Um Mecanismo de Comutação de Servidores CoAP para Aumento de Disponibilidade dos Serviços de IoT

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

[1]Universidade Federal do Rio de Janeiro – GTA/PEE-COPPE/DEL-Poli

{jasiel,miguel}@gta.ufrj.br

Abstract. *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 propose a mechanism that conducts CoAP servers selection and switching at the client, aiming to increase services availability for an application. The mechanism 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 that the mechanism increases the reliability and the robustness of the IoT service provisioning and, furthermore allows energy consumption balance among the servers.*

Resumo. *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 um mecanismo de seleção e comutação de servidores CoAP que visa aumentar a disponibilidade dos serviços necessários a uma aplicação. 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 que o mecanismo aumenta a confiabilidade e a robustez na obtenção de serviços de IoT, além de permitir o balanceamento do consumo de energia entre os servidores.*

1. Introdução

Bilhões de pessoas utilizam a Internet no mundo para aplicações que vão desde navegação *web* até interação em redes sociais [Group 2016]. Ao mesmo tempo em que o número de usuários da 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, onde os objetos também se conectam e interagem com o ambiente, transformando-se em produtores e consumidores

de dados. Esse novo paradigma, conhecido como Internet das Coisas (IoT - *Internet of Things*) [Al-Fuqaha et al. 2015], já é uma realidade na indústria, onde se busca constantemente eficiência no uso dos recursos e disponibilidade dos serviços.

A ampliação da diversidade de aplicações através da Internet das Coisas implica uma imensa quantidade de dispositivos conectados, exigindo um compromisso entre a grande quantidade de dados gerados e as limitações dos dispositivos em redes de IoT. Dessa forma, para desenvolver soluções adequadas, é necessário atentar para aspectos como: escalabilidade da rede, consumo 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 [Al-Fuqaha et al. 2015], já que o consumo de energia não foi um pré-requisito. Como parte do desenvolvimento de novos protocolos, é importante que a análise de desempenho dos mecanismos empregados considere as limitações dos dispositivos de IoT em termos de processamento, armazenamento e energia. Além disso, a imensa quantidade de dispositivos conectados [Cisco 2017] 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*) [Granjal et al. 2015].

Na literatura, as restrições de bateria dos dispositivos de IoT provocam uma grande busca por novos mecanismos mais eficientes energeticamente. Há propostas que reduzem o consumo de energia através do aumento da eficiência no roteamento [Barbato et al. 2013, Iova et al. 2014], outros pela redução do número de pacotes transmitidos [Jan et al. 2014] e outros ainda pelo uso de controle adaptativo do ciclo de trabalho do dispositivo [Afzal et al. 2017]. Todos esses trabalhos, porém, não aliam a eficiência energética à garantia de confiabilidade e disponibilidade dos serviços de IoT. 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 [FIWARE 2011, Banks and Gupta 2014, Jeyaraman and Telfer 2012]. Como consequência, 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. Dentre os principais protocolos desenvolvidos para redes LLN, o CoAP (*Constrained Application Protocol*) [Shelby et al. 2014], 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. A comutação de servidores é essencial sobretudo em redes LLN, onde os servidores podem ficar indisponíveis por alguma instabilidade ou mesmo devido a mecanismos de controle de demanda [Bahia and Campista 2017].

Este trabalho visa aumentar a confiabilidade e a robustez na obtenção de serviços de IoT, através do aumento da disponibilidade de servidores CoAP. O principal objetivo é propor um mecanismo que faça a seleção e comutação de servidores CoAP, baseando-se numa lista ordenada de servidores. A lista é construída e fornecida por uma infraestrutura central de acordo com os requisitos da aplicação do cliente. Cada cliente, então, armazena a lista e realiza a comutação de servidores segundo a ordem de prioridade pré-definida, de modo a permitir a obtenção dos serviços necessários à sua própria 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 [Thingsquare 2016], que é o simulador da plataforma de desenvolvimento do Contiki OS, desenvolvido para testes de redes LLN. Os experimentos consideram um cenário de IoT no qual os dispositivos possuem recursos limitados e interagem entre si para realizar funções de controle, podendo haver recursos redundantes que permitam o aumento da disponibilidade dos serviços necessários às aplicações dos clientes. Os resultados mostram que o mecanismo aumenta a robustez dos serviços e permite o balanceamento do consumo de energia entre os servidores, mediante à comutação de servidores.

