atualizado ou criado de acordo com a descrição contida na mensagem. Finalmente, o método DELETE solicita que um recurso identificado pelo URI seja excluído. Nas interações de pedido/resposta é

possível incluir uma lista de uma ou mais opções. Por exemplo, várias informações relacionadas ao URI em um pedido podem ser transportadas no campo de opções da mensagem (ver Tabela 3.3). O CoAP define um conjunto de opções que podem ser usadas tanto no pedido quanto na resposta, tais como: formato do conteúdo, período máximo de validade de uma representação, tipo de provimento de serviço, endereço do *proxy*, dentre outras. Esse campo de opções pode ser usado também como um espaço para fornecer informações adicionais ao receptor da mensagem.

Uma aplicação de IoT frequentemente envolve um dispositivo exercendo a função de servidor e transmitindo informação sobre os seus recursos a outros dispositivos clientes. O CoAP suporta uma funcionalidade chamada de Observação [40], onde o cliente pode registrar-se como observador de um recurso do servidor através de um GET modificado. O servidor estabelece um relacionamento de observação entre o cliente e o recurso, onde cada recurso tem a sua própria lista de observadores e cada cliente só pode se registrar uma única vez na mesma lista. Uma vez registrado como observador de um recurso do servidor, o cliente passa a receber notificações do servidor sem a necessidade de gerar pedidos adicionais, como no caso dos clientes sob demanda. Contudo, é importante ressaltar que o número máximo de observadores que um servidor atende depende do espaço disponível em memória para armazenar as listas de cada recurso. As mensagens de registro e notificação são identificadas pela presença da opção Observação no campo de opções. No caso das notificações, a opção Observação inclui um número sequencial para possibilitar o ordenamento. Um cliente permanece na lista de observadores enquanto mantiver seu interesse no recurso. Normalmente, o servidor envia as notificações em mensagens NON, porém ele pode enviar uma notificação em uma mensagem CON para verificar a continuidade do interesse do cliente. A função Observação do CoAP evita a necessidade de o cliente demandar constantemente o servidor ou ter que manter uma sessão aberta como no caso do HTTP sobre TCP. Essa é uma das características que torna o CoAP indicado para uso em redes LLN. Essa extensão do CoAP permite que os clientes observadores acompanhem as mudanças nos recursos de seu interesse através das notificações do servidor, reduzindo a sobrecarga, o uso de banda e a latência da rede [41]. O servidor pode enviar notificações de três formas: 1) periodicamente, conforme o intervalo escolhido para o recurso; 2) sempre que houver alteração de status no recurso ou 3) sempre que determinada condição, previamente definida, for atingida. O recurso só pode ser disponibilizado para os clientes após a definição prévia dos seguintes atributos: URI do recurso, métodos aplicáveis ao recurso, tipo de provimento do serviço (sob demanda, observação) e período e/ou condição de notificação.

Incorporando ao CoAP/Observação as funcionalidades de *cache* e *proxy*, é possível aumentar a escalabilidade do sistema através do registro de observadores

em dispositivos intermediários, reduzindo, assim, o uso dos recursos do servidor. A função de Observação do CoAP favorece a escalabilidade e o consumo eficiente de energia, garantindo um maior tempo de disponibilidade do servidor. A Figura 3.3 ilustra a comunicação com o servidor quando o cliente interage sob demanda e quando o cliente se registra como observador. Note que nas interações sob demanda (Figura 3.3(a)), os clientes fazem um novo pedido sempre que desejam saber o status atual do recurso de interesse, recebendo a respectiva resposta do servidor. Já nas interações com observação (Figura 3.3(b)), os clientes observadores geram apenas um pedido de inscrição. Caso sejam inscritos na lista de observadores do recurso, esses clientes recebem as notificações sobre o recurso, sem gerarem novas demandas.



(a) Interação sob demanda. (b) Interação com observação.

Figura 3.3: Interação cliente/servidor.

Os aspectos fundamentais sobre a Internet das Coisas e o protocolo CoAP foram aqui apresentados, de modo a fornecer os elementos necessários para entender as funcionalidades do mecanismo proposto, expostas no próximo capítulo.

Capítulo 4

Mecanismo Proposto

Neste capítulo, o mecanismo de provisionamento de serviços com controle de demanda é apresentado. O mecanismo completo é formado por dois componentes principais:

1. mecanismo de controle de demanda;
2. mecanismo de comutação de servidores.

O primeiro componente é executado no servidor e faz o controle de demanda e a seleção de observadores. Esse componente baseia-se no ciclo de trabalho do rádio do servidor e no seu consumo de energia para definir o modo de operação no provimento dos serviços. O segundo componente é executado no cliente e faz a seleção e comutação de servidores CoAP, visando garantir a obtenção dos serviços necessários a uma aplicação e, adicionalmente, aumentar a disponibilidade dos serviços. Esse mecanismo utiliza recursos do próprio CoAP para realizar a comutação dos servidores, seguindo uma lista ordenada de endereços obtida a partir de uma infraestrutura central.

Um exemplo de aplicação do mecanismo no cenário industrial é o sistema de monitoramento e controle de escoamento de petróleo e derivados (Figura 4.1). Esse sistema faz o monitoramento, por exemplo, da quantidade de produto (recurso) em um tanque de petróleo. O tanque, por sua vez, armazena o produto recebido dos campos de extração e pode alimentar o sistema de refino ou o sistema de distribuição, fazendo a transferência dos produtos através do sistema de bombeio. Dessa forma, o processo de escoamento como um todo pode contar com vários sistemas interessados no mesmo recurso (nível do tanque), seja para controle operacional, fiscalização ou mesmo gestão de ativos. Cada um desses sistemas é composto por dispositivos (clientes) que dependem da informação do recurso para desempenharem suas funções corretamente. A prioridade no monitoramento deve ser dada aos sistemas críticos que estejam participando da operação de escoamento do produto,

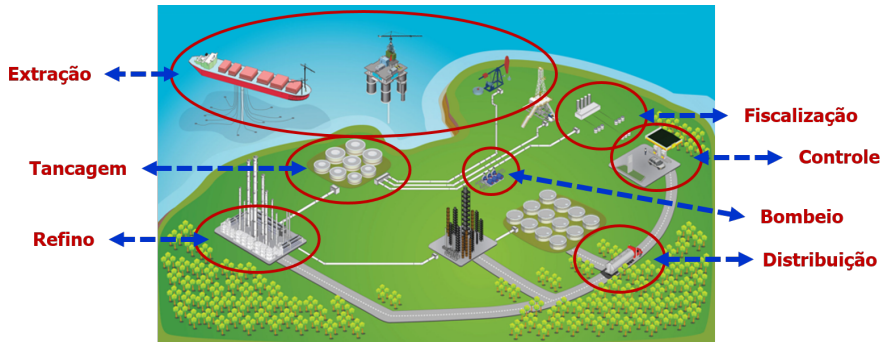


Figura 4.1: Um exemplo de aplicação na indústria de óleo e gás [1].

ou seja, esses sistemas devem ser tratados como clientes prioritários. Por exemplo, se a operação desejada é de transferência de produto do tanque para o sistema de refino, os dispositivos dos outros sistemas são considerados clientes não prioritários, enquanto que os dispositivos do sistema de refino devem ter garantida a disponibilidade da informação sobre o nível do tanque. Enquanto o servidor (instrumento de medição) estiver com carga normal na bateria, ele pode atender a consultas de outros sistemas (clientes sob demanda) dentro de um limite de consultas por unidade de tempo (modo de controle de demanda). Porém, caso a bateria esteja com carga baixa, o servidor só atende aos sistemas críticos (modo de economia de energia), de modo a permitir a execução plena da operação. Em geral, os tanques de armazenamento contam com instrumentos de medição redundantes a fim de conferir maior confiabilidade na medição do nível do tanque.

4.1 Mecanismo de Controle de Demanda

Esta seção apresenta o mecanismo de controle de demanda e seleção de observadores proposto, que tem por objetivo garantir a disponibilidade de servidores CoAP para clientes prioritários em um cenário de IoT. Esse mecanismo é executado nos servidores CoAP e é um dos componentes do mecanismo proposto de provisionamento de serviços de IoT.

Considerando que os pedidos CoAP gerados pelos clientes sob demanda tenham uma periodicidade t_n e desconsiderando os atrasos e as retransmissões, o número de pedidos gerados (demanda D_n) por um cliente pode ser estimado por:

$$D_n = \frac{t_{op}}{t_n}, \tag{4.1}$$

onde t_{op} é o tempo total de operação do dispositivo. Já para um cliente registrado na lista de observadores de um recurso do servidor, só é gerado um pacote de pedido CoAP, pois, como observador, esse cliente passa a receber pacotes CoAP de notificação sobre o recurso em observação. Para realizar a estimativa de demanda,

considerou-se uma aplicação de monitoramento operacional, onde o servidor envia periodicamente as notificações para os observadores, com periodicidade t_p. Sendo assim, o número de pacotes CoAP corresponderão somente ao pedido de inscrição do cliente, à resposta de registro e ao número de notificações (D_p), que pode ser estimado por:

$$D_p = \frac{t_{op}}{t_p} \tag{4.2}$$

Se, por exemplo, $t_{op} = 10\ min$, $t_n = 5\ s$ e $t_p = 10\ s$, então o número de pedidos CoAP gerados por um cliente sob demanda é $D_n = \frac{10\times60}{5} = 120$ e o número de notificações enviadas pelo servidor é $D_p = \frac{10\times60}{10} = 60$. Sendo assim, o número total de pacotes CoAP na rede em 10 minutos é 240 devido ao cliente sob demanda (o dobro porque considera o pedido e a resposta) e 62 devido ao cliente observador (incluindo o pedido de inscrição e a resposta do servidor). Neste caso, a redução de pacotes CoAP promovida pela função Observação é de 178 pacotes, ou seja, cerca de 75%. Essa redução pode ser ainda maior se as notificações forem enviadas somente quando uma dada condição for atendida. Quando $t_n = t_p = 10\ s$, obtém-se $D_n = D_p = \frac{10\times60}{10} = 60$. Neste caso, o cliente sob demanda é responsável por 120 pacotes CoAP na rede, enquanto o cliente observador responde por 62 pacotes, ou seja, uma redução de 58 pacotes. Finalmente, quando $t_n > t_p$, por exemplo $t_n = 20\ s$ e $t_p = 10\ s$, obtém-se $D_n = \frac{10\times60}{20} = 30$ e $D_p = \frac{10\times60}{10} = 60$, ou seja, $D_n < D_p$. Nessa condição, o cliente sob demanda e o cliente observador praticamente se equivalem em termos de quantidade de pacotes CoAP na rede, com cada um respondendo por 60 e 62 pacotes, respectivamente. Porém, quando se considera somente os pacotes enviados pelo servidor, o cliente sob demanda exige menos trabalho do rádio do servidor do que o cliente observador. Isso não necessariamente quer dizer que o consumo total de energia do servidor é maior com o cliente observador, pois o servidor também gasta energia para receber os pedidos dos clientes sob demanda. Isso mostra que a relação entre t_n e t_p pode determinar se há ganho ou não para a rede e para o servidor ter um cliente observador no lugar de um cliente sob demanda. Dessa forma, quando o cliente se torna um observador, a variação estimada no número total de pacotes é dada por:

$$\Delta D = 2 \times D_n - (D_p + 2) = t_{op}\left(\frac{2}{t_n} - \frac{1}{t_p}\right) - 2 \tag{4.3}$$

Note que ΔD pode ter valor negativo. Quando isso acontece, significa que o cliente observador promove maior fluxo de pacotes CoAP na rede do que o cliente sob demanda. Porém, mesmo no caso em que o cliente observador promove redução de carga na rede, é importante lembrar que o tamanho máximo da lista de observadores

depende do espaço na memória do servidor. Além disso, a demanda total (número de pacotes CoAP) depende do número de clientes observadores (p), do número de clientes sob demanda (n) e da relação entre t_n e t_p. Sendo assim, a demanda total na rede, pode ser dada por:

$$D_{total} = 2 \times n \times D_n + p \times (D_p + 2) = t_{op} \left(\frac{2 \times n}{t_n} + \frac{p}{t_p} \right) + (2 \times p) \tag{4.4}$$

Considerando somente os pacotes CoAP enviados pelo servidor, o que reflete diretamente no seu consumo de energia, temos uma demanda total dada por:

$$D_{servidor} = n \times D_n + p \times (D_p + 1) = t_{op} \left(\frac{n}{t_n} + \frac{p}{t_p} \right) + p \tag{4.5}$$

Pode-se, ainda, estimar a quantidade de pacotes que o servidor deve processar para atender aos pedidos dos clientes, o que está intimamente ligado ao cálculo da vazão de entrada do servidor. Considerando que todos os pedidos são processados com sucesso (sem perdas), a quantidade de pedidos enviados pelos clientes é igual a quantidade de processados pelo servidor. Dessa forma, enquanto o cliente sob demanda exige que o servidor processe D_n pedidos, o servidor só precisa processar um pedido de registro para o cliente que se torna observador. Considerando o número de clientes sob demanda (n) e o número de clientes observadores (p) presentes na rede, obtém-se que o número estimado de pedidos processados (N) pelo servidor durante a sua operação é dado por:

$$N_{servidor} = n \times D_n + p = \left(\frac{n \times t_{op}}{t_n} \right) + p \tag{4.6}$$

Note que, mantendo-se o número total de clientes ($n_c = n + p$) constante, se o número de observadores (p) e/ou o intervalo entre pedidos (t_n) aumentam, o número de pacotes processados pelo servidor diminui.

