Um Mecanismo de Controle de Demanda no Provimento de Serviços de IoT Usando CoAP

Jasiel G. Bahia[1] **e Miguel Elias M. Campista**[1]

[1]Universidade Federal do Rio de Janeiro – PEE/COPPE/GTA
Caixa Postal 68.504 – 21.945-970 – Rio de Janeiro – RJ – Brasil

{jasiel,miguel}@gta.ufrj.br

Abstract. *The Internet of Things (IoT) challenges network scalability, given the huge number of connected devices. Consequently, new 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 and performance analysis of CoAP in a typical IoT industrial scenario to show the influence of network configuration parameters on the protocol performance. Moreover, we propose a mechanism for demand control and selection of observers. The proposed mechanism is based on the radio duty-cycle at the server and its energy consumption for operation mode switching while delivering services. The mechanism is evaluated in a simulator designed for IoT (Cooja) and the results obtained show a significant reduction of energy consumption at the server compared with traditional CoAP.*

Resumo. *A Internet das Coisas (Internet of Things - IoT) desafia a escalabilidade em rede, dado o enorme número de dispositivos interconectados. Consequentemente, novos protocolos vêm sendo propostos, sendo o CoAP (Constrained Application Protocol) um dos principais para a camada de aplicação. Este trabalho apresenta uma análise pioneira de desempenho e escalabilidade do CoAP em um cenário industrial típico de IoT para demostrar a influência dos parâmetros de configuração da rede no desempenho do protocolo. Mais ainda, este trabalho propõe um mecanismo de controle de demanda e de seleção de observadores. O mecanismo proposto 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 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.*

1. Introdução

Bilhões de pessoas utilizam a Internet no mundo para aplicações que vão desde navegação *Web* até interação através de redes sociais [Group 2016]. Ao mesmo tempo em que o número de pessoas conectadas à Internet cresce, a tecnologia embarcada em dispositivos eletrônicos evolui, possibilitando que objetos do cotidiano também sejam habilitados com recursos de comunicação e processamento. Essa evolução leva a um novo paradigma de utilização da Internet, onde os objetos também se conectam e, assim, se transformam em produtores ou consumidores de dados. Esse paradigma, conhecido como Internet das Coisas (IoT - *Internet of Things*) [Al-Fuqaha et al. 2015], já é uma realidade

na indústria, sendo difundido como a quarta revolução industrial sob nomenclaturas como Indústria 4.0, Indústria Conectada e Fábrica Inteligente [Pwc 2016]. A possibilidade de obter informações mais precisas de ambientes e ativos industriais favorece a tomada de decisões e ações mais efetivas, promovendo economia, eficiência e segurança.

A escalabilidade, adaptabilidade, eficiência energética e segurança são requisitos fundamentais de IoT. Na Internet das Coisas, os protocolos devem ser compatíveis com as limitações dos dispositivos, de modo a proporcionar um consumo eficiente da energia e dos recursos de processamento e armazenamento. Com relação ao consumo de energia, é importante otimizar o ciclo de trabalho (*duty-cycle*) do sistema de comunicação, pois esse é o que mais demanda energia do dispositivo. Devido a essas características, os protocolos empregados na Internet convencional não são adequados para uso em dispositivos de IoT [Al-Fuqaha et al. 2015]. Como parte do desenvolvimento de novos mecanismos, a análise de desempenho ante cenários típicos de IoT é necessária para que se verifique o atendimento aos requisitos. A imensa quantidade de dispositivos conectados [Cisco 2016] provoca um aumento substancial no fluxo de dados e desafia o potencial de escalabilidade das redes, especialmente quando compostas por dispositivos limitados. 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.

As propostas de protocolos para IoT, comumente, não realizam análise de desempenho em cenários típicos. De modo geral, os protocolos são ou avaliados qualitativamente ou por meio de experimentos que não consideram as limitações dos dispositivos, como é o caso dos trabalhos apresentados em [Thangavel et al. 2014] e [Talaminos-Barroso et al. 2016]. Nesses trabalhos, as soluções propostas para aumentar a escalabilidade e reduzir o consumo de energia envolvem acordo entre cliente e servidor sobre os critérios para o provimento dos serviços, porém não garantem a disponibilidade dos serviços para clientes prioritários. Essa disponibilidade é essencial em ambientes como os industriais, pois permite o funcionamento correto dos sistemas, o gerenciamento eficiente dos ativos e promove segurança operacional.

