Um Protocolo IoT para Redução de Tráfego em Redes de Plano de Dados Programáveis

André L. R. Madureira[1], **Francisco R. C. Araújo**[1], **Leobino N. Sampaio**[1]

[1]Programa de Pós-Graduação em Ciência da Computação (PGCOMP)
Departamento de Ciência da Computação – Universidade Federal da Bahia (UFBA)
Salvador – BA – Brasil

{andre.romano, franciscorca, leobino}@ufba.br

Abstract. *IoT devices generate large continuous streams of data, causing congestion, which compromises the scalability of IoT networks. In order to solve these problems and facilitate interoperability between the various IoT devices, the Internet of Things Protocol (IoTP) was proposed in this study. Using IoTP, it's possible to implement data aggregation algorithms that adapt to the communication technologies (Ethernet, 802.11, etc.) used by IoT devices. In addition, such strategies can be implemented directly on the network's switch hardware. Emulation results show that IoTP brought a 78% improvement in network efficiency, as well as allowing control over the average delay generated by data aggregation techniques. IoTP has also been able to reduce the number of packets sent over the network, while also reducing the computational resource consumption of network switches.*

Resumo. *Dispositivos IoT geram grandes fluxos contínuos de dados, o que ocasiona congestionamentos, comprometendo a escalabilidade das redes IoT. A fim de solucionar esses problemas e facilitar a interoperabilidade entre os diversos dispositivos IoT, o Protocolo da Internet das Coisas (IoTP) foi proposto neste estudo. Ao utilizar o IoTP, é possível implementar algoritmos de agregação de dados que se adaptam às tecnologias de comunicação (Ethernet, 802.11, etc.) utilizadas pelos dispositivos IoT. Além disso, utilizando o IoTP, tais estratégias podem ser implementadas diretamente no hardware dos comutadores de pacotes. Os resultados de emulação mostram que o IoTP trouxe uma melhoria de 78% na eficiência da rede, além de permitir o controle sobre o atraso médio gerado pelas técnicas de agregação de dados. O IoTP também conseguiu reduzir o número de pacotes enviados pela rede, reduzindo o consumo de recursos computacionais dos comutadores de rede.*

1. Introdução

Em cenários de IoT (do inglês, *Internet of Things*), a maior parte do consumo de energia dos dispositivos ocorre na transmissão de pacotes [Rahman et al. 2016]. A agregação de pacotes em tais redes é, tradicionalmente, realizada dentro dos dispositivos, a fim de reduzir o número de comunicações com as estações rádio-base. A agregação de pacotes na rede é uma técnica que combina vários pacotes, possivelmente de diferentes origens, com requisitos e características heterogêneas. Essa combinação permite que um único pacote seja transmitido para o próximo nó da rede. Ao reduzir o número de pacotes na rede,

técnicas de agregação também conseguem reduzir o consumo de energia dos dispositivos, aumentar a eficiência da rede [Akyurek and Rosing 2018] e eliminar os cabeçalhos dos pacotes que são recorrentes. Além disso, a agregação de pacotes favorece a redução da latência e da variação do atraso (*jitter*) fim a fim [Akyurek and Rosing 2018].

Ao realizar agregação, dispositivos IoT, chamados de nós *sink*, requerem maior quantidade de recursos computacionais e apresentam um maior padrão de consumo energético que os demais nós da rede. Por tais motivos, estratégias de agregação de dados não se baseiam apenas no uso de nós *sink*. Na literatura é possível identificar propostas baseadas em estratégias hierárquicas (topologia em árvore) e em estratégias de *cluster* (agregação ocorre em conjuntos de dispositivos IoT interligados). A maioria dessas soluções pressupõe que o ambiente de rede seja homogêneo e que o fluxo de dados provenha de um único tipo de aplicativo [Akyurek and Rosing 2018]. Além disso, a maior parte dos estudos de agregação de dados que envolvem dispositivos IoT visa otimizar apenas uma única métrica de desempenho, como o consumo de energia [Akyurek and Rosing 2018].

A consolidação da programação de *hardware* de rede aliada ao aumento de desempenho de dispositivos que exercem funções de *gateways* em cenários de IoT [Yokotani et al. 2017], possibilita o desenvolvimento de estratégias de agregação mais eficientes. Neste trabalho, buscamos investigar como obter maior eficiência na agregação de dados IoT através de planos de dados programáveis. Para isso, propomos o protocolo IoTP (do inglês, *IoT Protocol*). Através do IoTP, é possível reduzir a sobrecarga resultante da repetição de cabeçalhos dos protocolos de rede e, assim, melhorar a eficiência da comunicação. Além disso, o mesmo proporciona um controle mais refinado sobre os atrasos associados à estratégia de agregação de dados. O IoTP foi projetado para ser facilmente embarcado em *hardwares* programáveis, tais como FPGAs, NPUs e ASICs. Assim, o mesmo pode ser utilizado por diferentes dispositivos IoT, servidores e *gateways* a fim de realizar agregações antes, durante e após o envio dos dados pelos dispositivos IoT.

O IoTP foi implementado através da linguagem P4 (do inglês, *Programming Protocol-Independent Packet Processors*) e sua avaliação foi realizada através de experimentos conduzidos por meio de emulação no ambiente do Mininet [Lantz et al. 2010]. O estudo consistiu em variar o número de dispositivos IoT conectados, a quantidade de dados agregados por pacote e a frequência de envio de pacotes IoTP com *Sync Flag*, comparando o desempenho da proposta com a estratégia de agregação IoT convencional (fim a fim). Os experimentos foram realizados a partir de um *trace* real extraído de um *data set* disponibilizado pelo Laboratório de pesquisas da Intel. Os resultados obtidos mostraram que o IoTP conseguiu melhorar a eficiência da rede em 78% e conseguiu controlar melhor os atrasos induzidos pela agregação de dados, atingindo um atraso médio de até 5 vezes mais rápido que a estratégia fim a fim. Além disso, considerando o cenário sem agregação de dados, o protocolo IoTP também obteve maior eficiência de rede que a pilha UDP/IP.