Assim, verifica-se que a funcionalidade de Observação do CoAP permite uma redução na demanda, porém não exerce controle sobre a demanda dos clientes não-observadores (sob demanda). O mecanismo de controle de demanda proposto promove a redução do consumo de energia do servidor, agindo sobre o ciclo de trabalho do rádio na transmissão de pacotes CoAP. O servidor monitora o ciclo de trabalho do rádio e, quando este ultrapassa um limiar Tx_{max}, o mecanismo faz o servidor entrar em modo de controle de demanda, somente usando o rádio para notificar os clientes observadores. Enquanto isso, os clientes sob demanda ficam com o serviço temporariamente suspenso. O motivo pelo qual o servidor atende somente aos clientes observadores é explicado mais adiante e tem a ver com a garantia de atendimento aos clientes prioritários. Esses clientes prioritários devem ser sempre registrados na

lista de observadores para que sejam atendidos quando o controle de demanda estiver ativo. No modo de controle de demanda, o mecanismo busca reduzir o ciclo de trabalho na transmissão, fazendo-o retornar a um valor inferior ao limiar definido. Isso é feito da seguinte forma: enquanto o intervalo de verificação, ou seja, o tempo de espera t_{espera} não é completado,

1. suspender o atendimento a clientes sob demanda;

2. quando houver uma notificação para cliente observador, enviar a notificação.

Quando o ciclo de trabalho do rádio volta ao nível desejado, o servidor volta a atender normalmente a todos os clientes. Caso a carga da bateria esteja abaixo do nível crítico $Q_{critico}$, o mecanismo faz o servidor entrar em modo de economia de energia. Nesse modo de operação, o rádio só é utilizado para atender a clientes observadores, suspendendo indefinidamente o atendimento a clientes sob demanda. Através desse mecanismo, é possível controlar o consumo de energia do servidor CoAP de forma a garantir a sua disponibilidade para o conjunto de clientes observadores, que deve incluir todos os clientes prioritários.

A segunda parte do mecanismo é a seleção de observadores dos recursos do servidor CoAP. Em qualquer cenário de IoT que contenha sistemas clientes críticos que devam ser priorizados, é importante garantir que esses sistemas consigam monitorar os recursos essenciais para o seu funcionamento. O mecanismo de seleção tem por objetivo selecionar quais clientes podem ser registrados na lista de observadores de um recurso do servidor CoAP. Para tanto, o servidor deve armazenar, para cada recurso, os endereços IP dos clientes prioritários. Sempre que houver um novo pedido de registro na lista de observadores, o endereço de origem do pedido é comparado com os endereços na lista de clientes prioritários do recurso. Caso esteja na lista, o cliente é registrado como observador; caso contrário, é verificado se há espaço na lista de observadores para registrá-lo. Caso um cliente prioritário deseje registrar-se como observador, mas a lista estiver cheia, o servidor deve excluir clientes não-prioritários da lista para incluir o cliente prioritário. Dessa forma, o mecanismo garante uma ordem de prioridade para monitoramento do recurso do servidor e, em conjunto com o mecanismo de controle de demanda, garante a disponibilidade para os clientes prioritários. Como já foi dito, o espaço em memória do dispositivo limita o tamanho máximo da lista de observadores de um recurso. Portanto, o dispositivo escolhido como servidor deve possuir capacidade de armazenamento suficiente para atender a todos os clientes prioritários ou o recurso deve ser disponibilizado por mais servidores. Isso deve fazer parte do projeto de automação do sistema, pois uma lista de observadores com tamanho menor que o número de clientes prioritários implica na rejeição de um ou mais clientes prioritários como observadores. Neste

caso, o mecanismo de controle de demanda pode comprometer o funcionamento dos sistemas clientes. A Figura 4.2 mostra o funcionamento do mecanismo de controle de demanda e a Figura 4.3 o mecanismo de seleção de observadores.

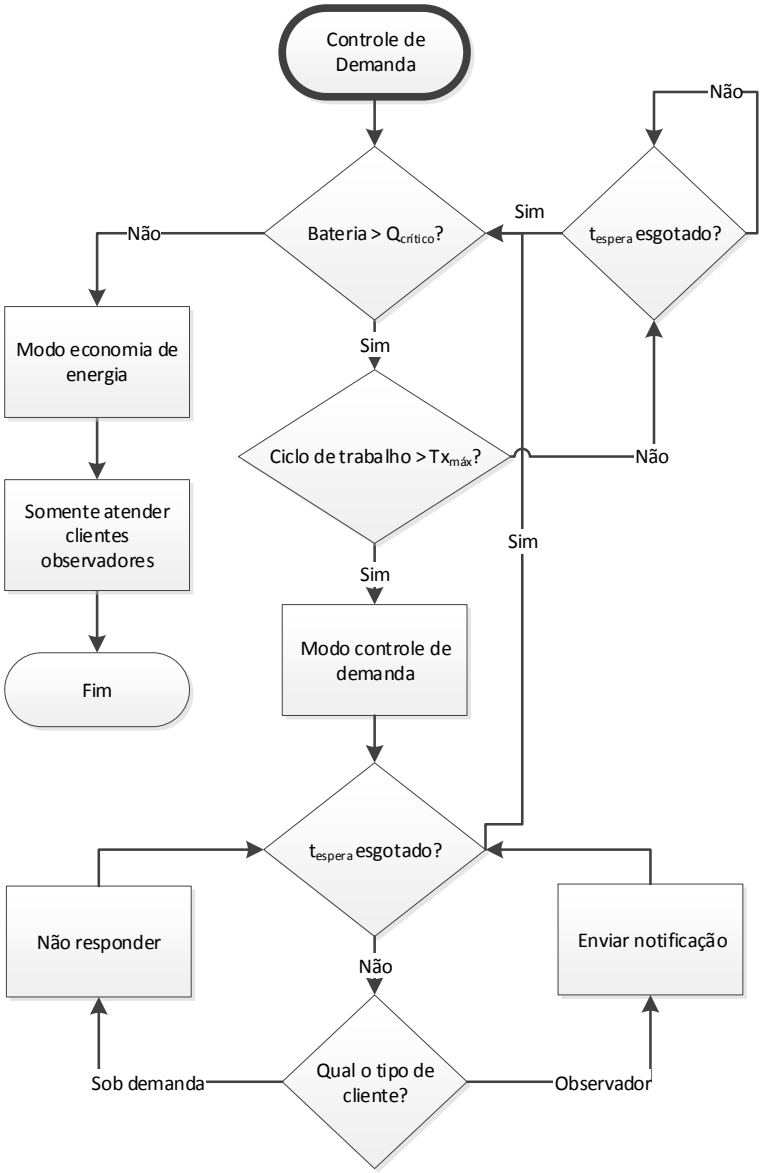


Figura 4.2: Algoritmo de controle de demanda.

Conforme mencionado anteriormente, no modo de controle de demanda o mecanismo busca levar a taxa de uso do rádio a um nível abaixo do limiar definido. Desse modo, quando a situação está controlada, o ciclo de trabalho (Tx_{normal}) apresenta um valor entre o mínimo requerido para atender aos clientes observadores (Tx_{min}) e o máximo admitido (Tx_{max}), ou seja,

$$Tx_{min} < Tx_{normal} < Tx_{max} \tag{4.7}$$

Já no modo de economia de energia, o ciclo de trabalho do rádio é igual ao mínimo

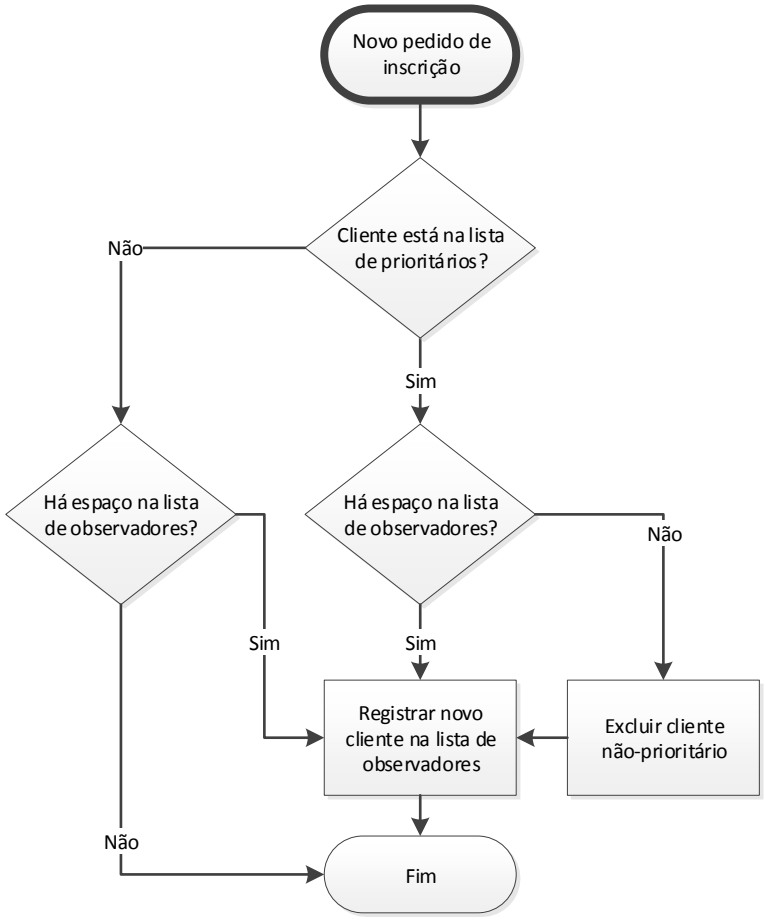


Figura 4.3: Algoritmo de seleção de observadores.

requerido para atender aos clientes observadores.

Para configurar o mecanismo de controle de demanda, é preciso definir o ciclo de trabalho máximo (Tx_{max}) admitido para o rádio do servidor. Esse valor pode ser estimado pela soma do ciclo de trabalho requerido para atender aos clientes observadores (Tx_{min}) com o ciclo de trabalho referente à demanda adicional admissível (Tx_D), ou seja,

$$Tx_{max} = Tx_{min} + Tx_D \tag{4.8}$$

O ciclo de trabalho representa a fração do período de tempo em que o rádio é efetivamente utilizado, conforme Equação 4.9:

$$Tx = \frac{t_{TX}}{t_{op}}, \tag{4.9}$$

sendo t_{TX} o tempo total de uso do rádio e t_{op} o tempo total de operação do servidor. Para definir Tx_{min}, considera-se o tempo total de operação desejado (t_{op}), o período de notificação necessário (t_p), o tempo de uso do rádio para cada notificação (t_{tx}) e o número de clientes observadores (p). O valor de t_{TX} é o produto do número de

notificações ($p \times D_p$) pelo tempo de uso do rádio para cada notificação (t_{tx}), ou seja,

$$t_{TX} = p \times \frac{t_{op}}{t_p} \times t_{tx} \tag{4.10}$$

Dessa forma, o valor mínimo do ciclo de trabalho é obtido pela Equação 4.11:

$$Tx_{min} = \frac{p \times t_{tx}}{t_p} \tag{4.11}$$

Note que quanto maior o número de observadores (p) e menor o intervalo entre notificações (t_p), maior é o ciclo de trabalho do rádio para transmitir as notificações. Para definir o ciclo de trabalho referente à demanda adicional admissível (Tx_D), um critério que pode ser adotado é considerá-lo um múltiplo do valor do ciclo de trabalho mínimo (Tx_{min}). Dessa forma, a partir da Equação 4.8, uma estimativa de limiar de ciclo de trabalho pode ser dada por:

$$Tx_{max} = Tx_{min} + d \times Tx_{min} = (d+1) \times \left(\frac{p \times t_{tx}}{t_p} \right), \tag{4.12}$$

onde d é um número inteiro que indica a admissibilidade do servidor à demanda adicional. Se $d = 0$, por exemplo, não há atendimento aos clientes sob demanda.

Para definir $Q_{critico}$, que é a energia mínima requerida para atender aos observadores durante o tempo de operação residual (t_{res}), considera-se o período entre notificações (t_p), a potência consumida pelo rádio para cada notificação (P_{tx}), o respectivo tempo de uso do rádio (t_{tx}) e o número de clientes observadores (p). Os valores de P_{tx} e t_{tx} podem ser obtidos experimentalmente ou consultando a folha de dados do dispositivo. A energia necessária para enviar uma notificação é $Q_{tx} = (P_{tx} \times t_{tx})$ e o número de notificações por observador no modo de economia de energia é $D_p = \frac{t_{res}}{t_p}$ (ver Equação 4.2). Assim, o produto $p \times D_p \times Q_{tx}$ fornece a energia total consumida pelo servidor durante o tempo de operação residual, ou seja,

$$Q_{critico} = \left(p \times \frac{t_{res}}{t_p} \times P_{tx} \times t_{tx} \right) \tag{4.13}$$

Quando o mecanismo de controle de demanda verificar que a carga da bateria está no nível crítico ($Q_{critico}$), então o modo de economia de energia será acionado.