Este trabalho visa aumentar o desempenho e a escalabilidade de servidores CoAP (*Constrained Application Protocol*) [Shelby et al. 2014] no provimento de serviços em um cenário industrial típico de IoT. O CoAP foi escolhido por atender aos requisitos das redes IoT e possuir funcionalidades úteis para o contexto industrial, além de ser um dos protocolos de IoT mais difundidos na literatura. O primeiro objetivo deste trabalho é mostrar a influência dos parâmetros de configuração da rede no desempenho do servidor CoAP, considerando como métricas vazão, perdas, quantidade de pacotes CoAP gerados e consumo de energia. O segundo e principal objetivo é propor um mecanismo de controle de demanda e de 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. Esse mecanismo aumenta a eficiência da rede, em termos de escalabilidade e consumo de energia no servidor, além de garantir a disponibilidade dos serviços para os clientes prioritários. Os experimentos foram realizados no Cooja [Thingsquare 2016], o simulador da plataforma de desenvolvimento do Contiki OS, desenvolvido para o teste de redes IoT. Considerou-se um cenário típico IoT na indústria, onde os dispositivos possuem recursos limitados e interagem entre si para realizar funções de controle. O mecanismo proposto é aplicável a qualquer cenário de IoT onde haja a ne-

cessidade de garantir a disponibilidade de um dado servidor para clientes prioritários. 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 prioritários de acordo com as especificações.

O restante deste artigo está organizado da seguinte forma. A Seção 2 apresenta uma arquitetura típica da Internet das Coisas e o funcionamento do CoAP. Na Seção 3, os trabalhos relacionados são listados com ênfase nos protocolos de comunicação para IoT. Na Seção 4, o funcionamento do mecanismo de controle de demanda proposto é apresentado e, a seguir, na Seção 5, as condições de análise do mecanismo são descritas. A análise dos resultados obtidos nos experimentos é discutida na Seção 6, concluindo na Seção 7 com um resumo dos resultados e sugestões de trabalhos futuros.

2. Arquitetura IoT e Protocolo CoAP

Em face das limitações dos dispositivos de IoT, as soluções empregadas na Internet convencional não são compatíveis com os novos cenários introduzidos pelas redes LLN (*Low-power and Lossy Networks*) [Granjal et al. 2015]. Sendo assim, uma nova pilha de protocolos vêm sendo desenvolvida com o objetivo de fomentar aplicações futuras de IoT [Palattella et al. 2013]. Dentro de uma arquitetura orientada a serviço [Xu et al. 2014], a camada de aplicação tem a função de apresentar a informação ao usuário através de uma interface amigável. De acordo com os requisitos da aplicação, a camada de serviço realiza a descoberta e composição de serviços, fazendo uso da infraestrutura da rede IoT para atender às solicitações. Os pedidos podem ser gerados por meio de protocolos padrão de serviços *Web* adaptados às restrições dos dispositivos de IoT, como é o caso do CoAP.

O CoAP é um protocolo de comunicação desenvolvido para dispositivos com recursos limitados e redes LLN [Shelby et al. 2014]. Esses dispositivos possuem baixa capacidade de processamento e armazenamento, enquanto que as redes LLN 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*) [Shelby et al. 2014].

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 pela URI) do servidor. O servidor então responde com o código de resposta, podendo incluir uma representação do recurso (p. ex., valor da temperatura). Diferentemente do HTTP, o CoAP lida com essas transações assincronamente por meio de transporte orientado a da-

tagrama, utilizando o UDP. Isso é feito logicamente usando uma camada de mensagens que suporta confiabilidade opcional (com *backoff* exponencial). Já retransmissões e reordenamento são implementados na própria camada de aplicação, excluindo a necessidade do TCP e permitindo o uso do protocolo em dispositivos limitados.

O CoAP define quatro tipos de mensagens: 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. O CoAP pode ser visto como um protocolo de duas camadas: uma camada de mensagens, usada para lidar com as interações assíncronas sobre UDP, e outra camada usada para lidar com interações de pedido/resposta. Essas duas camadas virtuais são integradas no cabeçalho das mensagens CoAP. Cada mensagem contém um identificador usado para detectar duplicatas e para 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 recipiente envie uma mensagem ACK com o mesmo identificador da mensagem CON original. Quando o recipiente 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. Se o servidor receber o pedido, mas não puder 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, contendo o conteúdo solicitado, e o cliente confirma o recebimento com um ACK. O CoAP utiliza um subconjunto dos métodos do HTPP (GET, PUT, POST e DELETE), os quais funcionam de maneira similar, porém ajustados para uso em redes LLN.

Uma aplicação de IoT frequentemente envolve um dispositivo transmitindo informação sobre os seus sensores a outros dispositivos clientes. O CoAP suporta uma funcionalidade chamada de Observação [Hartke 2014], onde o cliente pode registrar-se como observador de um recurso (sujeito) do servidor através de um GET modificado. O servidor estabelece um relacionamento de observação entre o cliente e o recurso, sendo que cada recurso tem a sua própria lista de observadores. Cada cliente só pode se registrar uma única vez em uma lista. Essa funcionalidade reduz o congestionamento na fila de entrada do servidor, pois 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 extensão do CoAP permite que os clientes observem as mudanças nos recursos de seu interesse sem gerar novos pedidos, reduzindo a sobrecarga, o uso de banda, a latência e aumentando a confiabilidade em comparação com o HTTP [Ludovici et al. 2012]. Porém, é importante lembrar que o tamanho da lista de observadores está limitado à capacidade de armazenamento do dispositivo. 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, otimizando, assim, o uso dos recursos do servidor. Esse recurso de Observação do CoAP, é muito importante para a escalabilidade e para o controle do consumo de energia, garantindo um maior tempo de disponibilidade

do servidor. A Figura 1 ilustra a comunicação com o servidor quando o cliente interage sob demanda e quando o cliente se registra como observador.