O restante do trabalho está organizado da seguinte forma: na Seção 2 são apresentados os trabalhos relacionados e na Seção 3 a proposta é detalhada. Os detalhes de implementação e os experimentos e resultados são discutidos nas Seções 4 e 5, respectivamente. Por fim, o trabalho é concluído na Seção 6.

2. Trabalhos Relacionados

Na literatura é possível encontrar trabalhos que propõem soluções de agregação de pacotes na rede. Em [Azevedo et al. 2011], os autores buscaram mitigar os problemas relacionados ao *jitter* da rede. O estudo de [Karlsson et al. 2009] analisou o efeito da agregação de dados no desempenho do protocolo TCP. Os resultados mostraram que a agregação aumenta o desempenho geral do TCP em até 73%. Além disso, a capacidade geral, a taxa de transferência e a eficiência do protocolo também são aprimoradas, enquanto o atraso de transmissão fim a fim é reduzido. Já em [Akyurek and Rosing 2018], os autores propuseram um algoritmo para agregação de pacotes, projetado para se adaptar às condições de fluxo de dados dos aplicativos e da própria rede. Essa nova abordagem superou os algoritmos comparados no estudo, proporcionando uma economia de energia de até 60%.

Outro conjunto de propostas trata do problema de agregação do ponto de vista dos dispositivos IoT. Em [Yokotani et al. 2017], os autores propuseram a agregação de dados em nível de pacote utilizando o padrão ZigBee, que permite enviar até 67 *bytes* de dados por pacote. Em [Zechinelli-Martini et al. 2011], foi proposta uma solução otimizada para uma aplicação específica, com restrições associadas a um determinado tipo de fluxo de dados.

Pesquisas mais recentes tentam integrar as diferentes técnicas de agregação em uma arquitetura IoT mais genérica [Karim and Al-kahtani 2016, Shen et al. 2017]. Na arquitetura PDDA proposta por [Karim and Al-kahtani 2016], a agregação de dados ocorre em três níveis (nos sensores, na estação base e na névoa) onde foram alcançados melhores resultados do que nas estratégias tradicionais (baseadas em árvore e em *cluster*). Já a proposta de [Shen et al. 2017] fornece uma arquitetura IoT de uso geral que facilita a interoperabilidade entre diferentes dispositivos IoT.

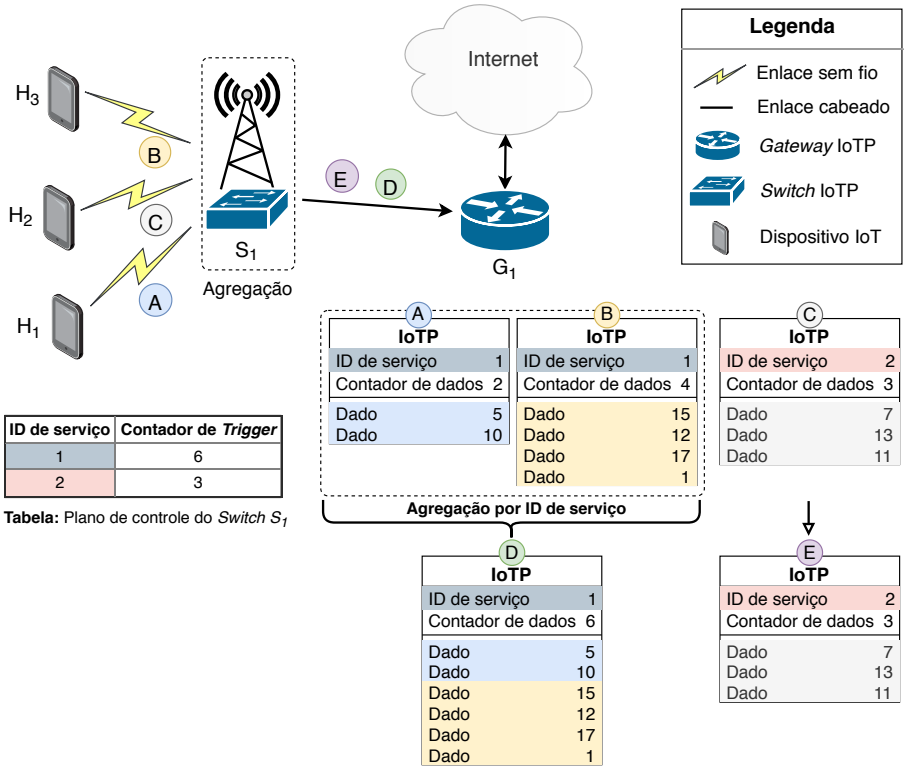
A maioria das soluções propostas acima pressupõe que o ambiente de rede seja homogêneo e que o fluxo de dados provenha de um único tipo de aplicativo. Além disso, de acordo com [Akyurek and Rosing 2018], a maior parte dos estudos de agregação de dados que envolve dispositivos IoT visa otimizar apenas uma única métrica de desempenho, como o consumo de energia. Vale ressaltar que, nos estudos citados, os algoritmos de agregação são projetados com foco nas redes de sensores sem fio (do inglês, *Wireless Sensor Networks* (WSN)). No entanto, outros tipos de rede que requerem soluções de agregação mais flexíveis podem ser utilizados no contexto IoT, como as redes celulares 5G. Simultaneamente, as aplicações IoT podem ter diferentes requisitos, características e métricas de otimização [Akyurek and Rosing 2018].

3. Protocolo IoTP

O IoTP tem como objetivo principal reduzir o *overhead* causado pela repetição dos cabeçalhos dos protocolos de rede. A partir da redefinição do processamento de quadros de enlace em camada 2 (*Link-Layer*), o protocolo aumenta a interoperabilidade entre dispositivos IoT heterogêneos e permite o desenvolvimento de estratégias mais eficientes de agregação. Tais vantagens são resultantes da execução em *hardware* de rede e, consequentemente, do processamento em velocidade de linha (*line rate*). As próximas subseções apresentam os detalhes do funcionamento do IoTP.