Os valores de Tx_{max}, t_{espera} e $Q_{critico}$, indicados na Figura 4.2, são os parâmetros de configuração do mecanismo que definem o seu desempenho e o grau de economia de energia. Quanto menor o t_{espera}, maior a taxa de verificação do ciclo de trabalho e mais rápido é o retorno ao atendimento dos clientes sob demanda quando o ciclo de trabalho do rádio é normalizado. O nível crítico de carga da bateria ($Q_{critico}$) depende do tempo residual de operação desejado para o servidor. O servidor deve

continuar operando durante tempo suficiente para que as ações de manutenção sejam tomadas (p. ex., troca da bateria), sem comprometer o funcionamento do sistema cliente. Ressalta-se, ainda, que o mecanismo de controle de demanda monitora o ciclo de trabalho e o nível da bateria durante todo o tempo em que o servidor estiver funcionando. Quando o mecanismo de controle de demanda suspende o atendimento aos clientes sob demanda e estes clientes não recebem resposta, eles fazem a retransmissão do pedido até concluírem que o servidor não está mais disponível. Para contornar essa indisponibilidade do servidor, foi proposto o mecanismo de comutação de servidores apresentado a seguir.

4.2 Mecanismo de Comutação de Servidores

Esta seção apresenta o mecanismo de comutação de servidores CoAP proposto, que tem por objetivo aumentar a robustez na obtenção de serviços de IoT, promovendo maior disponibilidade da rede. Esse mecanismo é executado nos clientes CoAP como parte do mecanismo proposto de provisionamento de serviços de IoT.

Através do mecanismo proposto, o cliente realiza a comutação dos servidores a partir da consulta à sua lista de endereços de servidores que prestam o serviço desejado. Considera-se que essa lista de servidores é obtida a partir de uma infraestrutura central, aqui representada pelo *gateway* (roteador de borda) da rede de IoT. Essa infraestrutura, dentre outras funções, gerencia os serviços disponíveis na rede local e na Internet, atualizando e armazenando as informações sobre os servidores, tais como: endereços IP, recursos, status do dispositivo (carga da bateria, taxa de uso do rádio, vagas na lista de observadores), etc. A atualização e o gerenciamento dessas informações é possível utilizando o próprio CoAP, abordado no Capítulo 3, contudo o detalhamento de como o *gateway* deve executar essas funções não será feito nesta dissertação. O cliente pode solicitar uma lista de servidores que oferecem o mesmo tipo de serviço (p. ex., medição de temperatura) ou serviços de tipos diferentes (p. ex., medição de temperatura e umidade) para atender a uma determinada aplicação. A lista de servidores possui uma ordem de prioridade de consulta baseada nos requisitos da aplicação do cliente. Desse modo, ao solicitar a lista ao *gateway*, o cliente deve informar tais requisitos para que seja gerada uma lista organizada da forma desejada. Esses requisitos podem caracterizar os servidores a serem selecionados através de parâmetros como: número de vagas na lista de observadores, grau de confiabilidade da informação, localização do dispositivo, taxa de uso do rádio (demanda), etc. O *gateway* deve possuir um mecanismo de seleção de servidores que, considerando os critérios do cliente, selecione os melhores servidores para comporem a lista a ser fornecida para o cliente. Se o critério de seleção for, por exemplo, servidores com o serviço desejado que estejam mais próximos de

uma determinada localização, o mecanismo de seleção deve calcular o grau de proximidade dos servidores candidatos e posicioná-los na lista de servidores por ordem de proximidade. Caso os servidores possuam o recurso de localização por GPS, o grau de proximidade pode ser definido como a distância vetorial entre o servidor e o ponto de interesse.

O mecanismo de comutação de servidores testado neste trabalho partiu de uma lista de endereços de servidores já ordenada da forma desejada de acordo com a aplicação, considerando o processo de obtenção da lista já ocorrido. Seguindo a ordem da sua lista, o cliente realiza a consulta aos servidores. A cada consulta, caso o servidor responda, o cliente atualiza o status do servidor para disponível; caso contrário, o servidor é considerado indisponível. Uma das razões que pode tornar um servidor indisponível para um cliente sob demanda, por exemplo, é a ativação do mecanismo de controle de demanda no servidor. Se o cliente quiser se tornar um observador, ele deve fazer o pedido de inscrição. Caso a inscrição seja aceita, o serviço é fornecido em forma de notificações; caso contrário, o cliente ainda pode solicitar o serviço sob demanda. Assim, o modo de provimento de serviços depende dos critérios de seleção de observadores e do número de vagas disponíveis em cada servidor, conforme mostrado na Seção 4.1. O servidor pode dispor de um número de vagas para observadores que não sejam clientes prioritários, registrando esses clientes por ordem de chegada. Se a preferência do cliente for por servidores que aceitem novos observadores, ele muda para o próximo endereço da lista e tenta se inscrever no novo servidor. Ao final da lista, caso nenhum servidor tenha aceito a sua inscrição, o cliente poderá solicitar o serviço sob demanda dos servidores disponíveis, conforme a ordem de prioridade da lista. Se todos os servidores da lista estiverem indisponíveis, o cliente solicita ao *gateway* uma nova lista de servidores. Caso a lista fornecida pelo *gateway* seja igual a anterior, o cliente aguarda um tempo de espera e reinicia o ciclo de consulta. Se houver mudança na lista de servidores, o mecanismo reinicia o ciclo imediatamente.

Caso a lista seja composta por servidores que oferecem o mesmo tipo de serviço, o cliente pode finalizar as suas consultas assim que obtiver a resposta desejada ou continuar consultando os demais servidores da lista até completar o ciclo. Dessa forma, o cliente pode obter a informação de um único servidor ou combinar as informações dos vários servidores. Caso a lista seja constituída de servidores com recursos de tipos diferentes, o cliente deve consultar todos os servidores da lista, obtendo os serviços requeridos pela aplicação. O mecanismo de comutação de servidores é bastante útil para tirar proveito das redundâncias de recursos, permitindo a recomposição dos serviços em caso de indisponibilidade de algum servidor da lista. Ao flexibilizar a obtenção dos serviços pelo próprio cliente, o mecanismo confere maior robustez à rede no provimento de serviços de IoT. Destaca-se que as arquite-

turas propostas para IoT, em muitos casos, consideram que a disponibilização dos recursos dos servidores é feita por um servidor central ou um *broker* (ver Capítulo 2). Porém, o mecanismo de comutação de servidores, em conjunto com o CoAP, possibilita o acesso aos recursos pelo próprio cliente, uma vez que ele possui os endereços dos servidores e pode comunicar-se com eles diretamente.

Se os servidores da lista ativarem o mecanismo de controle de demanda apresentado na Seção 4.1, o mecanismo de comutação de servidores permite ao cliente comutar para um servidor que possa garantir o provimento do serviço (p. ex., servidores com mais energia ou com menor demanda), possibilitando assim um balanceamento de carga e de consumo de energia entre os servidores. Quando um servidor fica indisponível, o cliente leva um tempo para identificar a situação, podendo haver perda de pacotes (pedidos e notificações). O cliente que não recebeu resposta ao pedido CoAP realiza retransmissões até atingir o número previsto na configuração do CoAP. Se nenhum dos pedidos for atendido, o cliente aguarda o tempo de estouro do CoAP para considerar o servidor indisponível. Neste caso, o cliente efetuará a comutação para outro servidor. No caso do cliente observador de um recurso, o servidor passa a condição de indisponível quando o cliente para de receber as notificações esperadas. O critério utilizado no mecanismo foi um tempo de tolerância para o atraso nas notificações, acima do qual o cliente considera que o recurso ou o servidor está indisponível. Mesmo com a comutação para outro servidor, o mecanismo pode verificar periodicamente se o servidor preferido está disponível novamente ou ainda atualizar a lista de servidores, perguntando ao *gateway* se há servidores melhores para compor a lista. A Figura 4.4 mostra o funcionamento do mecanismo de comutação de servidores.

A parte do mecanismo relativa à obtenção da lista de servidores não foi implementada para este trabalho, contudo a proposta inicial prevê a utilização do próprio CoAP para realizar as transações entre cliente e *gateway*. Dessa forma, o cliente envia ao *gateway* um pedido solicitando uma lista de servidores, informando os requisitos para a seleção no campo de opções da mensagem CoAP. O *gateway*, por sua vez, age como um servidor CoAP que possui um recurso que aciona um seletor de servidores. Esse recurso é identificado pelo seu próprio URI que pode ser endereçado pelos clientes. O *gateway*, ao receber pedidos para esse recurso, inicia a elaboração da lista de servidores baseando-se nos requisitos do cliente presentes no campo de opções da mensagem recebida. O *gateway* inclui nessa lista os endereços IP dos servidores, contidos na sua base de dados, que melhor atendem ao pedido e encaminha a lista na mensagem de resposta do recurso CoAP. Por padrão, o *gateway* monta e atualiza a sua base de dados, conforme recebe as solicitações dos clientes, buscando os servidores que oferecem os serviços desejados. O *gateway* obtém a lista de recursos de cada servidor utilizando o mecanismo de descoberta de serviços do próprio CoAP,

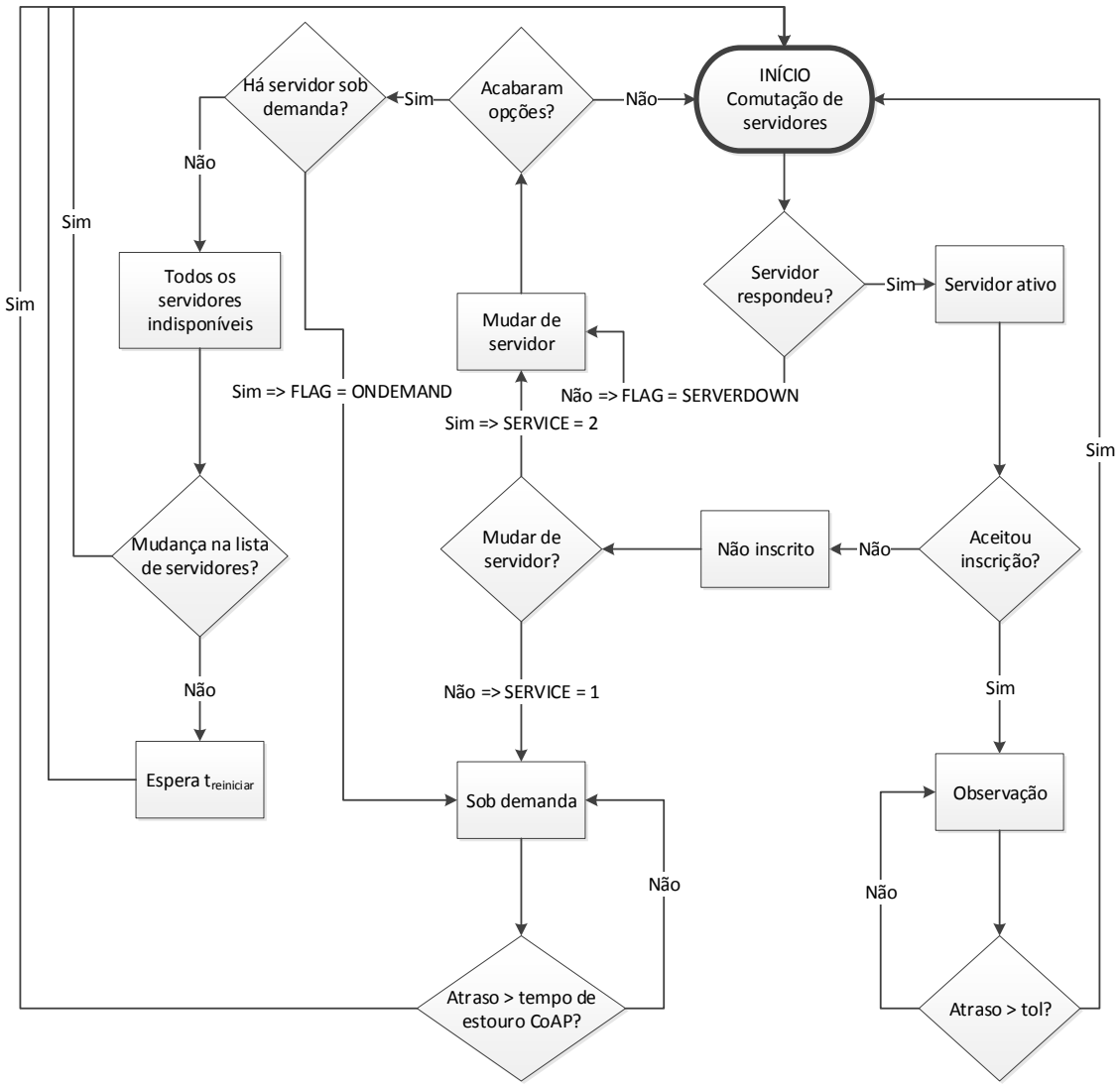


Figura 4.4: Algoritmo de comutação de servidores.

fazendo consultas sob demanda sempre que necessita dessa informação. Finalmente, o *gateway* envia a lista já com a ordem requerida, mas o cliente pode modificar essa ordem localmente a qualquer momento, conforme a sua conveniência. A Figura 4.5 mostra resumidamente a interação entre o cliente CoAP e o *gateway* para obtenção da lista de servidores CoAP requerida. Ressalta-se, novamente, que essa proposta de obtenção da lista de servidores não foi testada neste trabalho, tendo em vista que ainda não foi feita a sua implementação.