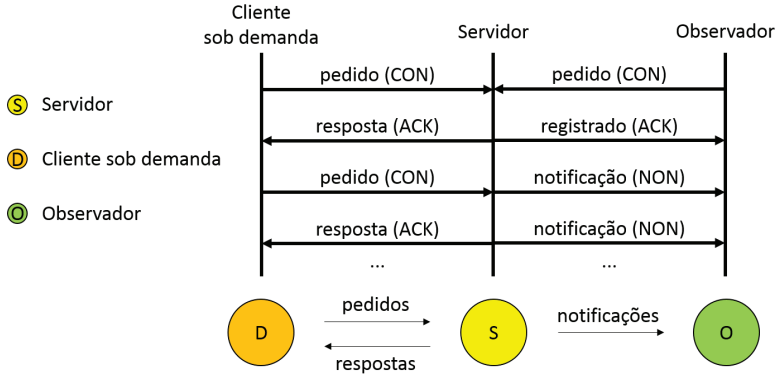


Figura 1. Interação cliente/servidor com e sem Observação.

3. Trabalhos Relacionados

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

Sutaria et al. [Sutaria and Govindachari 2013] comparam o HTTP e o CoAP em termos de tamanho de pacote e consumo de energia por pedido. No artigo, é mostrado que um pedido para um servidor HTTP implementado no Contiki OS, quando feito com o CoAP, usa 154 bytes; quando feito com o HTTP, usa 1451 bytes, consumindo 0,774 mW e 1,333 mW, respectivamente. Já Colitti et al. [Colitti et al. 2011] mostram que o CoAP gasta menos tempo na transmissão do pedido e consome menos energia que o HTTP. Dois dos protocolos de comunicação mais difundidos para dispositivos limitados são o CoAP e o MQTT (*Message Queue Telemetry Transport*). O MQTT [Banks and Gupta 2014], desenvolvido pela IBM, é um protocolo do tipo *publish/subscribe*, assim como o CoAP, porém utiliza o TCP. O uso do TCP não é indicado para redes LLN e dispositivos mais limitados, visto que exige mais memória e processamento, além de estar sujeito à instabilidade causada pelas 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. 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. Além disso, os endereços utilizados pelo MQTT são definidos por longas cadeias de caracteres, sendo incompatível com a tecnologia de rede IEEE 802.15.4. O MQTT-SN [Hunkeler et al. 2008] é uma alternativa que utiliza o UDP e tenta prover uma abstração para comunicação assíncrona.

Com relação a análise de desempenho de protocolos para IoT, Thangavel et al. [Thangavel et al. 2014] analisam o desempenho do MQTT e do CoAP em termos de atraso fim-a-fim e consumo de banda, quando usados em conjunto do *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. Kovatsch et al. [Kovatsch et al. 2011] 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. [Zhang and Li 2014] 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 um cenário típico de IoT contendo dispositivos com recursos limitados.

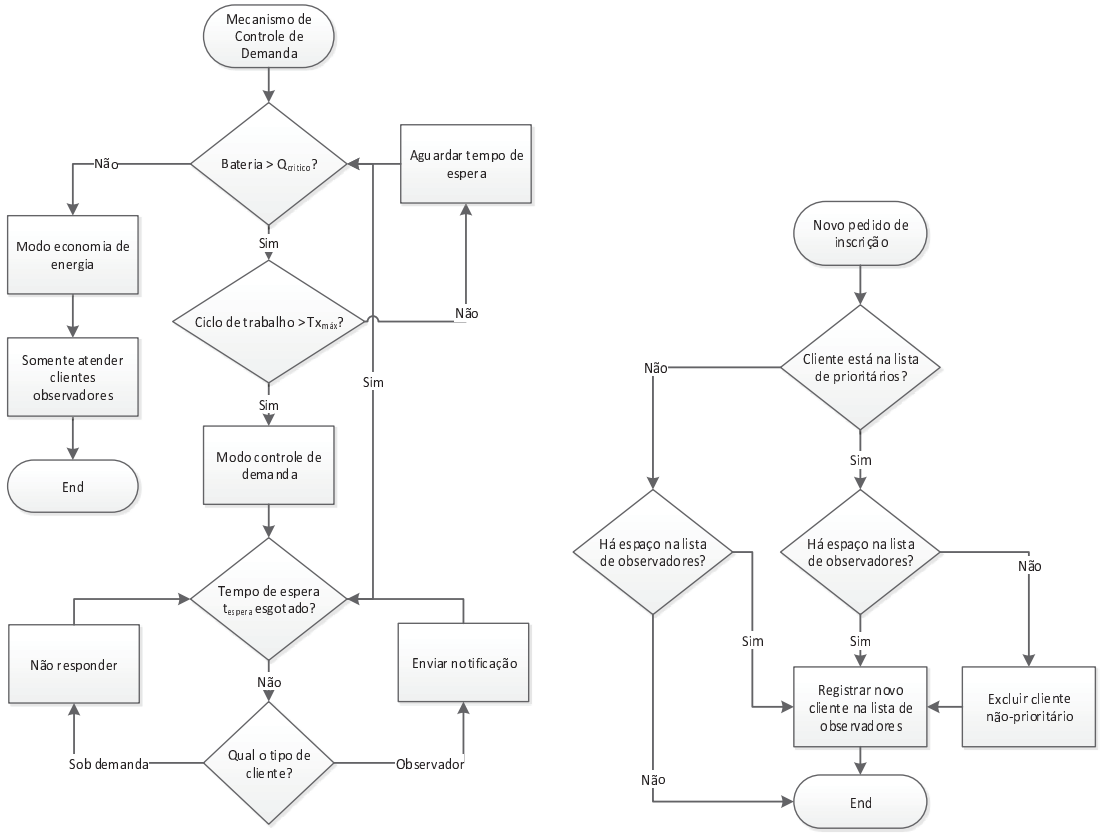
Pelo levantamento feito, não foi identificado nenhum trabalho de análise de desempenho e escalabilidade de servidores CoAP em redes IoT compostas por dispositivos com recursos limitados. Além disso, não foram encontrados outros trabalhos que mencionassem mecanismos de controle de demanda para servidores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