Este artigo está organizado da seguinte forma. A Seção 2 apresenta uma arquitetura típica de IoT e o funcionamento do CoAP. Na Seção 3, o funcionamento do mecanismo de seleção e comutação de servidores é apresentado e, a seguir, na Seção 4, as condições de análise do mecanismo são descritas. A análise dos resultados é discutida na Seção 5. Por fim, a Seção 6 conclui este trabalho e sugere direções futuras.

2. Visão Geral de IoT e do Protocolo CoAP

Um dos principais objetivos da Internet das Coisas (IoT) é promover a interoperabilidade entre os dispositivos de redes LLN [Granjal et al. 2015] e as aplicações da Internet convencional. As redes LLN são compostas por dispositivos limitados em capacidade de processamento, armazenamento e energia, que podem assumir o papel de servidor ou cliente conforme a aplicação. No papel de servidor, os dispositivos disponibilizam seus recursos para clientes na rede local e na Internet. Já no papel de cliente, os dispositivos consultam 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 entre si e à Internet, divulgando os recursos disponíveis dos servidores e fazendo a tradução entre os protocolos (p. ex., HTTP/CoAP). Os dispositivos de IoT tipicamente se comunicam através de uma rede sem fio, utilizando um rádio de baixa potência, tornando-se mais susceptível a perdas.

As soluções empregadas na Internet convencional não são compatíveis com os novos cenários introduzidos pelas redes LLN, dadas as limitações dos dispositivos de IoT e o nível de escalabilidade exigido. Sendo assim, uma nova pilha de protocolos vem sendo desenvolvida para fomentar aplicações futuras de IoT [Palattella et al. 2013, Granjal et al. 2015]. Nessa nova pilha, o protocolo proposto para a camada de aplicação é o CoAP [Shelby et al. 2014], projetado para aplicações M2M (*Machine-to-Machine*). O CoAP emprega 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 alta simplicidade

para ambientes com recursos limitados. 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 sobre um recurso (identificado pelo URI) do servidor. O servidor então responde, podendo incluir uma representação do recurso, p. ex., o valor da temperatura. 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.

Aplicações de IoT frequentemente envolvem dispositivos desempenhando ou o papel de servidor ou o papel de cliente de um dado recurso. Nessa direção, o CoAP suporta uma funcionalidade chamada de Observação [Hartke 2014], na qual 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. Um cliente registrado como observador de um recurso recebe notificações do servidor sem a necessidade de gerar pedidos adicionais, como no caso dos clientes chamados sob demanda. Contudo, a lista de observadores de um recurso tem um tamanho máximo, que depende do espaço disponível em memória no servidor para armazenamento das listas de cada recurso. A função Observação do CoAP evita que o cliente envie uma requisição ao servidor sempre que desejar um dado ou que tenha que manter uma sessão aberta, como no caso do HTTP sobre o TCP. Essa é uma das características que torna o CoAP indicado para redes LLN, já que permite que os clientes observadores acompanhem as mudanças nos recursos de seu interesse ao mesmo tempo que reduz a sobrecarga do servidor e da rede [Ludovici et al. 2012]. A Figura 1 ilustra uma arquitetura típica de IoT composta por nós servidores (nós **s**), nós clientes (nós **c**) e um roteador de borda **r** que interconecta a rede de IoT à Internet. Note ainda a presença de nós clientes observadores (nós **o**) e a presença de nós na Internet, que podem se comunicar com a rede de IoT usando HTTP/TCP ou CoAP/UDP, por exemplo.