Para configurar o mecanismo de comutação de servidores, é preciso definir os seguintes parâmetros: tipo de atendimento preferido ($SERVICE$), tolerância de atraso para notificações (tol) e tempo de espera para reinício do ciclo de comutação ($t_{reiniciar}$). O parâmetro $SERVICE$ determina se o mecanismo prioriza a busca por uma vaga de observador ($SERVICE = 2$) ou se prioriza o serviço do primeiro servidor que responder ($SERVICE = 1$), mesmo que o serviço disponível seja ofe-

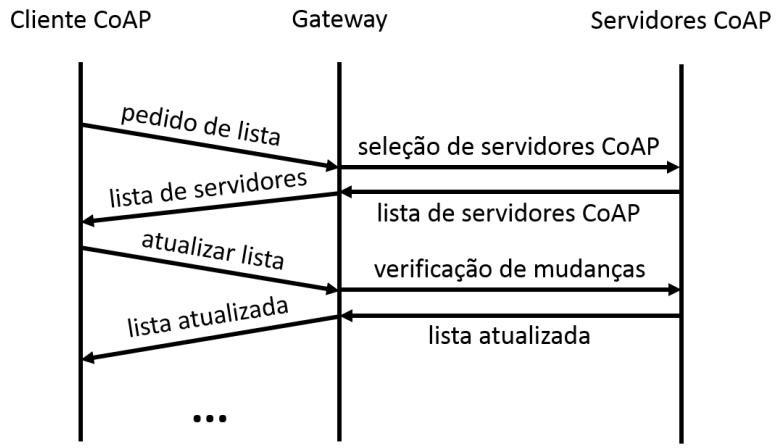


Figura 4.5: Obtenção da lista de servidores CoAP.

recido apenas sob demanda. O parâmetro *tol* especifica o número de notificações sucessivas não recebidas para que o cliente considere o servidor indisponível. Finalmente, o parâmetro $t_{reiniciar}$ estabelece o tempo de espera para reinício do mecanismo após todos os servidores serem considerados indisponíveis. Esses parâmetros devem ser definidos de acordo com a aplicação, considerando as restrições de tolerância a atraso dos serviços. Quanto maior o *tol*, maior será o tempo de espera para atualizar uma informação após sucessivas notificações não recebidas. Uma tolerância excessiva pode comprometer a confiabilidade da informação, pois ela pode estar muito defasada.

Esse mecanismo é aplicável a qualquer cenário de IoT, especialmente aquele composto por redes LLN. O mecanismo é útil, em particular, para aplicações que dependam de informações de diversos servidores, tirando proveito inclusive da redundância de recursos (p. ex., monitoramento ambiental). Dadas as limitações dos dispositivos de LLN, embora o cliente não consiga usar uma lista grande de servidores, o mecanismo proposto confere maior flexibilidade no provimento de serviços de IoT. Até o momento, não foram encontrados trabalhos que utilizem a comutação de servidores CoAP para aumentar a disponibilidade dos serviços de IoT. Os mecanismos de controle de demanda e de comutação de servidores descritos neste capítulo compõem conjuntamente o mecanismo de provisionamento de serviços proposto, que é avaliado no próximo capítulo.

Capítulo 5

Resultados e Discussões

Este capítulo apresenta as condições de análise do protocolo CoAP e do mecanismo proposto e os resultados dos experimentos realizados. Primeiramente, o ambiente de simulação é apresentado e, a seguir, o cenário considerado e as métricas utilizadas. Nas seções seguintes, a análise dos resultados é apresentada.

5.1 Ambiente de simulação

O simulador Cooja, desenvolvido para análise de redes LLN, foi utilizado nos experimentos. O Cooja pertence à plataforma de desenvolvimento do Contiki OS [20] e foi usado para aumentar o realismo das simulações. O Contiki OS é um sistema operacional de código aberto desenvolvido especialmente para dispositivos com recursos limitados, usados tipicamente em Redes de Sensores Sem Fio e em redes de IoT. O Contiki OS suporta inteiramente os protocolos IPv4 e IPv6, além dos protocolos criados recentemente para redes LLN, como 6LoWPAN [38, 39], RPL [23] e CoAP. O Contiki OS também possui mais de uma implementação da camada de Controle de Acesso ao Meio (MAC) para dispositivos de baixa potência (IEEE 802.15.4 [29]), sendo o ContikiMAC a principal. As aplicações do Contiki OS são escritas em C e a plataforma permite o carregamento do código e a compilação dinâmica em tempo de execução.

A plataforma de desenvolvimento do Contiki OS inclui o simulador Cooja, que permite simular o comportamento de aplicações de IoT em uma ampla gama de cenários. O simulador apresenta diversos relatórios a respeito do hardware (sinalização de rádio, alcance, potência, energia, luzes, botões, etc.) e das comunicações que ocorrem entre os dispositivos da rede (mensagens, protocolos, endereços, temporização, etc.). Esses relatórios podem ser visualizados em tempo de simulação e salvos para análise posterior. Os dispositivos usados na simulação são emulados a partir dos seus correspondentes comerciais, o que aumenta a proximidade do comportamento simulado com aquele que se espera encontrar em experimentos com

dispositivos reais.

5.2 Experimentos

Os experimentos foram realizados considerando uma rede de IoT LLN composta por dispositivos que assumem o papel de servidor ou cliente de acordo com a aplicação e a complexidade do sistema. O cenário utilizado considera a presença de clientes, servidores e roteador de borda (*gateway*). A Figura 5.1 ilustra a rede IoT considerada nos experimentos.

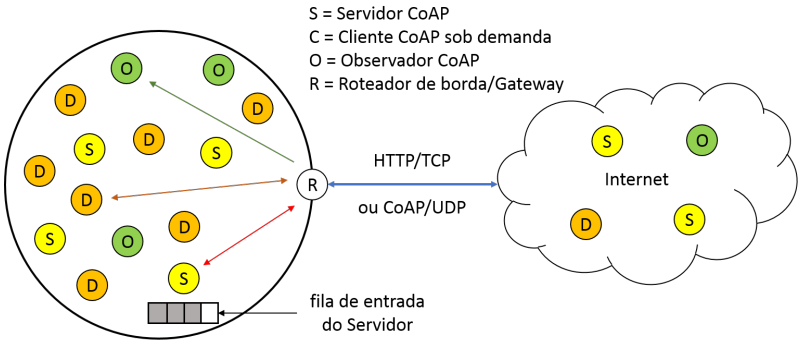


Figura 5.1: Cenário considerado nos experimentos.

Como cenário base considerou-se uma rede composta por um nó no papel de roteador de borda executando o protocolo RPL, um nó no papel de servidor CoAP e um número variável de nós como clientes CoAP. Posteriormente, o cenário foi expandido para considerar múltiplos servidores com recursos redundantes, permitindo aos clientes CoAP comutar entre os servidores para a obtenção dos serviços. A comunicação entre os clientes e o servidor ocorre sempre com a ajuda do roteador de borda, tornando indiferente ao servidor a origem do pedido, seja de cliente interno ou externo. Para cada configuração, a duração da simulação foi de 10 minutos. A maior aleatoriedade da simulação a cada iteração é promovida pela distribuição aleatória do posicionamento dos dispositivos na rede.

As simulações foram executadas em uma máquina virtual rodando o Contiki OS 3.0. A tecnologia utilizada em todos os dispositivos da rede foi o módulo TMote Sky (Mote IV), com tecnologia de rede IEEE 802.15.4, CPU de 16 bits, 10kB de RAM e 48 kB de memória Flash. Para reduzir o uso de memória de programa dos dispositivos, foi utilizada a seguinte configuração: NullRDC (mantém rádio sempre ligado para recepção), NullMAC (sem funcionalidades de MAC) e tamanho padrão de buffer uIP de 256 bytes. O tamanho da fila do servidor (buffer) deve ser definido previamente na configuração do dispositivo, de acordo com a sua capacidade de memória. Neste trabalho, o dispositivo foi configurado para permitir, no máximo, 4 transações CoAP na fila do servidor. O tamanho da lista de observadores pode ser no

máximo igual ao número máximo de transações CoAP permitidas simultaneamente. Nos experimentos, optou-se por deixar sempre no mínimo uma transação reservada para clientes sob demanda. Os recursos de cada servidor devem ser definidos antes da compilação do programa, incluindo o período de notificação para recursos periódicos.

Nos experimentos, os clientes CoAP podem ser clientes observadores (notificados periodicamente) ou clientes sob demanda. Para cada configuração, a periodicidade de notificações do servidor para os observadores foi mantida em $t_p = 10\ s$, intervalo razoável para a atualização da informação em aplicações de monitoramento. Ressalta-se que esse valor deve ser configurado previamente para cada recurso que oferece a opção de Observação, não podendo ser alterado após a sua disponibilização, a menos que seja feita uma nova compilação do código do servidor. Os parâmetros variáveis foram: número total de clientes (n_c) , período entre pedidos sob demanda (t_n), número de observadores (p) e tamanho da lista de servidores (n_s). Tomando-se o valor do período entre notificações ($t_p = 10\ s$) como referência, o valor do período entre pedidos sob demanda (t_n) foi variado para mostrar o comportamento do servidor quando t_n é menor, igual e maior que o valor de t_p. Sendo assim, os valores usados para t_n foram 5, 10 e 20 segundos. O número de clientes total foi variado entre $n_c = 1$ e $n_c = 10$. Embora o simulador permita a inclusão de mais dispositivos na rede, esses parâmetros foram considerados suficientes para realizar as observações desejadas, evitando-se forçar o dispositivo servidor a operar no limite de sua capacidade.

Tabela 5.1: Parâmetros de Configuração da Simulação.

Parâmetro	**Descrição**
t_p	Intervalo entre notificações
t_n	Intervalo entre pedidos sob demanda
p	Número de observadores
n_s	Tamanho da lista de servidores
n_c	Número total de clientes
Tx_{max}	Limiar do ciclo de trabalho
t_{espera}	Intervalo entre verificações
$SERVICE$	Tipo de atendimento preferido
tol	Tolerância de atraso para notificações
$t_{reiniciar}$	Tempo de espera para reinício do ciclo de comutações

As métricas de desempenho utilizadas na análise do servidor CoAP foram: quantidade de pacotes CoAP gerados, taxa de perdas, carga no servidor e consumo de energia (taxa de uso do rádio). Os intervalos de confiança mostrados nos gráficos são de 95%, representados por barras verticais. O número de perdas (L) é calculado somando-se o número de retransmissões de pacotes CoAP. Para obter a taxa de perdas (l), divide-se o número de perdas (L) pelo número total de pacotes CoAP

($N_{pacotes}$), incluindo as retransmissões, conforme Equação 5.1:

$$l = \frac{L}{N_{pacotes}} \tag{5.1}$$

A carga no servidor é a quantidade de dados processados pelo servidor dividido por um dado intervalo de tempo (Δt), que, neste trabalho, é o tempo decorrido entre o primeiro pacote CoAP gerado na rede e o último. Sendo assim, considerando o número de pedidos processados pelo servidor ($N_{servidor}$) dado pela Equação 4.6, convertido em bits pelo fator k (tamanho médio do pacote em bits), a carga no servidor (V) pode ser estimada por:

$$V = k \times \frac{N_{servidor}}{\Delta t} = k \times \frac{\left(\frac{n \times t_{op}}{t_n}\right) + p}{\Delta t} \tag{5.2}$$

Finalmente, a taxa de uso do rádio é calculada dividindo-se o tempo total de uso do rádio pelo tempo total de operação do dispositivo, conforme Equação 4.9, e multiplicando-se por 100 para obter o valor percentual.

Nas seções a seguir, os resultados estão organizados da seguinte forma. Primeiro, a influência dos parâmetros de configuração da rede no desempenho do servidor CoAP é discutida. Em seguida, os resultados dos experimentos utilizando o mecanismo proposto são apresentados e analisados. O servidor CoAP foi analisado com o fim de se entender o comportamento do servidor diante de diferentes demandas e configurações de rede. Com base nos resultados encontrados, o mecanismo proposto foi desenvolvido para aumentar a robustez no provimento de serviços, ao mesmo tempo que controla o consumo de energia. Na análise do mecanismo proposto, primeiramente discute-se o desempenho do controle de demanda no servidor e, posteriormente, discute-se o desempenho do mecanismo completo, com seus dois componentes em funcionamento.