4. Mecanismo de Controle de Demanda Proposto

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 industrial IoT.

Considerando o caso em que os pedidos CoAP gerados pelos clientes sob demanda tenham uma periodicidade t_n, o número de pedidos gerados (demanda D) por um cliente, pode ser estimado, fazendo $D = \frac{t_{op}}{t_n}$, onde t_{op} é o tempo total de operação do dispositivo cliente. Já para um cliente registrado na lista de observadores de um recurso do servidor, só é gerado um pacote de pedido CoAP. Como observador, esse cliente só recebe pacotes CoAP de notificação sobre o recurso em observação com periodicidade t_p. O tamanho máximo da lista de observadores depende do espaço na memória do servidor e a redução de demanda depende do número de clientes observadores (p) registrados na lista e da relação entre t_n e t_p. 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 observadores. Nesse modo de operação, 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 tempo de espera t_{espera} não é completado, 1) Quando receber um pedido de cliente sob demanda, não responder; 2) Quando for um pacote para cliente observador, enviar a notificação. 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. Através desse mecanismo, é possível controlar o consumo de energia do servidor CoAP de forma a garantir a sua disponibilidade para os clientes observadores.

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 os clientes poderão ser registrados na lista de observadores de um recurso do servidor CoAP. Para tanto, o servidor deve armazenar, para cada recurso, os endereços 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. A Figura 2 mostra o funcionamento do mecanismo de controle de demanda e seleção de observadores.



(a) Algoritmo de controle de demanda. (b) Algoritmo de seleção de observadores.

Figura 2. Funcionamento do mecanismo proposto.

Para configurar o mecanismo, é preciso definir o ciclo de trabalho máximo (Tx_{max}) desejado para o rádio do servidor. O ciclo de trabalho representa a fração do período de tempo em que o rádio é efetivamente utilizado, ou seja, $Tx = \frac{t_{TX}}{t_{op}}$, sendo t_{TX} o tempo total de uso do rádio e t_{op} o tempo total de operação do servidor. Para definir Tx_{max}, considera-se a energia total disponível na bateria (Q_{max}), o tempo total de operação desejado (t_{op}), o período de notificação necessário (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). A energia necessária para enviar uma notificação é $Q_{tx} = (P_{tx} \times t_{tx})$ e o número de notificações por observador é $D_p = \frac{t_{op}}{t_p}$. Assim, o produto $p \times D_p \times Q_{tx}$ fornece a energia total consumida pelo rádio, ou seja, $Q_{max} = \left(p \times \frac{t_{op}}{t_p} \times P_{tx} \times t_{tx}\right)$. 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, $t_{TX} = \left(p \times \frac{t_{op}}{t_p} \times t_{tx}\right)$. Dessa forma, obtém-se que $Tx_{max} = \frac{Q_{max}}{P_{tx} \times t_{op}}$. Note que quanto maior o tempo de operação desejado, menor deve ser ciclo de trabalho para a mesma energia disponível.

Os valores de Tx_{max} e de t_{espera} 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. Outro parâmetro que deve ser definido é o nível crítico de carga da bateria $Q_{critico}$, o qual 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.

5. Análise do Mecanismo de Controle de Demanda

Esta seção apresenta as condições de análise do mecanismo de controle de demanda para servidores CoAP, as métricas utilizadas e o cenário considerado. Antes, porém, é necessário introduzir o ambiente de simulação utilizado nos experimentos.