3.1. Agregação de dados em cenários de IoT

A Figura 1 ilustra uma das estratégias de agregação de dados suportada pelo IoTP. Nesse exemplo, o ID de serviço 1 possui o Contador de *Trigger* igual a 6. Já o ID de serviço 2 possui Contador de *Trigger* igual a 3. Os dispositivos IoT H_1, H_3 e H_2 enviam os respectivos pacotes A, B e C para o *switch* IoTP S_1. À medida que S_1 recebe esses pacotes, os dados contidos neles são agregados pelo *switch* nos novos pacotes D e E. Esses novos pacotes são então encaminhados para o *gateway* IoTP G_1.



ID de serviço	Contador de *Trigger*
1	6
2	3

Tabela: Plano de controle do *Switch* S_1

Figura 1. Agregação de dados no cenário de IoT através do IoTP.

O pacote D, do exemplo, é composto pelos dados contidos nos pacotes A e B, uma vez que ambos têm o mesmo ID de serviço 1. Esse pacote é criado pelo *switch* e enviado para o *gateway* IoTP. Isto ocorre, pois o *switch* recebeu uma quantidade de dados igual ou superior ao Contador de *Trigger* do ID de serviço 1. Ou seja, o *switch* recebeu pelo menos 6 blocos de dados IoTP marcados com o ID de serviço 1. Já o pacote E é composto apenas pelos dados contidos no pacote C, uma vez que nenhum outro pacote com o ID de serviço 2 foi recebido pelo *switch*. O *switch* cria e envia esse pacote para o *gateway* IoTP, uma vez que recebeu uma quantidade de dados igual ou superior ao Contador de *Trigger* do ID de serviço 2 (i.e., o *switch* recebeu 3 blocos de dados IoTP com o ID de serviço 2).

Tendo em vista que os cabeçalhos do IoTP fornecem informações acerca dos dados transmitidos, o IoTP pode também ser utilizado para implementação de outros algoritmos de agregação. Tais estratégias podem ser implementadas diretamente no *hardware* dos comutadores de rede, de forma a serem executadas pelo plano de dados. Assim, os algoritmos de agregação podem ser processados em velocidade de linha e dentro dos dispositivos de comutação. Dessa forma, o exemplo ilustrado na Figura 1 é apenas um dos exemplos de estratégias de agregação que podem ser implementadas utilizando o IoTP.

3.2. Pacotes IoTP

Para viabilizar o processamento em L2 (camada de enlace), o IoTP se baseia no processamento de pacotes composto por um cabeçalho fixo formado por cinco campos e uma pilha (*stack*) de blocos de dados, conforme ilustrado na Figura 2. Através deste cabeçalho, o IoTP torna-se genérico e independente de protocolos de roteamento e das tecnologias de comunicação subjacentes. Consequentemente, a interoperabilidade entre dispositivos de IoT heterogêneos torna-se mais factível.

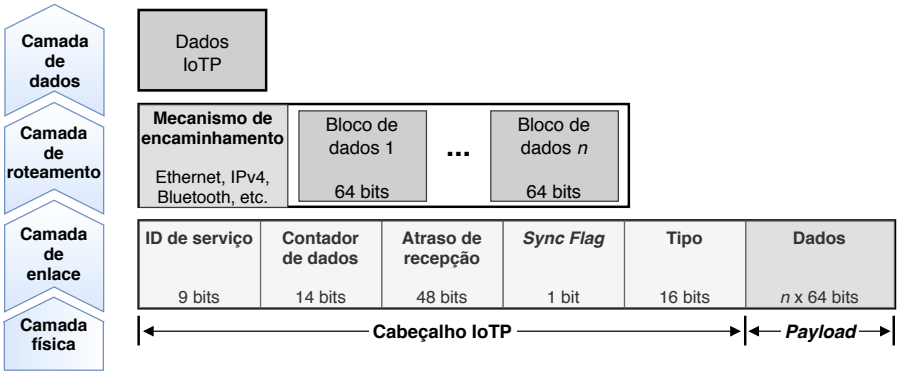


Figura 2. Especificação do Protocolo IoTP.

3.2.1. Cabeçalho dos pacotes

O cabeçalho do protocolo IoTP descrito na Figura 2 é formado pelos seguintes campos:

- **ID de serviço**: Esse campo de 9 *bits* identifica um conjunto de dados semelhantes, pertencentes à mesma aplicação IoT. Com essa abordagem, os dispositivos de rede podem agrupar os dados recebidos em um único pacote.
- **Contador de dados**: O IoTP permite que os dispositivos IoT enviem vários blocos de dados em um único pacote, desde que o limite previsto pela MTU da rede seja respeitado. Para fazer isso, o campo "Contador de dados" é utilizado para identificar a quantidade de blocos de dados que o pacote possui. Esse campo possui 14 *bits* e, portanto, o protocolo IoTP permite até 16383 blocos de dados por pacote, cada um contendo 8 *bytes*, totalizando 128 KB de dados.
- **Atraso de recepção**: Esse campo de 48 *bits* indica o tempo gasto desde a captura dos dados pelo sensor IoT até a recepção dessas informações pelo *gateway* IoTP. Assim, o comutador IoTP armazena os dados IoT internamente e, quando a condição para enviar o pacote é atendida, o comutador IoTP atualiza o atraso do pacote e envia os dados agregados. Essa abordagem permite que o *gateway* IoTP, ao receber os dados agregados, tenha conhecimento do tempo gasto desde a captura dos primeiros dados até a recepção dessas informações.
- ***Sync Flag***: Quando um pacote IoTP com o *Sync Flag* habilitado é recebido pelo comutador, o mesmo envia para o *gateway* IoTP todos os dados armazenados internamente que estão associados ao ID de serviço do pacote *Sync Flag*. Ao enviar periodicamente pacotes com *Sync Flag*, o *gateway* consegue controlar o *trade-off* entre o atraso de recepção dos dados e a eficiência da rede.

- **Tipo**: Esse campo de 16 *bits* é usado para informar à rede qual mecanismo de encaminhamento deve ser utilizado para encaminhar pacotes IoTP. Assim, o IoTP pode ser usado em qualquer rede, independentemente das tecnologias de comunicação e roteamento subjacentes.