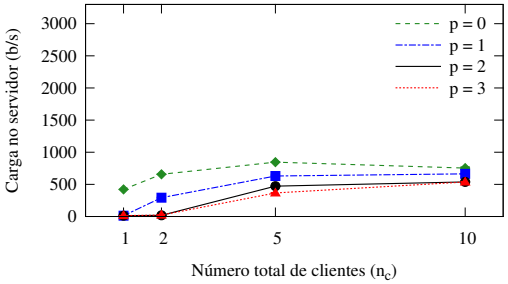
5.3 Desempenho do servidor CoAP

A seguir, o desempenho do servidor CoAP é discutido a partir da análise das métricas de desempenho definidas na seção anterior.

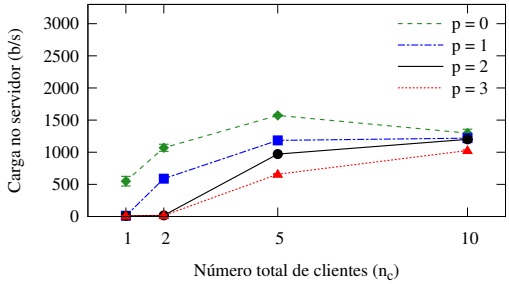
5.3.1 Carga no servidor

Avaliando a carga no servidor (Figura 5.2), observa-se que ela aumenta quando o número total de clientes ($n_c = n + p$) permanece constante e o número de clientes sob demanda cresce (menor p). Isso acontece porque a carga depende da quantidade de dados processados pelo servidor, conforme mostrado na Equação 5.2. Dessa forma, quanto mais clientes se tornam observadores (maior p), menos pedidos o

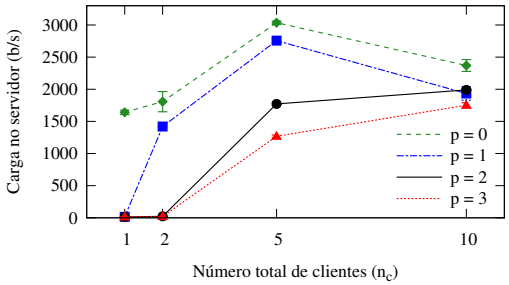
servidor recebe, reduzindo, assim, a quantidade de pedidos processados. Observa-se, também, que a carga cresce com a redução do intervalo entre pedidos sob demanda (t_n). A carga chega a atingir cerca de 3000 b/s para a configuração com 5 clientes sob demanda, nenhum cliente observador e período entre pedidos sob demanda igual a 5 segundos ($n_c = 5; p = 0; t_n = 5\ s$), conforme Figura 5.2(c). Essa taxa é compatível com aquela esperada para redes LLN mais limitadas, ou seja, da ordem de kb/s (Capítulo 3). Porém, possivelmente, ocorre um aumento no número de colisões no meio sem fio, à medida que o número de clientes não-observadores e a demanda aumentam, reduzindo, assim, a carga no servidor. A redução de carga pode ser observada quando o intervalo entre pedidos sob demanda é de 5 segundos ($t_n = 5\ s$), o número total de clientes é igual a 10 ($n_c = 10$) e o número de observadores é menor ou igual a 1 ($p \leq 1$). Conclui-se, portanto, que o fluxo de dados na entrada do servidor, para o mesmo n_c, é reduzido com o aumento de p. Além disso, o grau dessa redução é maior quando o intervalo entre pedidos é menor. Como já comentado, o cliente observador, uma vez inscrito no servidor, não gera mais requisições para o recurso em observação, reduzindo assim a carga sobre o servidor. Logo, o tamanho da lista de observadores, que está relacionado ao uso de memória, tem papel fundamental no desempenho do servidor com o aumento da demanda.



(a) $t_n = 20\ s$.



(b) $t_n = 10\ s$.

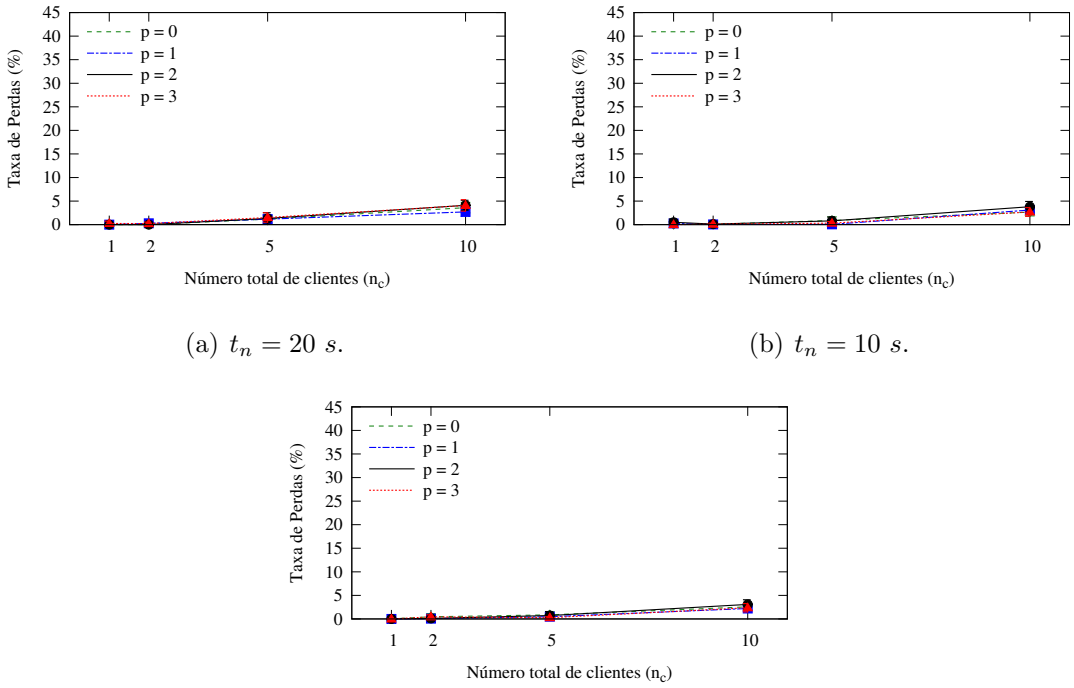


(c) $t_n = 5\ s$.

Figura 5.2: Carga no servidor.

5.3.2 Taxa de perdas

Na Figura 5.3, observa-se que a taxa de perdas se torna maior à medida que o número de dispositivos na rede aumenta, chegando a cerca de 3% quando há um total de 10 clientes na rede (n_c). Além disso, a variação do número de clientes observadores (p) e do intervalo entre pedidos sob demanda (t_n) pareceu não influenciar significativamente nos resultados. Assim, o aumento na taxa de perdas pode ser consequência do aumento na quantidade de colisões no meio sem fio, característica comum em redes LLN.



(a) $t_n = 20\ s$.

(b) $t_n = 10\ s$.

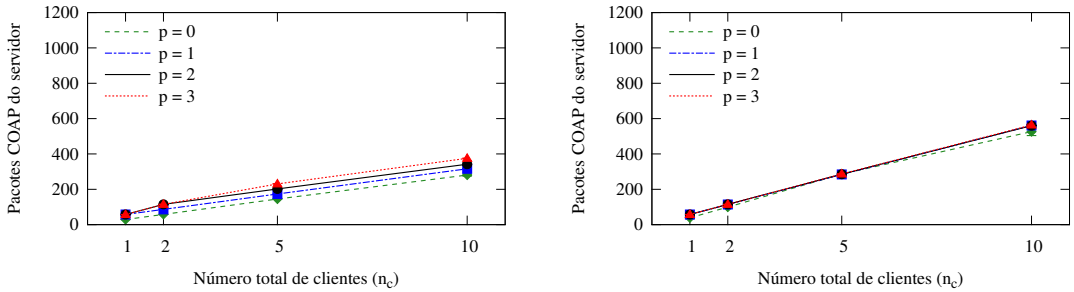
(c) $t_n = 5\ s$.

Figura 5.3: Taxa de perdas.

5.3.3 Número de pacotes CoAP

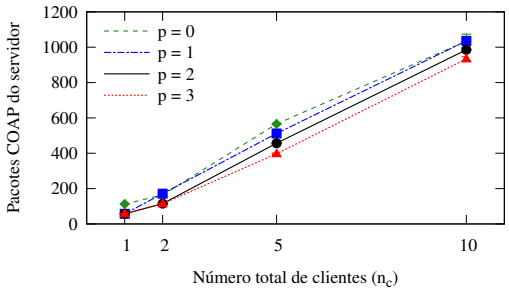
A Figura 5.4 mostra a evolução no número de pacotes CoAP gerados pelo servidor com o aumento da demanda. Observa-se que, quando o intervalo entre pedidos sob demanda é igual a 20 segundos ($t_n = 20\ s$), os clientes observadores demandam mais pacotes do servidor que os clientes sob demanda, como discutido na Seção 4.1, pois o intervalo entre notificações é menor ($t_p = 10\ s$). Por essa razão, nota-se na Figura 5.4(a) que o número de pacotes gerados pelo servidor é maior quando há mais clientes observadores. Por outro lado, quando o intervalo entre pedidos sob demanda é igual ao de notificações (Figura 5.4(b)), verifica-se que não diferença notável entre os tipos de clientes em questão de demanda do servidor. Como esperado, quando

o intervalo entre pedidos sob demanda é menor que o intervalo entre notificações (Figura 5.4(c)), o número de respostas a pedidos sob demanda é maior que o número de notificações por intervalo de tempo. Logo, os clientes sob demanda passam a demandar mais do servidor que os clientes observadores. Nessa situação, torna-se fundamental o uso do mecanismo de controle de demanda, a fim de manter o uso do rádio do servidor dentro dos limites especificados mesmo que a demanda continue crescendo.



(a) $t_n = 20\ s$.

(b) $t_n = 10\ s$.



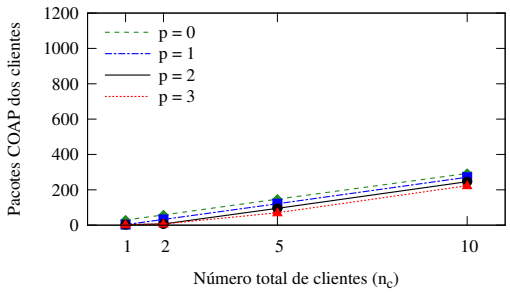
(c) $t_n = 5\ s$.

Figura 5.4: Pacotes CoAP transmitidos pelo servidor.

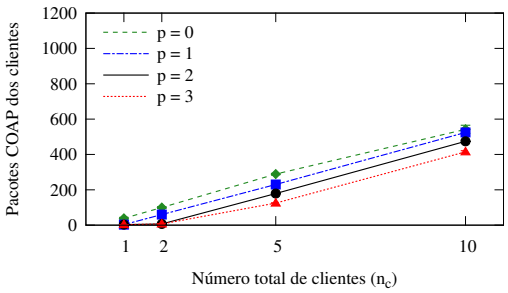
Do lado do cliente, a Figura 5.5 mostra que a influência do número de observadores (p) torna-se mais evidente quanto menor for o intervalo entre pedidos sob demanda (t_n) em relação ao intervalo entre notificações (t_p). Além disso, essa figura mostra que, quanto mais clientes se tornam observadores, menos energia eles consomem, devido à diminuição do número de pedidos gerados.

Consumo de energia

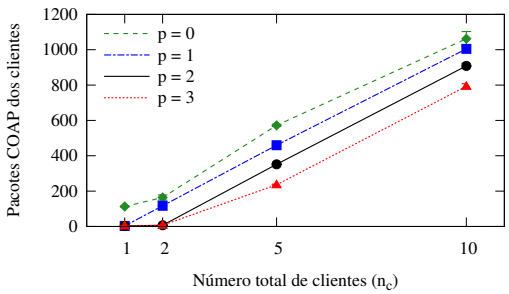
Os resultados apresentados na Figura 5.6 representam a percentagem de tempo em que o servidor utilizou o rádio para receber ou transmitir dados. A taxa de uso do rádio foi usada para avaliar o consumo de energia porque, além de serem diretamente proporcionais (vide folha de dados do dispositivo), a análise do ciclo de trabalho permite avaliar o quanto o sistema acompanha os parâmetros de configuração do mecanismo de controle de demanda. Observa-se que, para $t_n = 20\ s$



(a) $t_n = 20\ s$.



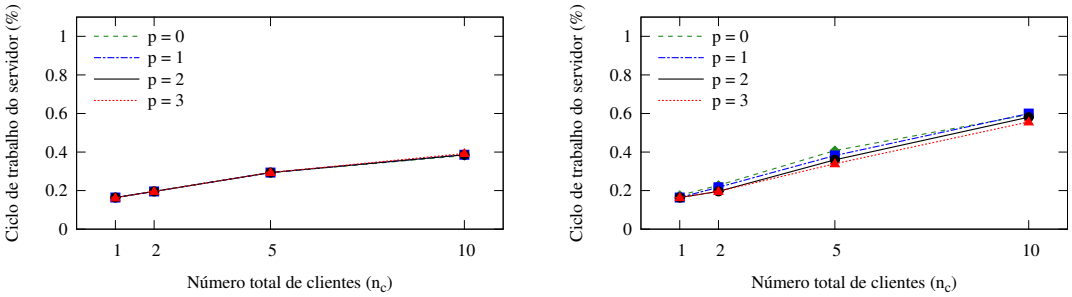
(b) $t_n = 10\ s$.