5.1. Ambiente de simulação

A ferramenta utilizada nos experimentos foi o Cooja, que é o simulador de redes LLN da plataforma de desenvolvimento do Contiki OS [Thingsquare 2016]. O Contiki 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 IoT. O Contiki suporta inteiramente os padrões IPv4 e IPv6, além dos padrões criados recentemente para redes LLN, como 6LoWPAN, RPL e CoAP. O Contiki também possui mais de uma implementação da camada MAC para dispositivos de baixa potência (IEEE 802.15.4), sendo o ContikiMAC a principal.

A plataforma de desenvolvimento do Contiki 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 um cenário industrial típico de IoT, onde a rede é composta por dispositivos 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, através da Internet, podendo atender a clientes internos e externos. 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 a Internet, divulgando os recursos disponíveis. Um exemplo de cenário industrial típico é o sistema de monitoramento e controle de dutos de recebimento de petróleo em terminais de armazenamento. Esse sistema faz o monitoramento da vazão (recurso) no duto de petróleo que alimenta tanques, bombas de transferência e outros dutos. A prioridade no monitoramento deve ser para os sistemas diretamente afetados pelo duto de petróleo (observadores prioritários). Enquanto o servidor 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). A Figura 3 ilustra a rede IoT considerada nos experimentos.

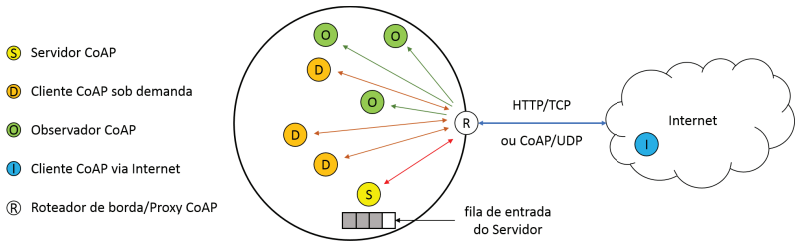


Figura 3. Um cenário industrial típico de IoT.

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. A comunicação entre os clientes e o servidor ocorre sempre com a intermediação 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.

As simulações foram executadas em uma máquina virtual rodando o Contiki 3.0. A tecnologia utilizada nos dispositivos da rede foi o módulo TMote Sky (Mote IV), que utiliza a tecnologia de rede IEEE 802.15.4 e possui 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, NullMAC e tamanho 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 mensagens CoAP em espera na fila do servidor.

Nos experimentos, os clientes CoAP podem ser clientes observadores (notificados periodicamente) ou clientes sob demanda. Para cada configuração, a periodicidade de notificações (t_p) do servidor para os observadores foi mantida, enquanto que os parâmetros variáveis foram: número total de clientes (n), periodicidade de requisições do cliente sob demanda (t_n) e número de observadores (p). O número máximo de clientes no simulador é restrito, dada a limitação do próprio padrão CoAP [Shelby et al. 2014]. Em vista disso, o número de clientes total foi variado entre $n = 1$ e $n = 10$.

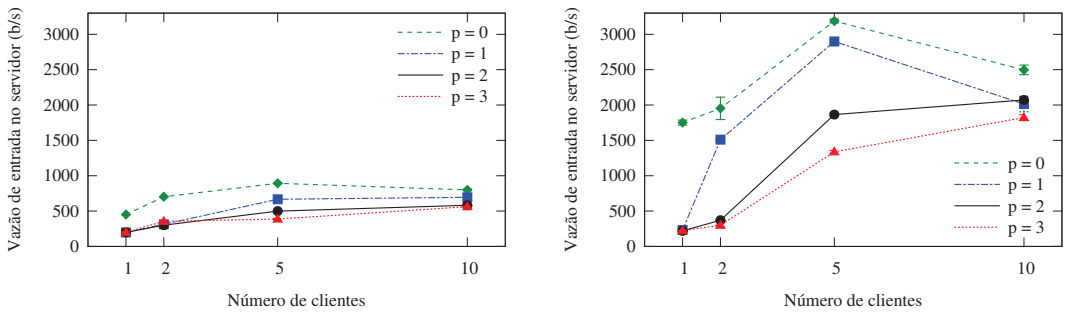
As métricas de desempenho utilizadas na análise do servidor CoAP foram: quantidade de pacotes CoAP gerados, perdas, vazão de dados e consumo de energia. Os intervalos de confiança mostrados nos gráficos são de 95%, representados por barras verticais.

6. Análise dos Resultados

Nesta seção, a influência dos parâmetros de configuração da rede no desempenho do servidor CoAP é discutida. Em seguida, os resultados dos experimentos utilizando o mecanismo de controle de demanda proposto são apresentados.