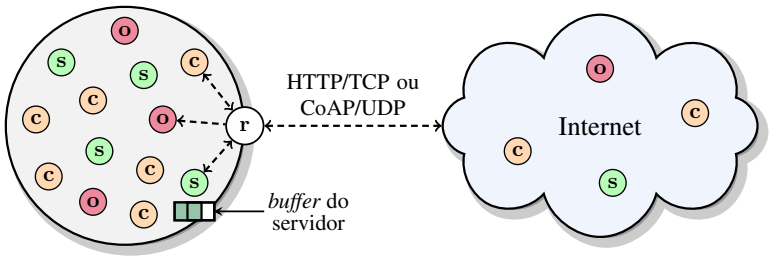


Figura 1. Arquitetura de uma rede de IoT com a presença de observadores.

3. Mecanismo de Comutação de Servidores CoAP Proposto

O principal objetivo do mecanismo proposto é aumentar a disponibilidade da rede e, consequentemente, dos serviços de IoT prestados. Para isso, 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, papel aqui desempenhado pelo *gateway* (roteador de borda) da rede de IoT. Essa infraestrutura, dentre outras funções, gerencia os serviços disponibilizados na rede local e para a 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 e vagas na lista de observadores). A atualização e o gerenciamento dessas informações é possível utilizando o próprio CoAP [Shelby et al. 2014]. Esta seção apresenta primeiramente a *obtenção da lista de servidores* e, em seguida, o processo de *comutação de servidores*.

Obtenção da lista de servidores: O cliente pode solicitar uma lista de servidores que ofereçam o mesmo tipo de serviço (p. ex., medição de temperatura) ou serviços de tipos diferentes (p. ex., medição de temperatura e umidade) para atender 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 os requisitos para que seja gerada uma lista ordenada da forma desejada. Esses requisitos são usados para caracterizar os servidores através de parâmetros como: número de vagas na lista de observadores, grau de confiabilidade da informação, localização do dispositivo e taxa de uso do rádio (demanda). Se o requisito para ordenação for, por exemplo, servidores com o serviço desejado que estejam mais próximos de uma determinada localização, o *gateway* deve calcular o grau de proximidade dos servidores candidatos e posicioná-los na lista de servidores por ordem de proximidade.

A obtenção da lista de servidores 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 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, fazendo consultas sob demanda sempre que necessitar 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.

Comutação de servidores: O mecanismo de comutação de servidores proposto assume que uma lista de endereços de servidores ordenada de acordo com a aplicação já exista. Sendo assim, o cliente realiza a consulta aos servidores seguindo a ordem da lista recebida. A cada consulta, caso o servidor responda, o cliente atualiza o status do servidor para disponível; caso contrário, o servidor é considerado indisponível. 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. 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.

No mecanismo proposto, 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, caso a lista seja composta por servidores que oferecem o mesmo tipo de serviço. 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 no provimento de serviços de IoT. Destaca-se que as arquiteturas propostas para IoT, em muitos casos, consideram que a disponibilização dos recursos dos servidores é feita por um servidor central ou um *broker*. 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 possuírem um mecanismo de controle de demanda baseado no ciclo de trabalho, como proposto em trabalho anterior [Bahia and Campista 2017], 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). Isso permite um balanceamento de carga e de consumo de energia entre os servidores.

Em caso de indisponibilidade do servidor, o cliente pode perder pacotes (pedidos e notificações) até identificar o problema. Nesse caso, um cliente sob demanda, ao não receber resposta a um pedido, realiza retransmissões até atingir o número máximo 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. No mecanismo proposto, o cliente pode efetuar a comutação para outro servidor da lista. No caso do cliente observador de um recurso, o servidor passa a condição de indisponível quando o cliente deixa 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 já tendo comutado 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 recomposição da lista.

A configuração do mecanismo de comutação de servidores requer a definição dos 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 oferecido 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 é 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 ficar muito defasada.