(c) $t_n = 5\ s$.

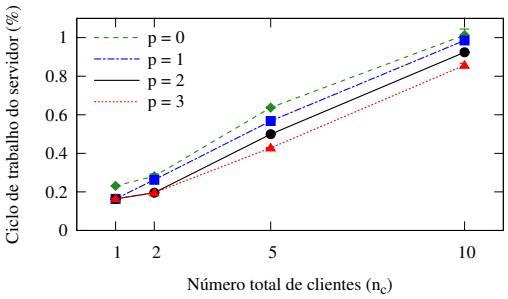
Figura 5.5: Pacotes CoAP transmitidos pelos clientes.

(Figura 5.6(a)), o consumo de energia é praticamente o mesmo quando apenas o número de observadores (p) é variado. Porém, o consumo cresce com o aumento do número total de clientes (n_c). Quando o intervalo entre pedidos é reduzido para 5 segundos (Figura 5.6(c)), os clientes sob demanda se tornam os maiores contribuintes para o uso do rádio do servidor. O mecanismo de controle de demanda visa justamente impedir que o servidor seja sobrecarregado pelo aumento da demanda dos clientes não-observadores, atuando na limitação do ciclo de trabalho do rádio.



(a) $t_n = 20\ s$.

(b) $t_n = 10\ s$.



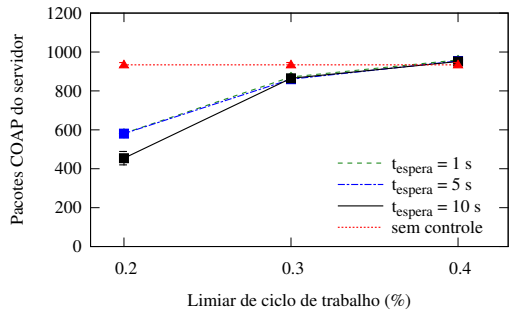
(c) $t_n = 5\ s$.

Figura 5.6: Ciclo de trabalho do rádio do servidor.

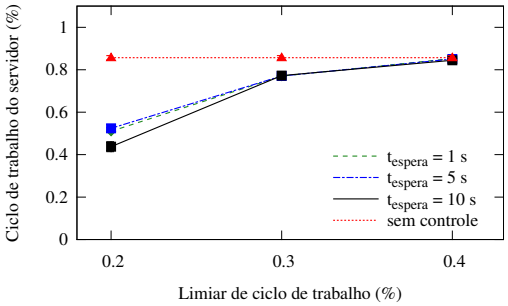
5.4 Desempenho do Mecanismo de Controle de Demanda

Conforme observado na análise de desempenho do servidor CoAP, o pior caso de demanda é aquele com menor intervalo entre pedidos e maior número de clientes sob demanda. Para fins de verificação da efetividade do mecanismo de controle de demanda, considerou-se um cenário com até três clientes prioritários ($p = 3$). Note que o mecanismo de controle de demanda depende da existência de clientes observadores para cumprir o seu objetivo. Sendo assim, o mecanismo de controle de demanda foi analisado para uma configuração contendo um total de 10 clientes, um intervalo entre pedidos sob demanda igual 5 segundos e número máximo de observadores igual a 3 ($n_c = 10; t_n = 5\ s; p = 3$). Os parâmetros de configuração do mecanismo de controle de demanda são o limiar do ciclo de trabalho Tx_{max} e o intervalo entre verificações da taxa de uso do rádio e do nível da bateria t_{espera}, conforme apresentado na Seção 4.1. Adotou-se como referência para o limiar de ciclo de trabalho o valor encontrado quando o número de clientes observadores (p) é igual ao número total de clientes (n_c). No caso ($n_c = 2; p = 3$) da Figura 5.6(c), o valor encontrado foi cerca de $0,2\%$. Nessa condição, o servidor só tem clientes observadores e, portanto, apresenta o mínimo ciclo de trabalho requerido. Dessa forma, o desempenho do mecanismo de controle foi comparado com essa condição

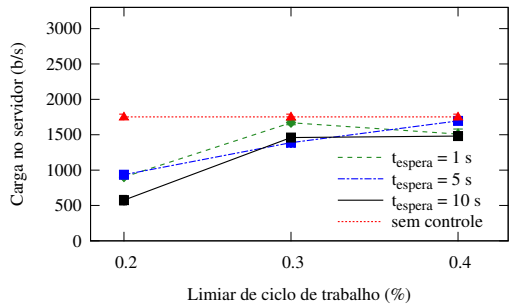
para verificar a sua capacidade de controlar a demanda ao rádio do servidor. A Figura 5.7 mostra que o mecanismo conseguiu controlar a taxa de uso do rádio, reduzindo a quantidade de pacotes gerados (Figura 5.7(a)) e economizando a energia do servidor (Figura 5.7(b)). A redução de uso do rádio do servidor foi de cerca de 50% para a configuração com limiar de ciclo de trabalho igual a 0,2% e tempo de espera igual a 10 segundos ($Tx_{max} = 0,2; t_{espera} = 10\ s$). Os melhores resultados foram obtidos com um intervalo entre verificações de 10 segundos ($t_{espera} = 10\ s$), conseguindo-se menor taxa de perdas, menor carga no servidor e menor consumo de energia, especialmente no caso mais restritivo ($Tx_{max} = 0,2$). Observa-se, ainda, que a influência da taxa de verificação diminui com o aumento do limiar de ciclo de trabalho. Isso acontece porque, quanto menos restritivo é o controle, menor é o número de ativações do modo de controle de demanda, o que prolonga o período de operação normal do servidor. O uso do mecanismo provoca redução de carga no servidor (Figura 5.7(c)) e, como esperado, maior taxa de perdas quanto mais restritivo for o Tx_{max} (Figura 5.7(d)). Isso mostra que o uso do mecanismo pode influenciar negativamente o provimento de serviços para os clientes sob demanda, pois, quando o mecanismo de controle de demanda é ativado, o servidor fica indisponível para novos pedidos e o serviço é interrompido para esses clientes. O mecanismo de comutação de servidores, analisado a seguir, visa compensar esse aspecto negativo, fornecendo um meio de aumentar a disponibilidade de servidores e garantir a manutenção dos serviços.



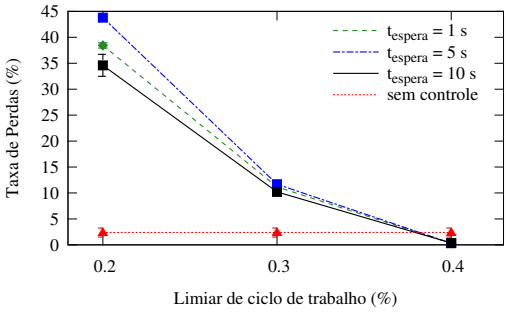
(a) Pacotes CoAP transmitidos pelo servidor.



(b) Ciclo de trabalho do rádio do servidor.



(c) Carga no servidor.



(d) Taxa de perdas.

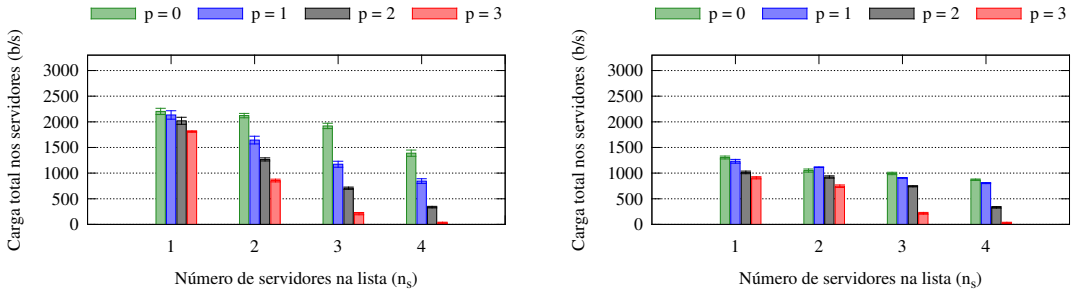
Figura 5.7: Desempenho do mecanismo de controle de demanda.

5.5 Desempenho do Mecanismo de Comutação de Servidores

Os testes com o mecanismo de comutação de servidores (sem controle de demanda) e com o mecanismo completo (com controle de demanda) foram realizados considerando o cenário com um total de 10 clientes e intervalo entre pedidos sob demanda de 5 segundos ($n_c = 10; t_n = 5\ s$). Essa configuração corresponde àquela de maior demanda utilizada na análise do CoAP, como visto na Seção 5.3. O tamanho da lista de servidores variou entre 1 e 4 ($1 \leq n_s \leq 4$). Para cada cenário, o tamanho da lista de observadores por servidor variou entre 0 e 3 ($0 \leq p \leq 3$). Os parâmetros de configuração para a comutação foram o tipo de atendimento preferido ($SERVICE$), a tolerância de atraso para notificações (tol) e o tempo de espera para reinício do ciclo de comutação ($t_{reiniciar}$), conforme apresentado na Seção 4.2. Os valores dos parâmetros de comutação utilizados nos experimentos foram $SERVICE = 2$ (preferência por servidores com vagas para observadores), $tol = 6$ (tolera esperar o equivalente a 3 períodos de notificação) e $t_{reiniciar} = 20\ s$ (espera 20 segundos para reiniciar mecanismo de comutação quando chega ao fim do processo). A configuração utilizada no mecanismo de controle de demanda, foi aquela que apresentou maior redução de uso do rádio nos casos testados (Seção 5.4), ou seja, a configuração ($Tx_{max} = 0,2; t_{espera} = 10\ s$).

5.5.1 Carga nos servidores

Avaliando a carga total no conjunto de servidores (Figura 5.8), observa-se que, quando somente um servidor presta o serviço, há uma forte redução na carga com a ativação do controle de demanda. O grau de redução da carga é menor quanto maior for o tamanho da lista de servidores (n_s) e o número de observadores (p). O aumento do número de observadores diminui o número de pedidos processados pelos servidores e, consequentemente, a carga. Dessa forma, quanto mais servidores recebem novos observadores, menor é a carga total sobre o conjunto de servidores e menor a frequência com que os servidores entram no modo de controle de demanda. Isso é possível graças a ação do mecanismo de comutação, que permite que o cliente obtenha o mesmo serviço de outro servidor. Esse resultado mostra a necessidade do planejamento da rede para que a demanda prevista seja atendida e o impacto no desempenho dos serviços sejam minimizados, mesmo diante do rearranjo da rede. Como observado, o aumento no número de observadores (p) reduz o fluxo de dados na entrada dos servidores, devido ao cliente observador não gerar requisições adicionais para o recurso em observação. Poder-se-ia concluir que uma boa prática é aumentar o tamanho da lista de observadores ao máximo possível, porém cabe lembrar que cada recurso de um servidor tem a sua lista de observadores e essas listas compartilham o mesmo espaço de memória, altamente restrito. Portanto, no projeto dos servidores CoAP deve-se levar em consideração quais os recursos a serem disponibilizados e o dimensionamento adequado de cada lista de observadores de acordo com a previsão de demanda.



(a) Sem controle de demanda.

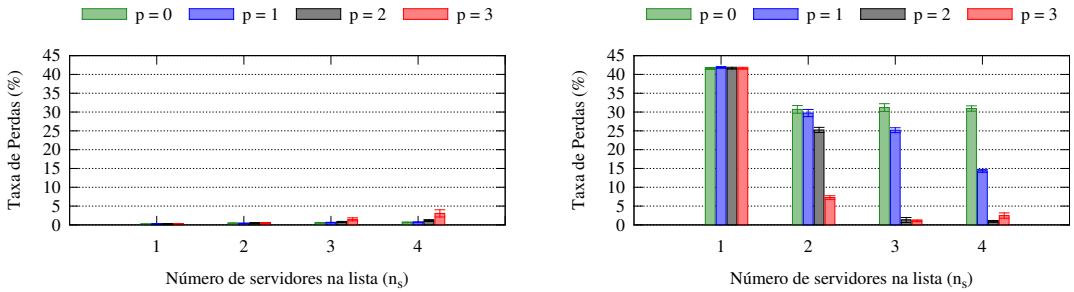
(b) Com controle de demanda.

Figura 5.8: Carga total nos servidores.

5.5.2 Taxa de perdas

A Figura 5.9(a) mostra que o mecanismo de comutação de servidores não impacta severamente na taxa de perdas da rede, mesmo quando há somente 1 servidor ($s = 1$) sendo consultado. Observa-se também que, quando o cliente pode consultar

até 4 servidores ($n_s = 4$) e estes oferecem 3 vagas para observadores ($p = 3$), há um ligeiro aumento na taxa de perdas. Isso acontece porque, nessa configuração, todos os clientes conseguem se tornar observadores (3 servidores com 3 observadores e 1 servidor com 1 observador), o que reduz drasticamente a quantidade de pedidos gerados. Essa redução faz com que o impacto de uma perda seja maior no valor final da taxa de perdas (ver Equação 5.1). Já na Figura 5.9(b), observa-se que a ativação do controle de demanda pode provocar a perda de mais de 40% dos pacotes quando há somente 1 servidor ($n_s = 1$). Entretanto, o aumento na quantidade de servidores redundantes e de clientes observadores permite compensar a indisponibilidade temporária de um servidor causada pelo mecanismo de controle de demanda.