6.1. Influência dos parâmetros de configuração no desempenho do servidor CoAP

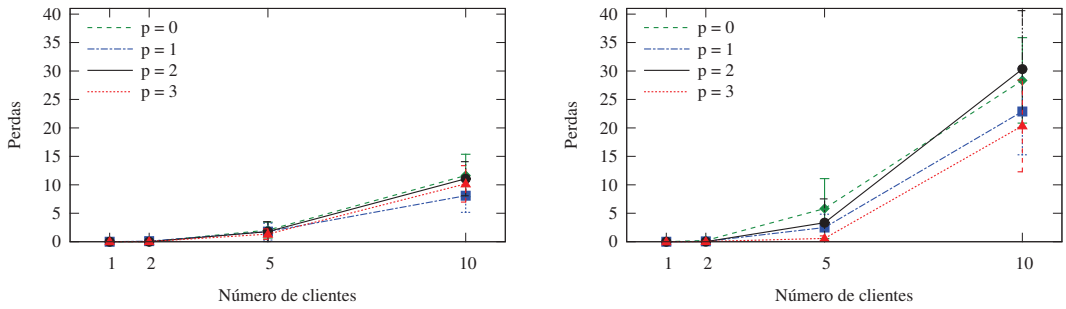
Vazão de dados: Avaliando a vazão de dados no servidor, observa-se que ela chega a atingir cerca de 3200 b/s para ($n = 5; p = 0; t_n = t_p/2$), conforme Figura 4(b). Essa taxa máxima é compatível com aquela esperada para redes LLN (Seção 2). Porém, a limitação da fila de entrada e da capacidade de processamento do servidor provoca perdas de pacotes à medida que a diferença entre o número de clientes não-observadores e observadores ($n - p$) e a demanda aumentam, reduzindo, assim, a vazão. A partir da análise do tráfego da rede, percebe-se que as perdas ocorrem sobretudo quando, antes do servidor responder a um dado pedido, o número de pedidos recebidos excede o tamanho da fila ou quando o servidor demora mais que o tempo de estouro do cliente para responder. A Figura 4 mostra também que o fluxo de dados na entrada do servidor, para o mesmo n, é reduzido com o aumento de p, especialmente quando a periodicidade de pedidos é menor que a de notificações. 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 a entrada do servidor. Logo, o tamanho da lista de observadores (uso de memória), o tamanho da fila de entrada no servidor (uso de memória) e a capacidade de processamento têm papel fundamental no desempenho do servidor com o aumento da demanda.



(a) $t_n = 2t_p$ (b) $t_n = t_p/2$

Figura 4. Vazão de dados no servidor.

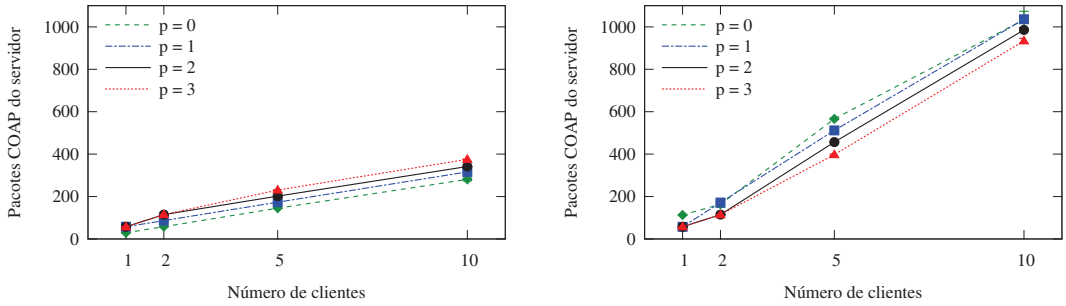
Perdas de pacotes: Na Figura 5, observa-se que o número médio de perdas se torna maior à medida que a demanda de clientes não-observadores aumenta (aumento da diferença $n - p$ e redução de t_n), chegando a representar um pouco mais de 3% dos pedidos para $(n = 10; p = 2; t_n = t_p/2)$. Pelo mesmo motivo da redução da vazão, o aumento das perdas é consequência da limitação do tamanho da fila de entrada do servidor utilizado (4 pacotes CoAP no máximo). Uma solução, portanto, é utilizar um servidor com maior capacidade de memória, permitindo um tamanho de fila maior.



(a) $t_n = 2t_p$ (b) $t_n = t_p/2$

Figura 5. Perdas de pacotes CoAP no servidor.

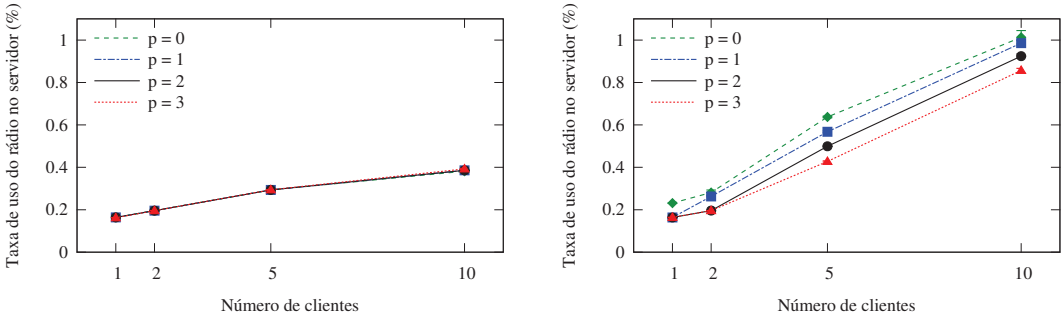
Número de pacotes CoAP: A Figura 6 mostra a evolução no número de pacotes CoAP gerados pelo servidor com o aumento da demanda (aumento de $n - p$ e redução de t_n). Observa-se que, para pedidos sob demanda com periodicidade $t_n = 2t_p$, o valor de p dita a quantidade de pacotes gerados. Porém, para $t_n = t_p/2$, o número de respostas a pedidos sob demanda é maior que o número de notificações por intervalo de tempo. Logo, a quantidade de clientes sob demanda é que dita a quantidade de pacotes gerados pelo servidor. O mecanismo de controle de demanda deve atuar nesse tipo de situação, a fim de manter o uso do rádio do servidor dentro dos limites especificados.



