Um mecanismo para redundância de dados entre *Brokers* MQTT com uso de *Distributed Hash Table*

Lucas Vargas Dias[1], **André Rech Eichner**[1] **e Tiago A. Rizzetti**[1]

[1] Curso Superior de Tecnologia em Redes De Computadores
Universidade Federal de Santa Maria (UFSM)
Av. Roraima nº 1000. Cidade Universitária. Camobi. Santa Maria - RS

{lucas_dias,eichner,rizzetti}@redes.ufsm.br

Abstract. *The Message Queuing Telemetry Transport (MQTT) is a delivery message protocol to devices in the Internet of Things (IoT) concept. It uses the Publisher/Subscriber paradigm. The clients exchange information through the broker, then, creating a single point of failure. Some works in literature propose the use of clusters with load balancers. However, data availability is not treated. This paper presents the implementation of a mechanism for synchronism and data redundancy between brokers using a Distributed Hash Table network.*

Resumo. *O Message Queuing Telemetry Transport (MQTT) é um protocolo de entrega de mensagens para dispositivos no conceito de Internet das Coisas (IoT). Ele usa do paradigma Publisher/Subscriber. Os clientes trocam mensagens através de um broker, assim, criando um ponto único de falha. Algumas soluções apresentadas na literatura propõem um cluster para balanceamento de carga como mecanismo de defesa. Entretanto, a disponibilidade dos dados não é tratada. Este trabalho apresenta a implementação de um mecanismo para sincronismo e redundância de dados entre brokers MQTT com uso da rede Distributed Hash Table (DHT).*

1. Introdução

A interação entre o mundo físico e digital mostra-se inevitável, no qual diversos fatores vem incentivando esta integração. A comunicação nos dispositivos atuadores e de sensoriamento é realizada através de redes de comunicação. Todos os dados coletados por estes dispositivos são armazenados e processados de maneira "inteligente". No caso de atuadores, o processamento deve resultar na modificação dos parâmetros no ambiente físico [Sethi and Sarangi 2017]. Tudo isso tornou-se comumente conhecido como Internet das Coisas ou *Internet of Things* (IoT).

Devido a limitações computacionais nos sensores e atuadores, novos protocolos de comunicação foram surgindo. Dentre eles, destaca-se o *Message Queuing Telemetry Transport* (MQTT). Ele possui características como a pouca largura de banda necessária e o baixo processamento [Kashyap et al. 2018]. Ele utiliza um paradigma de comunicação *Publisher/Subscriber* em que um *broker* repassa as mensagens aos clientes inscritos em tópicos. Isso remove a necessidade de comunicação direta entre os clientes. Entretanto, caso o *broker* seja comprometido ou venha a falhar, a rede simplesmente para de funcionar e não há mais comunicação entre os dispositivos [Quincozes et al. 2019]. Isso

pode acarretar em problemas como a perda de dados de sensoriamento em ambientes de refrigeração.

Dito isto, este trabalho visa implementar um serviço de redundância e sincronização das informações entre os *broker*s, a fim de aumentar a disponibilidade dos dados na rede MQTT. Isso será realizado através de redes de sobreposição ponto-a-ponto (P2P) baseadas em *Distributed Hash Table* (DHT). A partir daqui, o trabalho é divido da seguinte forma: A Seção 2 discorre sobre os trabalhos relacionados. A Seção 3 apresenta o funcionamento do protocolo MQTT e da rede DHT. A arquitetura proposta por este trabalho encontra-se na Seção 4. A Seção 5 mostra os resultados obtidos, e junto abre espaço para uma discussão sobre o assunto. Finalizando o trabalho, a Seção 6 aborda os trabalhos futuros e as considerações finais.

2. Trabalhos relacionados

Em [Jutadhamakorn et al. 2017], os autores propuseram a utilização do conceito de *Clusterização* juntamente com uso de *containers* nos *brokers* MQTT. A proposta utiliza um balanceador de carga entre os clientes e os *brokers*. Ao receber uma nova solicitação de conexão, o balanceador de carga contata o *broker* e verifica sua disponibilidade. Caso o balanceador de carga obtenha uma resposta não satisfatória, a requisição é emitida a outro, até encontrar um *broker* disponível para tratar a mensagem. O trabalho teve como resultado a diminuição da sobrecarga causada pelas requisições dos clientes. No entanto os autores não tratam da redundância das mensagens. Dessa forma, a disponibilidade dos dados não é garantida.

Em [Sen and Balasubramanian 2018], os autores também propõem um balanceador de carga, mas em conjunto com uma *cache* compartilhada. Ela é utilizada para tornar acessíveis as informações recebidas pelos *broker*s. Esta proposta também é promissora, e traz uma redundância dos dados. Entretanto, as consultas realizadas na *cache* podem causar um *delay* na rede. Um diferencial da arquitetura proposta e descrita na Seção 4 é a redundância e sincronismo de dados no momento que uma informação é publicada no *broker* MQTT.