3.2.2. Dados dos pacotes

Cada pacote do protocolo IoTP descrito na Figura 2 contém um conjunto finito de até n blocos de dados. Cada bloco transporta os dados coletados pelos dispositivos IoT, podendo conter até 64 *bits* (i.e., 8 *bytes*) de informação cada. Dessa forma, todos os elementos de uma rede IoTP estão cientes da quantidade e formato dos dados que cada pacote transporta, o que facilita a utilização das estratégias de agregação.

3.3. Agregação de dados através do IoTP

A fim de avaliar a efetividade do IoTP no suporte às técnicas de agregação, implementamos uma versão derivada do algoritmo de agregação *Accretion* [Kim et al. 2006]. Em nossa implementação do algoritmo, os dispositivos IoTP recebem diversos pacotes de outros dispositivos da rede e agregam seus dados de acordo com o ID de serviço dos pacotes. Ao final do processo de agregação, o dispositivo IoTP armazena essas informações temporariamente em sua memória interna (ver Algoritmo 1), até que essas informações sejam enviadas para o *gateway* IoTP (ver Algoritmo 2).

Conforme descrito pelo Algoritmo 1, a estratégia de agregação implementada dentro do *switch* armazena os dados enviados pelos dispositivos IoT, bem como os atrasos que estes dados sofreram desde o momento da captura até o envio para o *switch* IoTP. Além disso, o IoTP permite que os atrasos ocasionados pela estratégia de agregação sejam enviados para o *gateway*. Para isso, os *switches* também armazenam o momento exato em que o primeiro dado de um determinado ID de serviço foi recebido. Essa informação é utilizada para calcular quanto tempo os dados permaneceram dentro do *switch* e, em seguida, calcular o atraso total induzido pela estratégia de agregação escolhida. A partir dessa informação, é possível calcular uma estimativa do atraso acumulado desde a captura dos dados até a recepção dos mesmos pelo *gateway*. Ou seja, atraso total consiste na soma do atraso de captura com o atraso de agregação de pacotes.

Algoritmo 1: Armazenamento de dados.

```
1 Função ArmazenarDadosRecebidos(pkt):
2     se pkt.contadorDados > 0 então
3         ARMAZENARATRASO(pkt.id, pkt.atraso)
4         para cada dado em pkt.dados faça
5             ARMAZENARDADO(pkt.id, dado)
```

Após o armazenamento, as condições para envio dos dados são verificadas (ver Algoritmo 2). O comutador então calcula o atraso cumulativo da agregação dos dados, soma esta informação aos atrasos relatados pelos dispositivos IoT e envia os dados armazenados. O formato com o qual os dados armazenados serão enviados depende da estratégia de agregação implementada no *switch* IoTP. Se o *switch* implementar, por exemplo,

a estratégia média-móvel, o conjunto de n dados armazenados será comprimido em um único bloco de dados. Já se a estratégia de agregação por empilhamento for utilizada, como ilustrado na Figura 1, todo o conjunto de n dados será enviado em um mesmo pacote. Através do IoTP o comutador é responsável por controlar os atrasos relacionados à estratégia de agregação implementada e definir o momento no qual os dados armazenados serão enviados.

Algoritmo 2: Agregação e envio de dados.

```
1 Função EnviarDadosAgregados(pkt):
2     id = pkt.id
3     se DADOSARM(id) >= DADOSMIN(id) ou pkt.SF == 1 então
4         pkt.atraso = ATRASOSARMAZENADOS(id)
5         pkt.atraso += TEMPODADOSNOSWITCH(id)
6         pkt.contadorDados = QTDDADOSARMAZENADOS(id)
7         para cada dado em DADOSARMAZENADOS(id) faça
8             pkt.adicionarDado(dado)
9         ENVIARPACOTE(pkt)
```

A agregação considera duas condições para que ocorra o envio dos dados armazenados: (i) quantidade mínima de blocos de dados alcançada ou (ii) pacote IoTP recebido com *Sync Flag* habilitado. Na primeira, cada ID de serviço do protocolo IoTP está associado a uma quantidade mínima de dados, chamada de "Contador de *Trigger*", que indica para a estratégia de agregação quando enviar os dados armazenados dentro do *switch*. Quando a quantidade mínima de armazenamento é atingida, o *switch* aplica a estratégia de agregação sobre os dados armazenados localmente e envia os dados agregados para o *gateway* IoTP. A quantidade mínima de blocos de dados para cada ID de serviço é definida pelo controlador da rede SDN (do inglês, *Software-Defined Networks*), de acordo com as características dos dados recebidos e com os requisitos das aplicações IoT. Já a segunda condição permite que o IoTP controle os atrasos relacionados à estratégia de agregação. No entanto, ao contrário dos algoritmos *End-to-End*, *Hop-by-Hop* e *Accretion* [Kim et al. 2006], o IoTP não utiliza temporizadores (*Timers*) para realizar tal controle, em vez disso, envia pacotes com *Sync Flag* habilitado (i.e., *pkt.syncFlag == 1*) para o *switch* IoTP. À medida que tais pacotes são recebidos, o *switch* recupera todos os dados armazenados associados ao ID de serviço. Em seguida, a estratégia de agregação é aplicada sobre os dados e o agregado de dados resultante é enviado para o *gateway* IoTP. Dessa forma, o *gateway* IoTP da rede pode controlar a frequência de recepção de dados, ao enviar pacotes IoTP com *Sync Flag* periodicamente. Essa abordagem, portanto, permite controlar o *trade-off* entre atraso de recepção e eficiência da estratégia de agregação.

4. Implementação do IoTP

A linguagem P4 possibilitou o desenvolvimento do IoTP através de um plano de dados programável para o cenário de uma rede de sensores IoT WSN. Enquanto o ambiente de emulação Mininet/BMv2 foi utilizado na implementação e avaliação da proposta. As próximas subseções detalham essas ferramentas.

4.1. Linguagem P4

O P4 foi utilizado na implementação do IoTP por se tratar de uma linguagem de alto nível que permite a programação do plano de dados de redes SDN. Ao programar o plano de dados, foi possível implementar as características do protocolo proposto que tem como principal vantagem a execução no plano de dados e o processamento de pacotes em velocidade de linha. Ao utilizar a linguagem P4 foi possível descrever como os cabeçalhos dos pacotes são interpretados e quais ações podem ser executadas na chegada dos pacotes às interfaces de rede. Ademais, o controlador SDN da rede tem a função de preencher as tabelas *match-action* dos *switches* P4, informando as ações que podem ser executadas e quando executá-las. Com essas informações, o *switch* P4 analisa cada pacote IoTP recebido e executa uma ação específica.