(a) $t_n = 2t_p$ (b) $t_n = t_p/2$

Figura 6. Pacotes CoAP transmitidos pelo servidor.

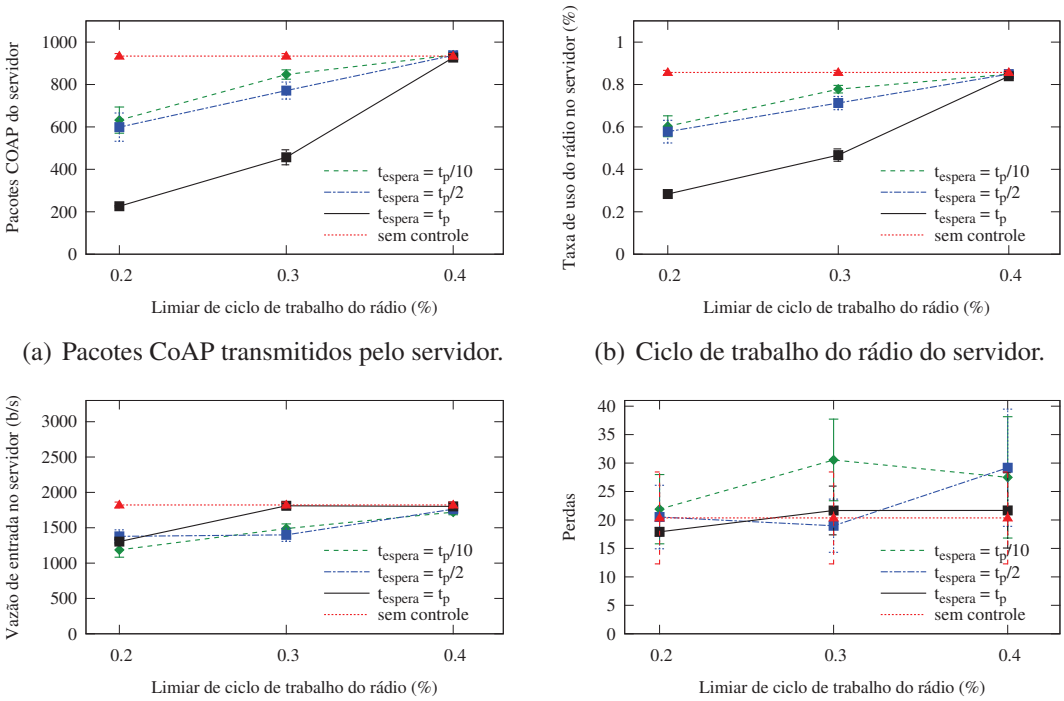
Consumo de energia: Os resultados apresentados na Figura 7 representam a percentagem de tempo em que o servidor utilizou o rádio para receber ou transmitir dados. A taxa de uso do rádio foi usada para avaliar o consumo de energia porque, além de serem diretamente proporcionais (vide folha de dados do dispositivo), a análise do ciclo de trabalho permite avaliar o quanto o sistema acompanha os parâmetros de configuração do



(a) $t_n = 2t_p$ (b) $t_n = t_p/2$

Figura 7. Ciclo de trabalho do rádio do servidor.

mecanismo proposto. Observa-se que, para $t_n = 2t_p$, o consumo de energia é praticamente o mesmo para as diversas configurações de rede. Porém, quando $t_n = t_p/2$, 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.



(a) Pacotes CoAP transmitidos pelo servidor.
(b) Ciclo de trabalho do rádio do servidor.
(c) Vazão de dados no servidor.
(d) Perdas de pacotes CoAP no servidor.

Figura 8. Desempenho do Mecanismo proposto.

6.2. Desempenho do Mecanismo Proposto de Controle de Demanda

Conforme observado na análise de desempenho do servidor CoAP, considerando o cenário descrito com três clientes prioritários ($p = 3$), o pior caso de demanda é aquele com menor período entre pedidos e maior número de clientes sob demanda. Sendo assim, o mecanismo de controle de demanda foi analisado para o caso ($n = 10; t_n = t_p/2; p =$

3). Os parâmetros de configuração do mecanismo foram Tx_{max} e t_{espera}, tomando como referência o ciclo de trabalho na transmissão encontrado para ($n = 10; t_n = 2t_p$). A Figura 8 mostra que o mecanismo conseguiu controlar a taxa de uso do rádio, reduzindo a quantidade de pacotes gerados (Figura 8(a)) e economizando a energia do servidor (Figura 8(b)). A redução de uso do rádio do servidor foi de cerca de 60% para a configuração ($Tx_{max} = 0.2; t_{espera} = t_p$). O uso do mecanismo provoca uma redução na vazão de entrada do servidor (Figura 8(c)), como esperado, porém não foi observado grande impacto no número de perdas na maioria dos casos. Isso mostra que, neste caso em específico, mesmo com o uso do mecanismo, os clientes sob demanda não foram tão prejudicados.