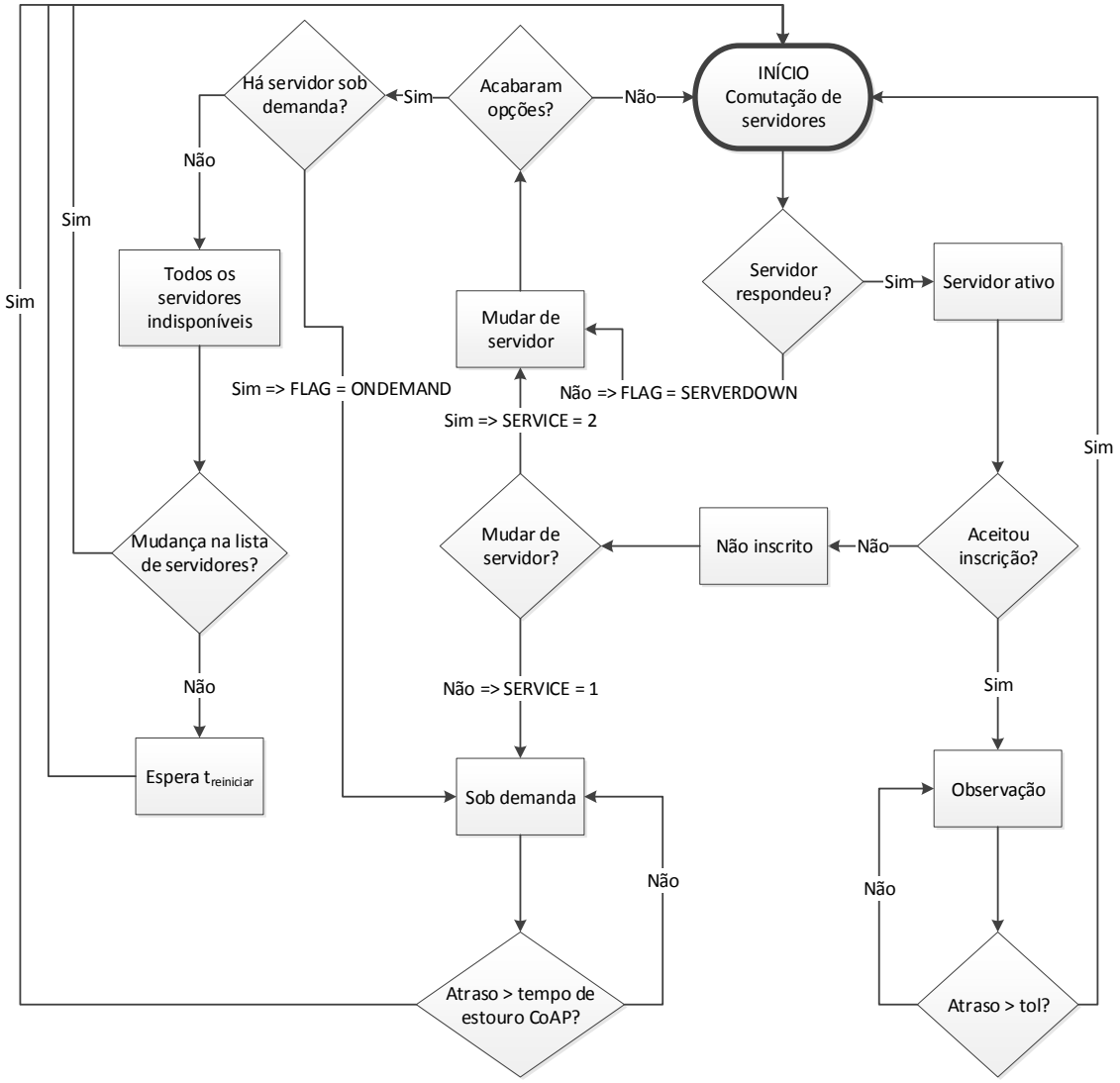


Figura 2. Funcionamento do mecanismo de comutação de servidores.