4.2. Ambiente de emulação Mininet/BMv2

O ambiente de emulação de rede Mininet/BMv2 apresenta limitações que afetam a execução de programas P4. Uma dessas limitações é o uso da memória RAM e os gargalos de processamento. Ao implementar o IoTP usando o ambiente Mininet/BMv2, foi constatado um limite máximo de memória que o ambiente pode alocar. Os testes de conformidade do IoTP demonstraram que quando o limite de 200 KB dos registradores P4 era atingido, o ambiente parava de funcionar. Por tais motivos, a quantidade de dados que o IoTP poderia armazenar em um único pacote foi limitada a 50 blocos de dados.

A implementação de um *switch* IoTP necessitou de um maior número de tabelas *match-action* e registradores P4 do que a implementação de um *switch* Ethernet convencional. Conforme esses recursos foram sendo utilizados, notou-se uma degradação substancial do desempenho do ambiente de emulação P4 (Mininet/BMv2). Essa degradação provavelmente ocorreu devido ao fato do ambiente executar de modo *single-threaded*. É preciso destacar que conforme mais tabelas *match-action* e registradores P4 são utilizados, espera-se uma maior degradação no desempenho do *switch*.

5. Estudo Experimental e Análise dos Resultados

A fim de avaliar a efetividade do IoTP no suporte às estratégias de agregação, realizamos experimentos através dos quais foram avaliadas as estratégias de agregação *End-to-End* (E2E) e a baseada no IoTP. Os experimentos foram conduzidos utilizando um *trace* criado a partir do *data set* do Laboratório de pesquisas da Intel (*Intel Berkeley Research Lab*)[1].

5.1. Ambiente de Experimentação

Para criar e avaliar a solução proposta neste estudo, uma máquina virtual Ubuntu 16.04 LTS com kernel 4.4.0-141-generic x86_64 SMP foi configurada no VirtualBox 5.2.26 PUEL através da ferramenta vagrant e das instruções contidas no repositório P4 Tutorials[2]. Essa máquina foi executada sobre o sistema operacional hospedeiro Debian 9 e configurada com duas CPUs virtuais *Intel Core i7 4500U* com *clock* de 2 GHz e com 4 GB de RAM DDR3 de 1666 MHz.

No ambiente de emulação Mininet/BMv2 executado na máquina virtual, conectamos um conjunto de dispositivos IoT e um *gateway* IoTP ao mesmo *switch* de saída.

[1] `http://db.csail.mit.edu/labdata/labdata.html`.
[2] `https://github.com/p4lang/tutorials`.

Os dispositivos IoT foram conectados ao *switch* por meio da tecnologia de enlace sem fio IEEE 802.11n e o *gateway* IoTP foi conectado por um enlace cabeado IEEE 802.3ab (Gigabit Ethernet). Para configurar o ambiente de emulação e os enlaces supracitados, utilizamos os parâmetros de emulação descritos na Tabela 1. Todos os demais parâmetros de emulação não contidos nessa tabela foram adotados a partir dos padrões do Mininet.

Tabela 1. Parâmetros da Emulação.

Geral	Tempo de emulação em cada execução	10 segundos
	Número de execuções	20 replicações
Tecnologia dos enlaces	Dispositivos IoT	IEEE 802.11n
	Gateway IoTP	IEEE 802.3ab
Capacidade dos enlaces	Dispositivos IoT	230 Mbps
	Gateway IoTP	1000 Mbps
Atraso de propagação	Dispositivos IoT	2 ms
	Gateway IoTP	0 ms

O *trace* utilizado contém dados gerados por 54 dispositivos IoT reais, relacionados à temperatura do ambiente, umidade, luz e tensão da bateria de cada dispositivo. Cada tipo de dados (temperatura, umidade, luz e tensão) foi associado aos IDs de serviço do IoTP 0, 1, 2 e 3, respectivamente. Em seguida, cada *host* Mininet foi configurado para ler os dados do *trace*, associados a um determinado dispositivo IoT. Assim, cada *host* conseguiu replicar as transmissões de dados de um determinado dispositivo IoT, preservando os registros de data e hora da captura dos dados.

5.2. Metodologia de Avaliação

Os experimentos tiveram como objetivo comparar os resultados com o algoritmo de agregação *End-to-End* [Kim et al. 2006], que executa dentro dos dispositivos IoT e em uma rede baseada em *switches* L2 com a pilha UDP/IP. Esse algoritmo armazena os dados capturados pelos sensores IoT dentro de cada dispositivo. Em seguida, tal dispositivo realiza a agregação dos dados e, quando uma certa quantidade de dados é agregada, os dados agregados são enviados para o *gateway* IoTP. Portanto, as principais diferenças entre o algoritmo *End-to-End* e o implementado dentro dos *switches* IoTP são a quantidade de dados disponíveis para agregação, a pilha de protocolos utilizada (IoTP versus UDP/IP) e o local em que a agregação ocorre (na rede ou nos dispositivos IoT).

Organizamos a avaliação do IoTP em duas etapas. Primeiro, aplicamos o projeto fatorial completo utilizando k fatores e dois níveis para cada fator (2^k combinações possíveis), conforme sugerido por [Jain 1990]. Em seguida, consideramos os fatores mais eficazes para nossa análise de desempenho. Na Tabela 2, descrevemos os fatores escolhidos e seus níveis. Os fatores foram escolhidos de acordo com as recomendações do estado da arte. Somente os níveis mínimo e máximo foram considerados para a fase de planejamento fatorial. Para fins estatísticos, replicamos o experimento 20 vezes, extraímos a média e o desvio padrão e calculamos o nível de confiança de 95%.

Ao analisar os efeitos dos fatores sobre a métrica atraso médio, percebemos que todos os fatores (A, B e C) têm um forte impacto sobre essa métrica e que eles também interagem entre si. Portanto, a combinação de todos os fatores e níveis foi levada em consideração durante a fase de análise de desempenho. Além disso, adicionamos ao fator

Tabela 2. Projeto Fatorial Completo.