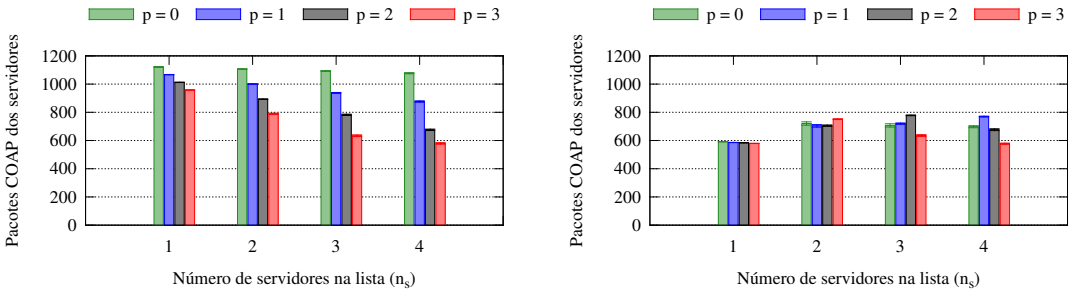
(a) Sem controle de demanda.

(b) Com controle de demanda.

Figura 5.9: Taxa de perdas.

5.5.3 Número de pacotes CoAP

O aumento do número de observadores (p) reduz o número de pedidos sob demanda na rede. No caso considerado, o intervalo entre pedidos sob demanda ($t_n = 5\ s$) é menor que o intervalo entre notificações ($t_p = 10\ s$). Por esse motivo, o aumento no número de observadores reduz também o número de pacotes enviados pelos servidores (Figura 5.10). Nota-se, também, que o aumento da lista de servidores (n_s) aumenta a oferta de vagas para observadores (cada servidor possui tamanho igual de lista de observadores), reduzindo ainda mais a quantidade de pedidos sob demanda. Quando o servidor possui o mecanismo de controle de demanda (Figura 5.10(b)), a frequência de ativação do controle de demanda é maior quanto maior for o número de clientes sob demanda ($n = n_c - p$). Por essa razão, o aumento do número de servidores na lista e do número de observadores reduz o impacto do uso do mecanismo de controle de demanda no atendimento aos clientes não prioritários.



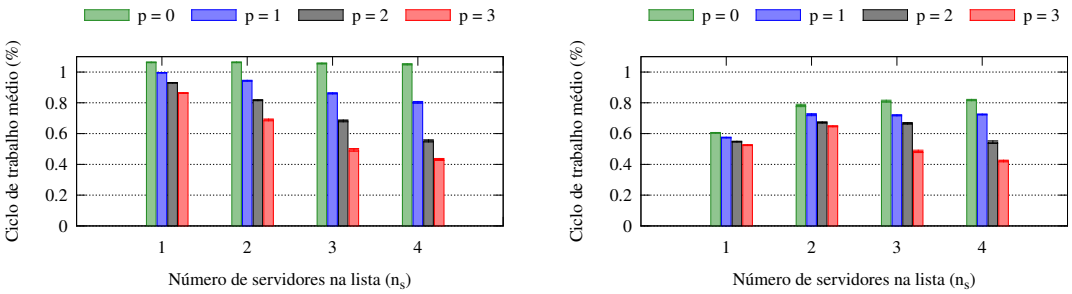
(a) Sem controle de demanda.

(b) Com controle de demanda.

Figura 5.10: Total de pacotes CoAP dos servidores.

5.5.4 Consumo de energia

Com relação ao consumo médio de energia no conjunto de servidores, a Figura 5.11 mostra um comportamento semelhante ao observado na Figura 5.10. Quando o controle de demanda é utilizado, quanto menos servidores, maior a frequência de ativação do controle para regular o consumo e, consequentemente, maior a redução da taxa de uso do rádio. À medida que os clientes comutam entre mais servidores e conseguem se tornar observadores, o ciclo de trabalho médio dos servidores passa a sofrer menos redução com a ativação do controle de demanda. Isso permite que os servidores tenham maior disponibilidade para atender a clientes não prioritários.



(a) Sem controle de demanda.

(b) Com controle de demanda.

Figura 5.11: Ciclo de trabalho médio do rádio.

5.5.5 Balanceamento de Carga e de Consumo de Energia

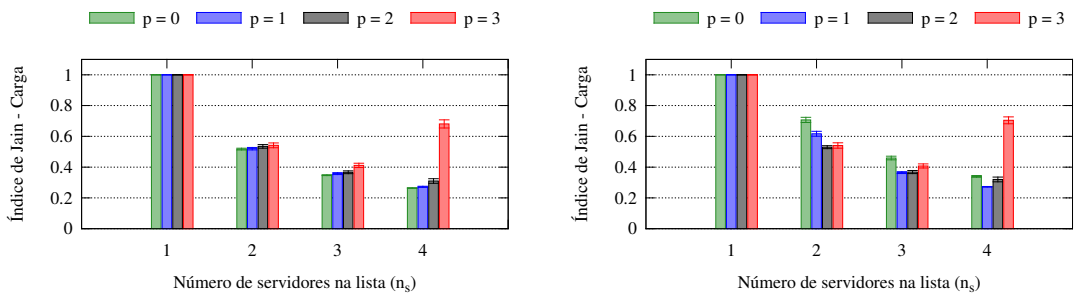
Neste tópico, os resultados mostram a distribuição de carga e de consumo de energia entre os servidores diante da aplicação do mecanismo proposto.

Como mencionado na Seção 4.2, a lista de servidores de cada cliente possui uma ordem de prioridade. Como a lista de servidores e a configuração utilizada no mecanismo de comutação de servidores foi a mesma para todos os clientes, o primeiro

servidor da lista foi o mais demandado. Através da comutação de servidores, é possível distribuir melhor a demanda e o consumo de energia entre os servidores. Por exemplo, se o cliente monitorar o nível de carga na bateria dos servidores, ele pode comutar para outro servidor depois de consumir certa quantidade de energia do servidor atual. Para avaliar o grau de distribuição de carga e de consumo de energia entre os servidores, ou seja, o índice de equidade, utilizou-se o índice de Jain, o qual é calculado pela Equação 5.3 e pode assumir valores entre 0 e 1. Quanto mais o índice de Jain se aproxima do valor 1, significa que melhor é o balanceamento entre os servidores, ou seja, melhor o índice de equidade. Quanto mais baixo o valor do índice de Jain, pior é o balanceamento.

$$f(x_1, x_2, ..., x_n) = \frac{(\sum_{i=1}^{N} x_i)^2}{n \sum_{i=1}^{N} x_i^2} \tag{5.3}$$

Carga nos servidores: A Figura 5.12 mostra como o valor do índice de Jain, calculado para a carga, varia com o tamanho da lista de servidores (n_s) e com o número de clientes observadores (p). Observa-se que, à medida que o tamanho da lista de servidores aumenta, o valor do índice de Jain decresce. Isso acontece porque, à medida que os clientes comutam para outros servidores e se tornam observadores, apenas parte do fluxo de dados é redistribuído. O primeiro servidor da lista absorve, num primeiro momento, todos os pedidos dos clientes sob demanda, exceto quando o servidor ativa o controle de demanda, forçando o cliente a comutar para outro servidor. Dessa forma, o aumento na quantidade de vagas para observadores ajuda a melhorar a distribuição de carga e, quando todos os clientes se tornam observadores ($n_s = 4; p = 3$), obtém-se um índice de equidade próximo de 0,7. Essa distribuição promove um balanceamento de carga entre os servidores, favorecendo a escalabilidade da rede. Note que a ativação do controle de demanda resulta num valor do índice de Jain ainda maior, pois o controle de demanda força a comutação de servidores, promovendo maior equidade em termos de carga.

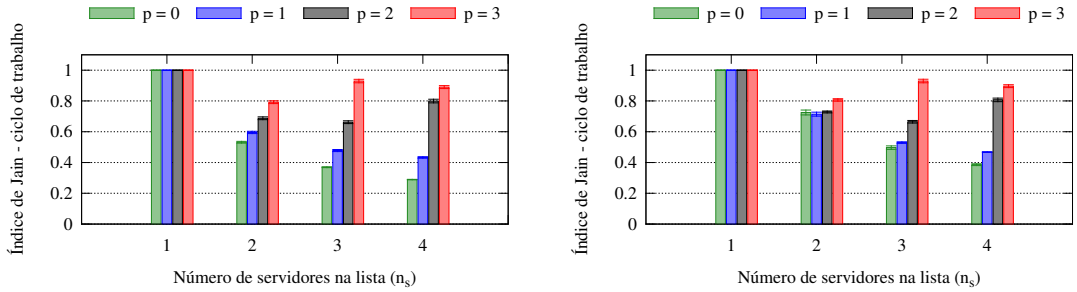


(a) Sem controle de demanda.

(b) Com controle de demanda.

Figura 5.12: Índice de Jain da carga nos servidores.

Consumo de energia nos servidores: Utilizando o índice de Jain para o consumo de energia, observa-se na Figura 5.13 que quanto mais clientes se tornam observadores, melhor é a distribuição do consumo de energia. Esse resultado evidencia que a comutação de servidores pode promover um balanceamento de consumo, obtendo-se um índice de equidade próximo de 0,9 quando há 3 ou mais servidores com 3 vagas para observadores cada um ($n_s \geq 3; p = 3$). O controle de demanda limita a taxa de uso do rádio do servidor e, apesar do servidor ficar indisponível para clientes não prioritários durante a ativação do controle, esses clientes podem garantir a obtenção do serviço comutando para outro servidor redundante.



(a) Sem controle de demanda.

(b) Com controle de demanda.

Figura 5.13: Índice de Jain do ciclo de trabalho dos servidores.

Os resultados mostram que o mecanismo proposto aumenta a confiabilidade no provimento de serviços de IoT e permite a distribuição de demandas e, consequentemente, o balanceamento do consumo de energia entre os servidores. Além disso, o controle de demanda regula o consumo de energia no servidor e garante a disponibilidade dos recursos para os clientes prioritários.

Capítulo 6

Conclusões e Trabalhos Futuros

A Internet das Coisas está sendo impulsionada pela evolução tecnológica dos dispositivos eletrônicos conectados à Internet. Essa tendência tem sido vista especialmente no âmbito industrial, no fomento da chamada quarta revolução industrial, onde toda a cadeia produtiva e logística funciona de forma autônoma. Para atender às novas demandas, torna-se fundamental desenvolver mecanismos e protocolos de comunicação que sejam compatíveis com as limitações dos dispositivos de IoT. A quantidade de dispositivos conectados à Internet cresce rapidamente, aumentando o fluxo de dados e desafiando o potencial de escalabilidade das redes, ao mesmo tempo que cria um cenário fértil em recursos para o provisionamento de serviços. Os mecanismos propostos devem garantir a disponibilidade dos dispositivos para o provimento confiável de serviços de IoT, promovendo o consumo eficiente dos recursos. Além disso, a análise de desempenho dos mecanismos deve ser feita em redes LLN, senão os resultados podem ser erroneamente otimistas, não considerando as restrições inerentes a essas redes.

Dentre os novos protocolos propostos para a Internet das Coisas, o CoAP destaca-se por ser desenvolvido por institutos de padronização reconhecidos internacionalmente, além de ser fortemente difundido na literatura. A escolha do CoAP para a aplicação do mecanismo proposto deve-se ao fato do CoAP, além de possuir a funcionalidade de Observação, ser adequado para uso em redes compostas por dispositivos com recursos limitados, condições fundamentais para uso no cenário considerado neste trabalho. Além disso, o CoAP guarda semelhança com o HTTP, facilitando assim a tradução de um protocolo para o outro, e oferece a possibilidade de comunicação por IP entre cliente e servidor via Internet. Essas vantagens tornam o CoAP um protocolo bastante promissor para uso em aplicações de IoT.

O mecanismo de provisionamento de serviços proposto é composto por dois componentes. O primeiro componente opera no servidor e controla a demanda dos clientes atuando sobre o ciclo de trabalho do rádio do servidor, reduzindo o consumo de energia e garantindo o provimento de serviços aos clientes prioritários. O se-

gundo componente opera no cliente e faz a comutação dos servidores que constam na lista de servidores do cliente para aumentar a robustez na obtenção dos serviços. A funcionalidade de observação do CoAP e a capacidade de comunicação fim-a-fim entre cliente e servidor são características fundamentais do protocolo para permitir o funcionamento eficiente do mecanismo proposto. Quanto mais clientes conseguirem se inscrever como observadores, maior é a redução de pedidos CoAP circulando na rede e menor será o impacto no provimento de serviços se o controle de demanda for ativado. No entanto, o espaço em memória é limitado e constitui uma restrição ao tamanho da lista de observadores permitida para os recursos de um servidor. Quanto maior a abundância de servidores CoAP, melhor é o desempenho do mecanismo de comutação de servidores, inclusive no balanceamento do consumo de energia. Quanto mais servidores disponíveis, maior é a garantia de continuidade dos serviços. O mecanismo é aplicável a qualquer cenário de IoT onde se queira controlar a demanda imposta a um servidor CoAP, sem comprometer a sua disponibilidade para os clientes prioritários. Além disso, o mecanismo também é útil para aproveitar a disponibilidade de servidores CoAP com o objetivo de aumentar a robustez no provisionamento de serviços.