7. Conclusões e Trabalhos Futuros

As principais contribuições deste artigo são a análise de desempenho e escalabilidade de servidores CoAP em um cenário industrial típico de IoT e a proposta de um novo mecanismo de controle de demanda e seleção de observadores. Ressalta-se que este trabalho apresenta uma proposta inédita para economia de energia em redes que usam servidores CoAP. A escolha do CoAP para a aplicação do mecanismo proposto deve-se ao fato do CoAP, além de possuir a funcionalidade de Observação, atender aos requisitos de IoT e de redes LLN compostas por dispositivos com recursos limitados, condições fundamentais para uso no cenário considerado neste trabalho. 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.

Como trabalhos futuros, pretende-se desenvolver um mecanismo de provimento de serviços adaptável ao contexto, considerando situações mais complexas como o uso de mais de um servidor. Sugere-se também a realização de testes reais, não só para avaliar o desempenho da proposta, mas também para verificar a representatividade do simulador Cooja. Finalmente, a análise de outros protocolos de comunicação para comparação com o CoAP poderá ser feita tomando este trabalho como referência.

Referências

Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., and Ayyash, M. (2015). Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Communications Surveys Tutorials*, 17(4):2347–2376.

Banks, A. and Gupta, R. (2014). MQTT Version 3.1.1. *OASIS Standard*.

Cisco (2016). Cisco Visual Networking Index: Global Mobile Data Traffic Forecast. [Online: 15-November-2016].

Colitti, W., Steenhaut, K., Caro, N. D., Buta, B., and Dobrota, V. (2011). Evaluation of Constrained Application Protocol for Wireless Sensor Networks. In *Local Metropolitan Area Networks (LANMAN), 2011 18th IEEE Workshop on*, pages 1–6.

Granjal, J., Monteiro, E., and Sa Silva, J. (2015). Security for the Internet of Things: A Survey of Existing Protocols and Open Research Issues. *Communications Surveys Tutorials, IEEE*, 17(3):1294–1312.

Group, M. M. (2016). Internet World Stats. [Online: 18-November-2016].

Hartke, K. (2014). Observing Resources in CoAP. *Internet Engineering Task Force (IETF)*, RFC 7641.

Hunkeler, U., Truong, H., and Stanford-Clark, A. (2008). MQTT-S: A Publish/Subscribe Protocol for Wireless Sensor Networks. *Workshop on Information Assurance for Middleware Communications (IAMCOM)*.

Kovatsch, M., Duquennoy, S., and Dunkels, A. (2011). A Low-Power CoAP for Contiki. In *Mobile Adhoc and Sensor Systems (MASS), 2011 IEEE 8th International Conference on*, pages 855–860.

Ludovici, A., Garcia, E., Gimeno, X., and Auge, A. C. (2012). 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)*, pages 195–202.

Palattella, M., Accettura, N., Vilajosana, X., Watteyne, T., Grieco, L., Boggia, G., and Dohler, M. (2013). Standardized Protocol Stack for the Internet of (Important) Things. *Communications Surveys Tutorials, IEEE*, 15(3):1389–1406.

Pwc (2016). Industry 4.0: Building the Digital Enterprise. [Online: 15-November-2016].

Shelby, Z., Hartke, K., and Bormann, C. (2014). The Constrained Application Protocol (CoAP). *Internet Engineering Task Force (IETF)*, RFC 7252.

Sutaria, R. and Govindachari, R. (2013). Understanding The Internet Of Things. *Electronic Design Magazine*.

Talaminos-Barroso, A., Estudillo-Valderrama, M. A., Roa, L. M., Reina-Tosina, J., and Ortega-Ruiz, F. (2016). A Machine-to-Machine protocol benchmark for eHealth applications - Use case: Respiratory rehabilitation. *Computer Methods and Programs in Biomedicine*, 129:1–11.

Thangavel, D., Ma, X., Valera, A., Tan, H. X., and Tan, C. K. Y. (2014). 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*, pages 1–6.

Thingsquare (2016). Contiki: The Open Source OS for the Internet of Things. [Online: 14-November-2016].

Xu, L. D., He, W., and Li, S. (2014). Internet of Things in Industries: A Survey. *Industrial Informatics, IEEE Transactions on*, 10(4):2233–2243.

Zhang, T. and Li, X. (2014). Evaluating and Analyzing the Performance of RPL in Contiki. In *Proceedings of the First International Workshop on Mobile Sensing, Computing and Communication*, MSCC '14, pages 19–24.