	Fatores	Mín - Máx	Unidade
A	Agregação de dados	10 - 50	blocos de dados
B	Número de dispositivos IoT	10 - 50	dispositivos
C	Atraso máximo para envio de pacotes	0 - 100	ms

B os níveis 20, 30 e 40 a fim de analisar o impacto das estratégias de agregação de acordo com o número de dispositivos IoT da rede. A partir dos fatores e níveis mencionados, as seguintes métricas de desempenho foram utilizadas para avaliar a solução proposta:

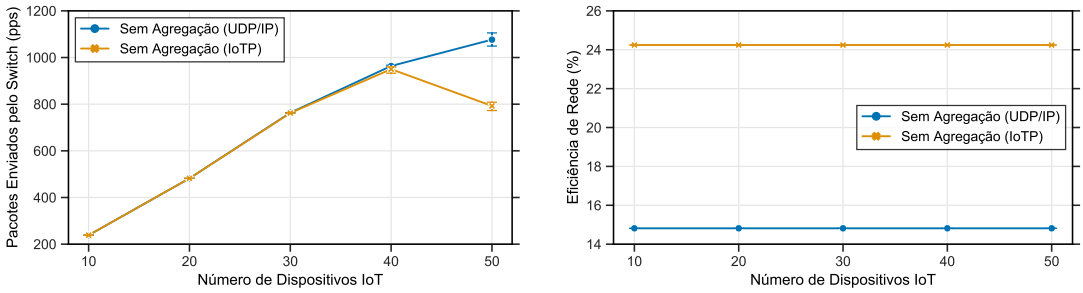
- **Total de dados enviados (Kbps)**: Total de dados contidos nos pacotes enviados para o *gateway*, incluindo os cabeçalhos e a carga útil de cada pacote;
- **Total de carga útil (Kbps)**: Total de carga útil contida em cada pacote enviado para o *gateway*, excluindo os cabeçalhos dos protocolos;
- **Eficiência da rede (%)**: Relação entre o total de carga útil e o total de dados enviados, que serve como um indicador de eficiência na transmissão de dados na rede, conforme descrito por [Zechinelli-Martini et al. 2011];
- **Total de pacotes enviados pelo *switch* (pps)**: Número total de pacotes enviados pelo *switch* para o *gateway* durante cada execução da emulação;
- **Atraso médio (ms)**: Intervalo de tempo entre a captura do dado, pelo sensor IoT, e a sua recepção pelo *gateway*.

5.3. Teste de Conformidade

A fim de avaliar o desempenho do ambiente de emulação, realizamos testes utilizando o protocolo IoTP e o UDP/IP sem agregação. Dessa forma, obtemos *baselines* para distinguir o impacto do desempenho do ambiente de emulação do impacto dos mecanismos de agregação avaliados.

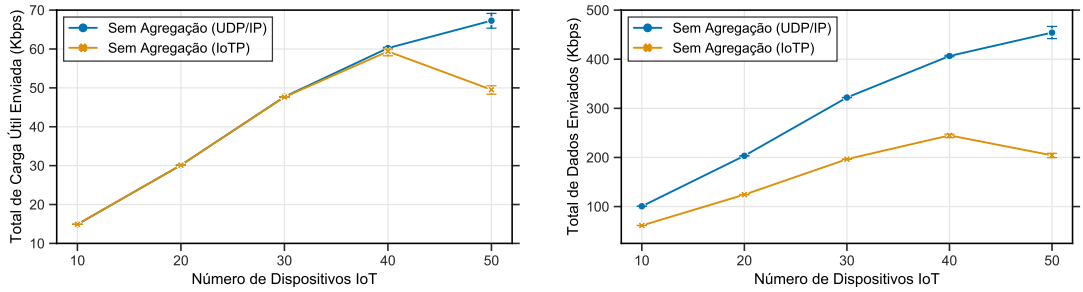
Ao compararmos as estratégias sem agregação de dados percebemos que, quando o número de dispositivos IoT aumenta, o IoTP envia menos pacotes para o *gateway* que a estratégia UDP/IP, o que indica um gargalo de processamento no *switch* IoTP (ver Figura 3(a)). De acordo com [Dang et al. 2017] e [Zhang et al. 2018], um *switch* P4 tem sua velocidade de processamento reduzida conforme o número de tabelas *match-action* cresce. Para que a estratégia de agregação no *switch* IoTP seja executada adequadamente, o *switch* precisa verificar suas tabelas *match-action* e seus registradores sempre que um pacote chega em suas interfaces de rede. A partir dessa verificação, o *switch* decide se há a execução da estratégia de agregação e de que forma a mesma acontece. Portanto, o gargalo de processamento presente no *switch* IoTP é decorrente da utilização de um maior número de tabelas *match-action* e registradores P4, quando comparado a um *switch* Ethernet convencional. No entanto, o protocolo IoTP possui uma eficiência de rede superior ao UDP/IP, para o cenário sem agregação de dados (ver Figura 3(b)).

Em relação à métrica total de carga útil, devido aos gargalos de processamento do protocolo IoTP supracitados, o total de carga útil que ele consegue enviar também é reduzido quando comparado ao UDP/IP no cenário sem agregação de dados (ver Figura 4(a)). Este mesmo padrão está presente também no total de dados enviados, como mostra a Figura 4(b). Contudo, mesmo com as restrições supracitadas, o IoTP possui maior eficiência de rede, como visto na Figura 3(b). Isto indica que o IoTP consegue enviar um volume maior de carga útil para uma determinada quantidade de dados enviados.



(a) Pacotes enviados pelo *switch*. (b) Eficiência da rede.

Figura 3. Pacotes enviados e eficiência de rede no cenário sem agregação de dados.