O mecanismo foi avaliado utilizando o simulador Cooja e as métricas empregadas foram carga no servidor, taxa de perdas, quantidade de pacotes CoAP gerados e consumo de energia. O cenário considerado nos experimentos foi uma rede de IoT LLN composta por múltiplos clientes CoAP e múltiplos servidores CoAP redundantes, ou seja, oferecendo o mesmo serviço. Os resultados mostraram que o servidor CoAP sem o controle de demanda sofre degradação de desempenho à medida que a demanda aumenta, tendo sua energia consumida mais intensamente. O uso do mecanismo de controle de demanda em conjunto com a funcionalidade de Observação do CoAP aumenta a eficiência da rede, reduzindo o consumo de energia do servidor significativamente e garantindo a disponibilidade dos seus recursos para os clientes prioritários. O controle de demanda reduz a carga no servidor e o consumo de energia, conseguindo controlar a taxa de uso do rádio. Enquanto o mecanismo está em modo de controle de demanda, somente os clientes observadores (prioritários) são atendidos.

Com relação ao mecanismo de comutação de servidores, os resultados mostraram que quanto maior a lista de servidores oferecendo vagas para observadores, menor é a carga nos servidores, tendo em vista a distribuição do fluxo de dados entre os servidores. O aumento na taxa de perdas promovido pelo controle de demanda é reduzido à medida que a lista de servidores e a quantidade de observadores aumentam, mostrando que o mecanismo de comutação consegue compensar a indisponibilidade do servidor que está com o controle de demanda ativado. Essa conclusão pode ser obtida também analisando o consumo de energia dos servidores. Observa-se que

a funcionalidade de observação e a comutação de servidores promovem a redução, bem como o balanceamento do consumo energético, na medida em que tornam menos frequente a atuação do mecanismo de controle de demanda nos servidores da rede.

As principais contribuições deste trabalho são a proposta de um novo mecanismo de provisionamento de serviços com controle de demanda e a análise de desempenho e escalabilidade de servidores CoAP em redes LLN. A análise dos servidores CoAP permitiu verificar a influência dos parâmetros de configuração da rede no desempenho do CoAP. Já o mecanismo proposto melhora o desempenho e a escalabilidade dos servidores CoAP, mantendo sob controle o consumo de energia e garantindo a disponibilidade para clientes prioritários. Além disso, a comutação de servidores permite a obtenção de serviços de forma distribuída, aumentando a disponibilidade dos serviços. Ressalta-se que este trabalho apresenta uma proposta inédita para economia de energia em redes com servidores CoAP e a proposta de controle de demanda, juntamente com a análise de servidores CoAP, rendeu a publicação de um artigo em congresso nacional [21]. Ademais, este é o primeiro trabalho a propor um mecanismo que permita a obtenção de serviços no próprio cliente CoAP através da comutação de servidores.

Como trabalhos futuros, sugere-se implementar a proposta do mecanismo de obtenção da lista de servidores de acordo com os requisitos da aplicação do cliente. Esse é um passo necessário para completar o ciclo de comunicação para o provimento de serviços de IoT. Um passo posterior pode ser o desenvolvimento de um mecanismo de gerenciamento dos servidores cadastrados no *gateway* e seus respectivos serviços. Esse mecanismo pode partir de alguma arquitetura proposta em trabalhos desenvolvidos na área de descoberta de serviços. Uma outra possibilidade é a ampliação do mecanismo proposto ou a criação de um novo para interagir com *proxies* ou servidores auxiliares, visando o aumento na quantidade de observadores e a possibilidade de tratamento prévio da informação (p. ex., análise de confiabilidade e consolidação da informação). Finalmente, sugere-se também a realização de testes de laboratório em dispositivos reais para avaliar o desempenho da proposta e comparar com os resultados obtidos no simulador Cooja.

Referências Bibliográficas

[1] 3M. "3M Oil and Gas". 2017. Disponível em: <http://www.3m.com/intl/AE/OilGas/interactive_solutions_map/index.html>. [Online: 01-December-2017].

[2] GROUP, M. M. "Internet World Stats". 2016. Disponível em: <http://www.internetworldstats.com/stats.htm>. [Online: 18-November-2016].

[3] AL-FUQAHA, A., GUIZANI, M., MOHAMMADI, M., et al. "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications", *IEEE Communications Surveys Tutorials*, v. 17, n. 4, pp. 2347–2376, June 2015.

[4] PWC. "Industry 4.0: Building the Digital Enterprise". 2016. Disponível em: <http://www.pwc.com/gx/en/industries/industry-4.0.html>. [Online: 15-November-2016].

[5] XU, L. D., HE, W., LI, S. "Internet of Things in Industries: A Survey", *Industrial Informatics, IEEE Transactions on*, v. 10, n. 4, pp. 2233–2243, Nov 2014.

[6] ATZORI, L., IERA, A., MORABITO, G. "The Internet of Things: A survey", *Computer Networks*, v. 54, n. 15, pp. 2787 – 2805, Oct 2010.

[7] CISCO. "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast". 2017. Disponível em: <www.cisco.com>. [Online: 18-December-2017].

[8] GRANJAL, J., MONTEIRO, E., SA SILVA, J. "Security for the Internet of Things: A Survey of Existing Protocols and Open Research Issues", *Communications Surveys Tutorials, IEEE*, v. 17, n. 3, pp. 1294–1312, Jan 2015.

[9] BARBATO, A., BARRANO, M., CAPONE, A., et al. "Resource Oriented and Energy Efficient Routing Protocol for IPv6 Wireless Sensor Networks". In: *Online Conference on Green Communications (GreenCom), 2013 IEEE*, pp. 163–168, Piscataway, USA, Oct 2013.

[10] IOVA, O., THEOLEYRE, F., NOEL, T. "Improving the Network Lifetime with Energy-Balancing Routing: Application to RPL". In: *Wireless and Mobile Networking Conference (WMNC), 2014 7th IFIP*, pp. 1–8, Vilamoura, Portugal, May 2014.

[11] JAN, M., NANDA, P., HE, X., et al. "A Robust Authentication Scheme for Observing Resources in the Internet of Things Environment". In: *Trust, Security and Privacy in Computing and Communications (TrustCom), 2014 IEEE 13th International Conference on*, pp. 205–211, Beijing, China, Sept 2014.

[12] AFZAL, B., ALVI, S. A., SHAH, G. A., et al. "Energy efficient context aware traffic scheduling for IoT applications", *Ad Hoc Networks*, v. 62, pp. 101 – 115, July 2017. ISSN: 1570-8705.

[13] FIWARE. "Context Broker (Orion)". 2011. Disponível em: <`http://fiware-iot-stack.readthedocs.io/en/latest/context_broker/index.html`>. [Online: 20-January-2018].

[14] BANKS, A., GUPTA, R. "MQTT Version 3.1.1", *OASIS Standard*, pp. 1–81, Oct 2014.

[15] JEYARAMAN, R., TELFER, A. "AMQP Version 1.0", *OASIS Standard*, Oct 2012.

[16] SHELBY, Z., HARTKE, K., BORMANN, C. "The Constrained Application Protocol (CoAP)", *Internet Engineering Task Force (IETF)*, v. RFC 7252, June 2014.

[17] MUN, D. H., DINH, M. L., KWON, Y. W. "An Assessment of Internet of Things Protocols for Resource-Constrained Applications". In: *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, pp. 555–560, Atlanta, USA, June 2016.

[18] THANGAVEL, D., MA, X., VALERA, A., et al. "Performance evaluation of MQTT and CoAP via a Common Middleware". In: *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2014 IEEE Ninth International Conference on*, pp. 1–6, Singapore, Singapore, April 2014.

[19] TALAMINOS-BARROSO, A., ESTUDILLO-VALDERRAMA, M. A., ROA, L. M., et al. "A Machine-to-Machine protocol benchmark for eHealth applications - Use case: Respiratory rehabilitation", *Computer Methods and Programs in Biomedicine*, v. 129, pp. 1–11, June 2016.

[20] THINGSQUARE. "Contiki: The Open Source OS for the Internet of Things". 2016. Disponível em: <http://www.contiki-os.org>. [Online: 14-November-2016].

[21] BAHIA, J. G., CAMPISTA, M. E. M. "Um Mecanismo de Controle de Demanda no Provimento de Serviços de IoT Usando CoAP". In: *XXII Workshop de Gerência e Operação de Redes e Serviços (WGRS'2017)*, pp. 123–136, Belem, Brasil, May 2017.

[22] ALAM, S., DE, D., RAY, A. "Analysis of Energy Consumption for IARP, RIP and STAR Routing Protocols in Wireless Sensor Networks". In: *Advances in Computing and Communication Engineering (ICACCE), 2015 Second International Conference on*, pp. 11–16, Dehradun, India, May 2015.

[23] WINTER, T., THUBERT, P., BRANDT, A., et al. "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", *Internet Engineering Task Force (IETF)*, v. RFC 6550, March 2012.

[24] TWAYEJ, W., AL-RAWESHIDY, H. S. "An energy efficient M2M routing protocol for IoT based on 6LoWPAN with a smart sleep mode". In: *2017 Computing Conference*, pp. 1317–1322, London, UK, July 2017.

[25] FIWARE. "FIWARE IoT Stack". 2011. Disponível em: <http://fiware-iot-stack.readthedocs.io/en/latest/index.html>. [Online: 20-January-2018].

[26] OMA. "Next Generation Service Interfaces Architecture", *Open Mobile Alliance (OMA)*, May 2012.

[27] SUTARIA, R., GOVINDACHARI, R. "Understanding The Internet Of Things", *Electronic Design Magazine*, May 2013. Disponível em: <http://electronicdesign.com/iot/understanding-internet-things>.

[28] COLITTI, W., STEENHAUT, K., CARO, N. D., et al. "Evaluation of Constrained Application Protocol for Wireless Sensor Networks". In: *Local Metropolitan Area Networks (LANMAN), 2011 18th IEEE Workshop on*, pp. 1–6, Chapel Hill, USA, Oct 2011.

[29] IEEE. "IEEE Standard for Low-Rate Wireless Networks", *IEEE Std 802.15.4-2015*, pp. 1–709, April 2016.

[30] HUNKELER, U., TRUONG, H., STANFORD-CLARK, A. "MQTT-S: A Publish/Subscribe Protocol for Wireless Sensor Networks", *Workshop on*

Information Assurance for Middleware Communications (IAMCOM), pp. 1–28, Nov 2008.

[31] YASSEIN, M. B., SHATNAWI, M. Q., AL-ZOUBI, D. "Application layer protocols for the Internet of Things: A survey". In: *2016 International Conference on Engineering MIS (ICEMIS)*, pp. 1–4, Agadir, Morocco, Sept 2016.

[32] NAIK, N. "Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP". In: *2017 IEEE International Systems Engineering Symposium (ISSE)*, pp. 1–7, Vienna, Austria, Oct 2017.

[33] HEDI, I., SPEH, I., SARABOK, A. "IoT network protocols comparison for the purpose of IoT constrained networks". In: *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pp. 501–505, Opatija, Croatia, May 2017.

[34] MIJOVIC, S., SHEHU, E., BURATTI, C. "Comparing application layer protocols for the Internet of Things via experimentation". In: *2016 IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI)*, pp. 1–5, Bologna, Italy, Sept 2016.

[35] KOVATSCH, M., DUQUENNOY, S., DUNKELS, A. "A Low-Power CoAP for Contiki". In: *Mobile Adhoc and Sensor Systems (MASS), 2011 IEEE 8th International Conference on*, pp. 855–860, Valencia, Spain, Oct 2011.

[36] ZHANG, T., LI, X. "Evaluating and Analyzing the Performance of RPL in Contiki". In: *Proceedings of the First International Workshop on Mobile Sensing, Computing and Communication*, MSCC '14, pp. 19–24, New York, USA, Aug 2014.

[37] PALATTELLA, M., ACCETTURA, N., VILAJOSANA, X., et al. "Standardized Protocol Stack for the Internet of (Important) Things", *Communications Surveys Tutorials, IEEE*, v. 15, n. 3, pp. 1389–1406, Jan 2013.

[38] KUSHALNAGAR, N., MONTENEGRO, G., SCHUMACHER, C. "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals", *Internet Engineering Task Force (IETF)*, v. RFC 4919, Aug 2007.

[39] KUSHALNAGAR, N., MONTENEGRO, G., HUI, J., et al. "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", *Internet Engineering Task Force (IETF)*, v. RFC 4944, Sept 2007.

[40] HARTKE, K. "Observing Resources in CoAP", *Internet Engineering Task Force (IETF)*, v. RFC 7641, July 2014.

[41] LUDOVICI, A., GARCIA, E., GIMENO, X., et al. "Adding QoS support for Timeliness to the Observe Extension of CoAP". In: *2012 IEEE 8th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pp. 195–202, Barcelona, Spain, Oct 2012.