O mecanismo proposto 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. Além disso, a comutação de servidores CoAP promove o aumento da disponibilidade dos serviços de IoT. A Figura 2 mostra o fluxograma do funcionamento do mecanismo de comutação de servidores.

4. Análise do Mecanismo de Comutação de Servidores

Esta seção apresenta as condições de análise do mecanismo proposto.

4.1. 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 [Thingsquare 2016] 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. 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.

4.2. Experimentos

Os experimentos consideram uma rede de IoT LLN, semelhante a da Figura 1, composta por dez clientes CoAP, um número variável de servidores CoAP e um roteador de borda (*gateway*). Os servidores possuem recursos redundantes e o *gateway* executa o protocolo RPL. A comunicação entre os clientes e o servidor ocorre sempre com a ajuda do roteador de borda, tornando indiferente ao servidor se o pedido veio de um cliente interno ou externo. Para cada configuração, a duração da simulação foi de 10 minutos. O posicionamento dos dispositivos na rede muda aleatoriamente a cada rodada de simulação.

As simulações foram executadas em uma VM 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 48kB de memória Flash. Para reduzir o uso de memória dos dispositivos, foi utilizada a 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 de observadores (p) e tamanho da lista de servidores (n_s). 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.

A análise do servidor CoAP usou as seguintes métricas: quantidade de pacotes CoAP gerados, taxa de perdas, carga na entrada do 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. A carga no servidor é a quantidade de dados recebidos e processados pelo servidor dividida por um intervalo de tempo (Δt), que, neste trabalho, é o tempo decorrido entre o primeiro pacote CoAP gerado na rede e o último. 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.

5. Análise dos Resultados

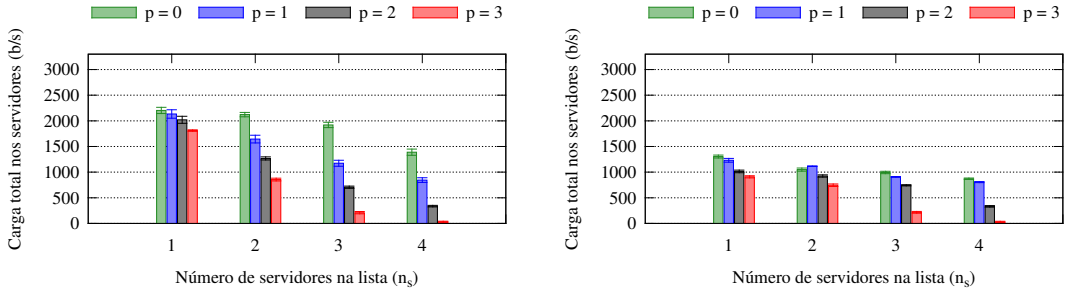
Esta seção apresenta os resultados dos experimentos utilizando o mecanismo de comutação de servidores proposto. A parte relativa à obtenção da lista de servidores não foi simulada, já que o foco é a comutação.

5.1. Desempenho do mecanismo de comutação de servidores

Os testes com o mecanismo de comutação de servidores foram realizados considerando um 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 feita em trabalho anterior [Bahia and Campista 2017] e foi empregada neste trabalho para fins de comparação de desempenho do mecanismo de comutação quando se considera servidores sem controle de demanda e com controle demanda. O mecanismo de controle de demanda proposto em [Bahia and Campista 2017] entra em modo de controle de demanda sempre que o servidor estiver operando em ciclo de trabalho maior que o limiar $Tx_{max} = 0,2\%$. Nesse modo, o servidor deixa de atender clientes sob demanda, enviando apenas notificações a clientes observadores. Após um intervalo de tempo $t_{espera} = 10\ s$, o servidor volta a verificar o ciclo de trabalho e assim o faz enquanto o nível de bateria estiver acima de um limiar $Q_{critico}$. Se o nível estiver abaixo de $Q_{critico}$, apenas os clientes sob demanda são atendidos e o processo não se repete mais. 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 ($t_{reiniciar}$). 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 em [Bahia and Campista 2017].