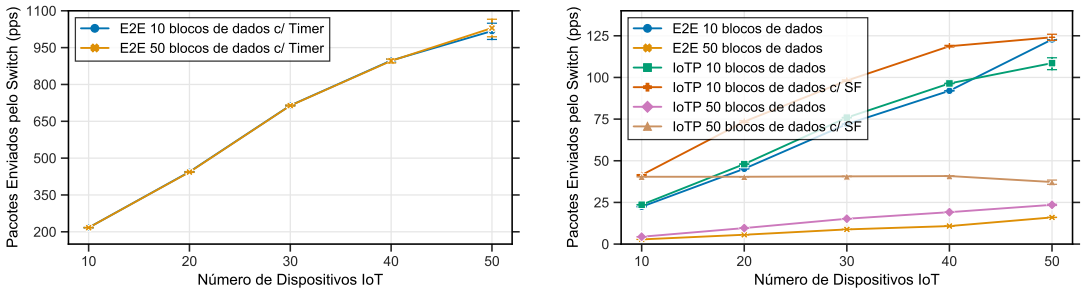


(a) Total de carga útil enviada. (b) Total de dados enviados.

Figura 4. Total de dados no cenário sem agregação de dados.

5.4. Resultados e Discussões

Para simplificar a notação, nesta seção utilizaremos o termo IoTP para nos referirmos ao algoritmo de agregação implementado no *switch* IoTP. Ao avaliarmos os resultados, observamos que, na comparação do IoTP sem envio periódico de pacotes *Sync Flag* (SF) com o IoTP com envio periódico de pacotes *Sync Flag*, há um aumento no número de pacotes recebidos pelo *gateway IoTP* quando os pacotes *Sync Flag* são enviados (ver Figura 5).

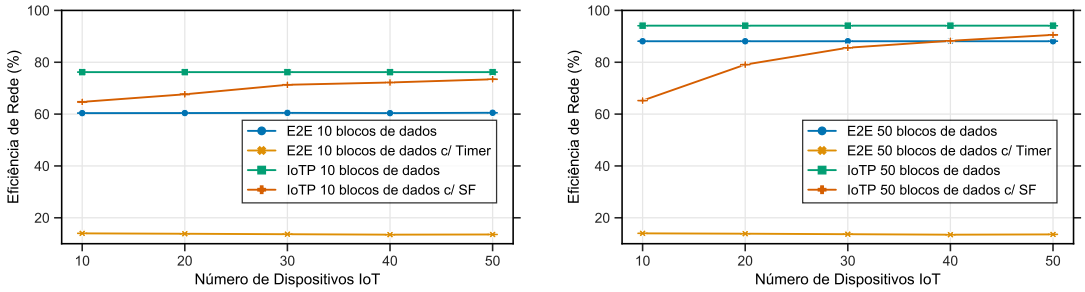


(a) Estratégia *End-to-End* com *Timer*. (b) Outras estratégias de agregação.

Figura 5. Pacotes enviados pelo *switch* no cenário com agregação de dados.

Do ponto de vista estatístico, o *End-to-End* (E2E) e o IoTP tiveram uma redução

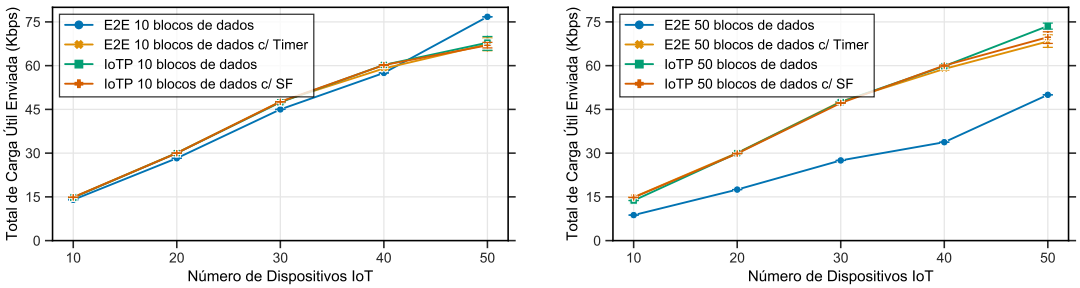
similar no número total de pacotes enviados pela rede, sendo o *End-to-End* ligeiramente melhor que o IoTP. Como resultado dessa redução no número de pacotes, espera-se um aumento na eficiência da rede [Akyurek and Rosing 2018]. Pudemos verificar que, de fato, a eficiência da rede foi aprimorada usando o IoTP, tendo em vista que o IoTP conseguiu obter métricas de eficiência de rede melhores do que o *End-to-End*. Também verificamos que a eficiência de rede do IoTP diminui quando ele está enviando pacotes com *Sync Flag* periodicamente (ver Figura 6). Isto se deve ao fato de, ao utilizar *Timers* (*End-to-End*) ou *Sync Flags* (IoTP), o atraso médio de agregação recebe maior prioridade do que a eficiência de rede.



(a) Até 10 blocos de dados agregados. (b) Até 50 blocos de dados agregados.

Figura 6. Eficiência da rede no cenário com agregação de dados.

Com relação ao total de carga útil enviado, verificamos uma diferença significativa entre a estratégia *End-to-End* com 50 blocos de dados quando comparada às demais (ver Figura 7). Isso se deve ao fato de não haver tempo suficiente para os dispositivos IoT acumularem 50 blocos de dados para enviar um pacote com o agregado de dados. Assim, os dados agregados que não atingirem os 50 blocos não são enviados pelos dispositivos IoT, o que resulta em perda de carga útil. Tal efeito não foi constatado no IoTP, posto que o mesmo opera com os dados dos dispositivos IoT como um todo, realizando a agregação dentro da rede. Dessa forma, há mais oportunidades para o IoTP agregar dados e atingir a quantidade de blocos de dados configurada.



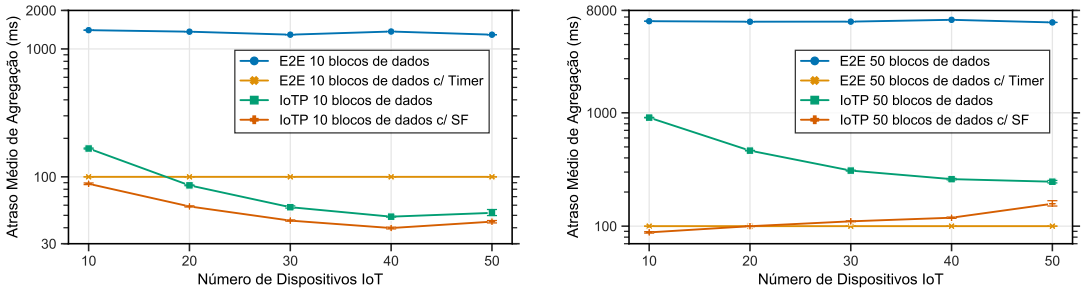
(a) Até 10 blocos de dados agregados. (b) Até 50 blocos de dados agregados.

Figura 7. Total de carga útil enviada no cenário com agregação de dados.

Ainda com relação ao total de carga útil enviado, o IoTP consegue enviar uma quantidade de carga útil similar às demais estratégias de agregação (ver Figura 7). No entanto, o IoTP possui maior eficiência de rede quando comparado ao *End-to-End*. Isso

indica que o IoTP utiliza menos os enlaces de rede, sendo, portanto, mais eficiente do que o *End-to-End*, conforme já observado na Figura 6.

O IoTP, quando realiza agregação de até 10 blocos de dados, apresenta um atraso médio mais baixo quando comparado ao *End-to-End* (ver Figura 8). No entanto, quando o IoTP opera com 50 blocos de dados agregados, o *End-to-End* com *Timer* consegue atingir menor atraso médio. Acreditamos que essa diferença entre o IoTP e o *End-to-End* em termos de atraso médio deriva do fato do IoTP agregar um volume maior de dados, posto que a agregação do IoTP ocorre dentro dos *switches* da rede e opera com todos os dados dos dispositivos IoT. Além disso, ao contrário do *End-to-End*, o IoTP não opera com *Timers*. O IoTP é um protocolo reativo que é capaz de controlar o atraso médio através do recebimento de pacotes *Sync Flag* pelos *switches* IoTP. Assim, o controle dos atrasos médios do IoTP depende da periodicidade do envio dos pacotes *Sync Flag* e de características da rede, como atraso de propagação e capacidade de transmissão dos enlaces.



(a) Até 10 blocos de dados agregados.

(b) Até 50 blocos de dados agregados.

Figura 8. Atraso médio no cenário com agregação de dados.

6. Conclusão e Trabalhos Futuros

O IoTP é um protocolo de baixo *overhead* para redes de sensores IoT que pode melhorar a eficiência da rede, controlar o atraso médio induzido pela agregação de dados e reduzir a sobrecarga que os cabeçalhos de protocolo repetidos ocasionam na rede. Além disso, o IoTP também é capaz de reduzir a quantidade de pacotes que circulam pela rede, o que deve resultar em um menor consumo de recursos computacionais dos comutadores de rede, como CPU e memória, e consequente melhora no desempenho da rede. Nossos resultados mostram que o IoTP tem uma eficiência de rede 78% melhor e é 5 vezes mais rápido em termos de atraso médio quando comparado à estratégia de agregação de dados *End-to-End*. Considerando o cenário sem agregação de dados, o protocolo IoTP também obteve melhor eficiência de rede que o UDP/IP.

Como sugestão de trabalhos futuros, pretende-se utilizar o IoTP dentro dos dispositivos IoT, associado à estratégia de agregação *End-to-End*. Dessa forma, pretende-se analisar como essa integração interfere na autonomia energética dos dispositivos IoT, na eficiência da rede e nos atrasos médios.

Agradecimentos

Os autores agradecem o apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) e da Fundação de Amparo à Pesquisa do Estado da Bahia (FAPESB).

Referências

Akyurek, A. S. and Rosing, T. S. (2018). Optimal packet aggregation scheduling in wireless networks. *IEEE Transactions on Mobile Computing*, 17(12):2835–2852.

Azevedo, P. H., Caetano, M. F., and Bordim, J. L. (2011). A packet aggregation mechanism for real time applications over wireless networks. In *2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum*, pages 648–655.

Dang, H. T., Wang, H., Jepsen, T., Brebner, G., Kim, C., Rexford, J., Soulé, R., and Weatherspoon, H. (2017). Whippersnapper: A p4 language benchmark suite. In *Proceedings of the Symposium on SDN Research*, SOSR '17, pages 95–101, New York, NY, USA. ACM.

Jain, R. (1990). *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. John Wiley & Sons.

Karim, L. and Al-kahtani, M. S. (2016). Sensor data aggregation in a multi-layer big data framework. In *2016 IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pages 1–7.

Karlsson, J., Kassler, A., and Brunstrom, A. (2009). Impact of packet aggregation on tcp performance in wireless mesh networks. In *2009 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks Workshops*, pages 1–7.

Kim, K., Ganguly, S., Izmailov, R., and Hong, S. (2006). On packet aggregation mechanisms for improving voip quality in mesh networks. In *2006 IEEE 63rd Vehicular Technology Conference*, volume 2, pages 891–895.

Lantz, B., Heller, B., and McKeown, N. (2010). A network in a laptop: Rapid prototyping for software-defined networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, Hotnets-IX, pages 19:1–19:6, New York, NY, USA. ACM.

Rahman, H., Ahmed, N., and Hussain, I. (2016). Comparison of data aggregation techniques in internet of things (iot). In *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, pages 1296–1300.

Shen, Y., Zhang, T., Wang, Y., Wang, H., and Jiang, X. (2017). Microthings: A generic iot architecture for flexible data aggregation and scalable service cooperation. *IEEE Communications Magazine*, 55(9):86–93.

Yokotani, T., Shimuzu, A., Sasaki, Y., and Mukai, H. (2017). Proposals for packet processing and performance evaluation of iot devices. In *2017 Japan-Africa Conference on Electronics, Communications and Computers (JAC-ECC)*, pages 5–8.

Zechinelli-Martini, J. L., Bucciol, P., and Vargas-Solar, G. (2011). Energy aware data aggregation in wireless sensor networks. In *2011 2nd International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace Electronic Systems Technology (Wireless VITAE)*, pages 1–5.

Zhang, C., Bi, J., Zhou, Y., Zhang, K., and Ma, Z. (2018). B-cache: A behavior-level caching framework for the programmable data plane. In *2018 IEEE Symposium on Computers and Communications (ISCC)*, pages 00084–00090.