Carga nos servidores: Avaliando a carga total no conjunto de servidores (Figura 3), 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. Como observado, o aumento no número de observadores (p) reduz o fluxo de dados na entrada dos servidores, já que o cliente observador não gera 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.



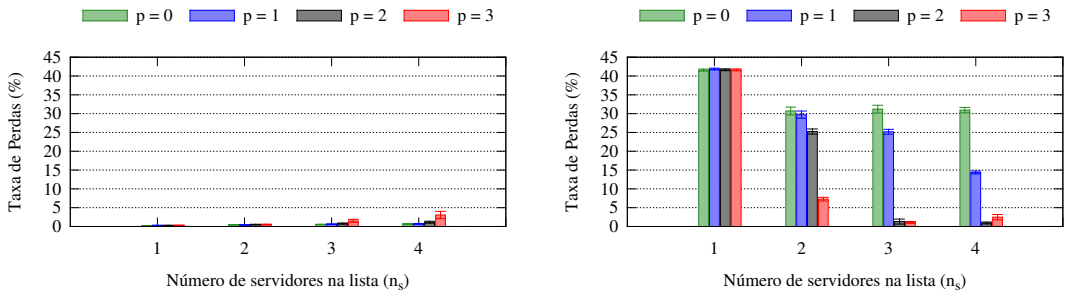
(a) Sem controle de demanda.

(b) Com controle de demanda.

Figura 3. Carga total nos servidores.

Taxa de perdas: A Figura 4(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 pois, nessa configuração, todos os clientes conseguem se tornar observadores (3 servidores com 3 observadores e 1 servidor com 1 observador), reduzindo a quantidade de pedidos gerados. Essa redução faz com que o impacto de uma perda seja maior no valor final da taxa de perdas. Já na Figura 4(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 compensa a indisponibilidade temporária de um servidor causada pelo mecanismo de controle de demanda.

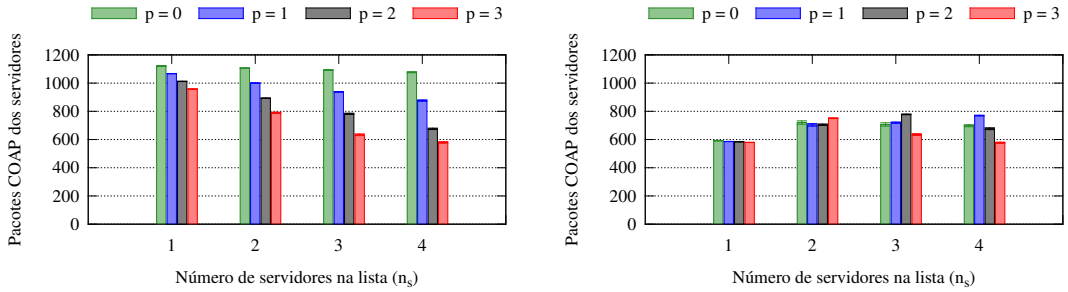
Número de pacotes CoAP: O aumento do número de observadores (p) reduz o número de pedidos sob demanda na rede. No caso considerado, o intervalo entre pedidos sob demanda ($t_n = 5\ s$) é menor que o intervalo entre notificações ($t_p = 10\ s$). Consequentemente, o aumento no número de observadores reduz também o número de pacotes enviados pelos servidores (Figura 5). 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(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



(a) Sem controle de demanda. (b) Com controle de demanda.

Figura 4. Taxa de perdas.

na lista e do número de clientes observadores reduz o impacto do uso do mecanismo de controle de demanda no atendimento aos clientes.



(a) Sem controle de demanda. (b) Com controle de demanda.

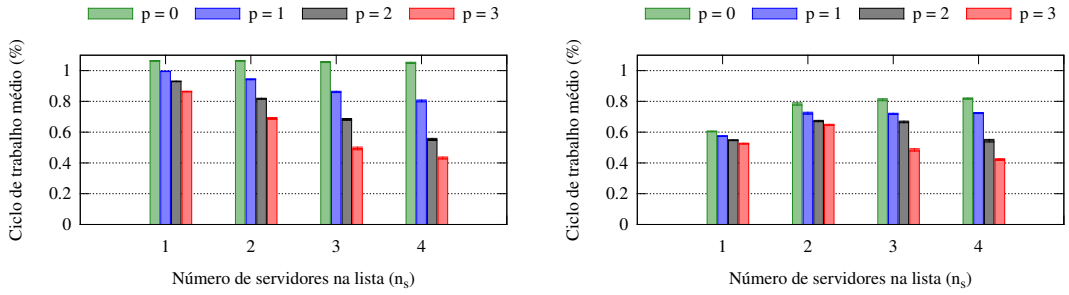
Figura 5. Total de pacotes CoAP dos servidores.

Consumo de energia: Com relação ao consumo médio de energia no conjunto de servidores, a Figura 6 mostra um comportamento semelhante ao observado na Figura 5. 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 observadores.

5.2. Balanceamento de Carga e de Consumo de Energia

Esta seção mostra a distribuição de carga e de consumo de energia entre os servidores diante da aplicação do mecanismo proposto.

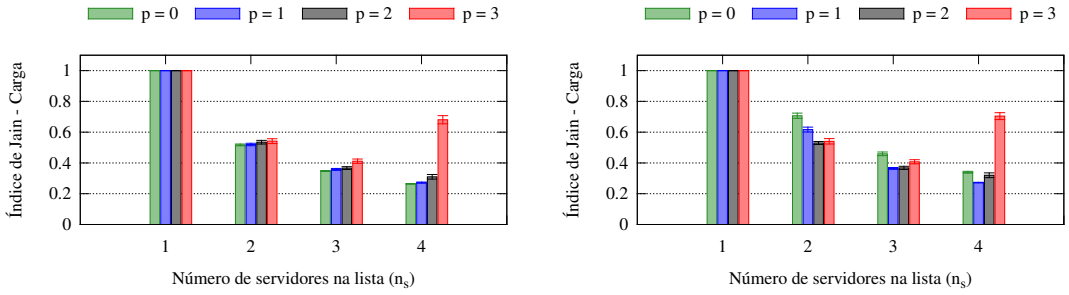
A lista de servidores de cada cliente possui uma ordem de prioridade. Como a lista de servidores e a configuração utilizada no mecanismo de comutação de servidores foi a mesma para todos os clientes, o primeiro servidor da lista foi o mais demandado. Através da comutação de servidores, é possível distribuir 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



(a) Sem controle de demanda. (b) Com controle de demanda.

Figura 6. Ciclo de trabalho médio do rádio.

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, calculado como $f(x_1, x_2, ..., x_n) = (\sum_{i=1}^{N} x_i)^2/(n \sum_{i=1}^{N} x_i^2)$. O índice de Jain pode assumir valores entre 0 e 1, sendo que quanto mais próximo de 1, melhor é o balanceamento entre os servidores, ou seja, melhor o índice de equidade.

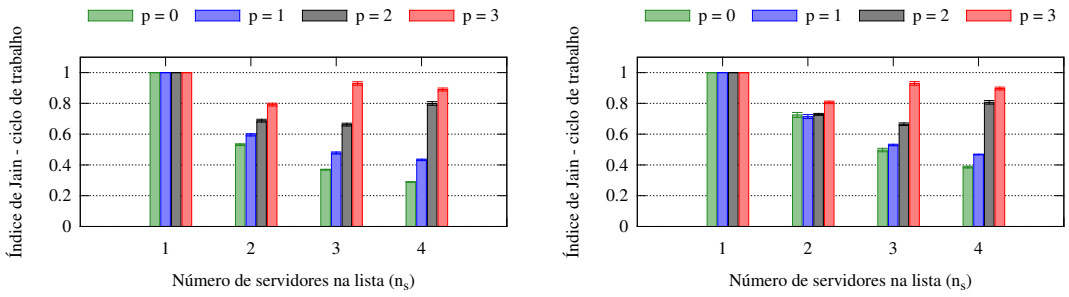


(a) Sem controle de demanda. (b) Com controle de demanda.

Figura 7. Índice de Jain da carga nos servidores.

Carga nos servidores: A Figura 7 mostra como o 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.

Consumo de energia nos servidores: Utilizando o índice de Jain para o consumo de energia, observa-se na Figura 8 que, quanto mais clientes se tornam observadores, me-



(a) Sem controle de demanda.

(b) Com controle de demanda.

Figura 8. Índice de Jain do ciclo de trabalho dos servidores.

lhor 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 observadores durante a ativação do controle, esses clientes podem garantir a obtenção do serviço comutando para outro servidor redundante. Os resultados mostram que o mecanismo proposto aumenta a confiabilidade dos serviços de IoT ao distribuir as demandas. Consequentemente, o consumo de energia torna-se balanceado entre os servidores, aumentando a disponibilidade dos serviços na rede.

6. Conclusões e Trabalhos Futuros

Este trabalho propõe um novo mecanismo de seleção e comutação de servidores para clientes CoAP. O mecanismo proposto melhora o desempenho e a escalabilidade dos servidores CoAP, distribuindo e aumentando a disponibilidade dos serviços. Até onde se sabe, este é o primeiro trabalho a propor um mecanismo que permita a obtenção de serviços no próprio cliente CoAP através da comutação de servidores. Os resultados mostraram que quanto maior a lista de servidores e o número de 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 pela indisponibilidade de um servidor 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 de servidores. Observa-se ainda que a funcionalidade de observação e a comutação de servidores promovem a redução, bem como o balanceamento do consumo energético.

Como trabalhos futuros, sugere-se implementar a proposta do mecanismo de obtenção da lista de servidores, ampliar o mecanismo proposto para interagir com *proxies* ou servidores auxiliares, visando o aumento na quantidade de observadores, e validar os resultados simulados com dispositivos reais.

Referências

Afzal, B., Alvi, S. A., Shah, G. A., and Mahmood, W. (2017). Energy efficient context aware traffic scheduling for iot applications. *Ad Hoc Networks*, 62:101 – 115.

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.

Bahia, J. G. and Campista, M. E. M. (2017). 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)*, pages 123–136, Belem, Brasil.

Banks, A. and Gupta, R. (2014). MQTT Version 3.1.1. *OASIS Standard*, pages 1–81.

Barbato, A., Barrano, M., Capone, A., and Figiani, N. (2013). Resource Oriented and Energy Efficient Routing Protocol for IPv6 Wireless Sensor Networks. In *Online Conference on Green Communications (GreenCom), 2013 IEEE*, pages 163–168, Piscataway, USA.

Cisco (2017). Cisco Visual Networking Index: Global Mobile Data Traffic Forecast. [Online: 18-December-2017].

FIWARE (2011). Context Broker (Orion). [Online: 20-January-2018].

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.

Iova, O., Theoleyre, F., and Noel, T. (2014). Improving the Network Lifetime with Energy-Balancing Routing: Application to RPL. In *Wireless and Mobile Networking Conference (WMNC), 2014 7th IFIP*, pages 1–8, Vilamoura, Portugal.

Jan, M., Nanda, P., He, X., Tan, Z., and Liu, R. P. (2014). 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*, pages 205–211, Beijing, China.

Jeyaraman, R. and Telfer, A. (2012). AMQP Version 1.0. *OASIS Standard*.

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, Barcelona, Spain.

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.

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

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