3. Descrição de funcionamento do Protocolo MQTT e da rede DHT

O MQTT é um protocolo de entrega de mensagem do tipo *Publisher/Subscriber*. Os clientes podem se inscrever ou publicar informações em tópicos. Quando uma nova mensagem é emitida a um determinado tópico, os clientes inscritos nele recebem a mensagem através do *broker* MQTT [Niruntasukrat et al. 2016].

A Figura 1 mostra um exemplo de implementação deste protocolo. Ela é composta por três dispositivos em modo cliente, e um *broker* MQTT central. Um dos dispositivos conectado na rede MQTT é uma *smart TV*, com IoT integrado. Esse cliente está inscrito no tópico ”Canal TV”e atua como *Subscriber* na rede. Outro dispositivo é um refrigerador inteligente, que atua na rede MQTT como *Publisher*. Sua principal função é divulgar sua temperatura interna periodicamente. O terceiro dispositivo é um *smartphone*. Esse, atua como cliente *Publisher* e *Subscriber* simultaneamente [Soni and Makwana 2017]. O dispositivo publica informações para alteração de canal da *smart TV*, no tópico ”Canal TV”, e verifica periodicamente a temperatura interna do refrigerador.

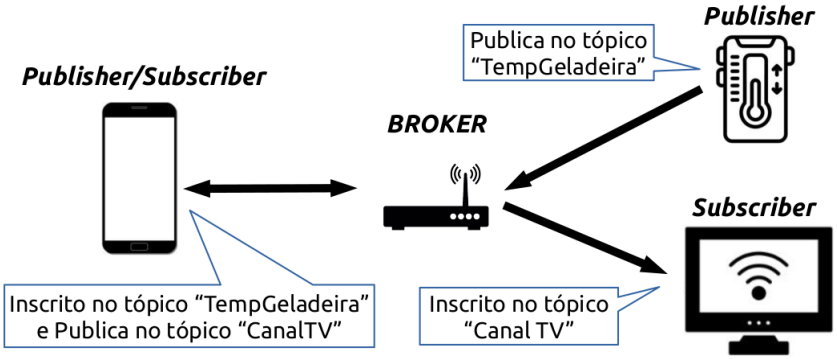

Figura 1. Exemplo de aplicação com arquitetura MQTT

No entanto todos os dispositivos ficam dependentes do pleno funcionamento do serviço do *broker* MQTT. Dessa forma, é criado um ponto de falha na aplicação MQTT [Kotak et al. 2019]. Um agente malicioso pode personificar um cliente e emitir pacotes a uma frequência maior que a capacidade de processamento do *broker* ou que o enlace de rede dele suporte. Desta forma a rede toda fica instável, causando um ataque de negação de serviço [Firdous et al. 2017]. Para contornar o problema, este trabalho utiliza da rede DHT para sincronismo e redundância de dados entre diferentes *brokers*. Optou-se pelo uso rede DHT devido a características como tolerância a falhas e escalabilidade [Rahimi et al. 2016].

Na DHT, os dados estão vinculados a uma chave, que é um *hash* criptográfico. Além disso, cada cliente também é identificado por um *hash*. Ao alocar um novo recurso na rede, a informação fica armazenada junto ao dispositivo com identificador mais próximo da chave resultante. Para recuperar uma informação, um nó deve fazer a busca através de uma *Infohash*, no qual é a ”chave”do valor que retorna o valor do dado consultado. Uma vez que a informação tenha sido recuperada, o dispositivo passa a fornecê-lá. Com isso, qualquer nó desta rede pode alocar ou recuperar valores, tornando a rede DHT vulnerável [Srinivasan and Aldharrab 2019]. Este trabalho utiliza uma Infraestrutura de Chave Pública (ICP) para autenticação e revogação de nós. A validade dos participantes é em tempo de execução. Dessa forma, apenas nós legítimos alocam recursos ou ingressam na DHT [Garcia et al. 2020].

4. Arquitetura Proposta

Um *broker* MQTT tem o funcionamento de um servidor centralizado, e pode ser explorado por um atacante causando um ataque de negação de serviço. Para contornar esse problema, a arquitetura proposta é apresentada na Figura 2.

A arquitetura é composta por duas entidades. O *broker* MQTT, que permite o ingresso dos clientes na rede e um módulo de sincronismo. O primeiro, já teve seu funcionamento descrito anteriormente e o segundo é um cliente MQTT. Esse, pode estar localizado junto ao *broker* ou qualquer outro local. No segundo caso, é interessante realizar uma autenticação mútua com uso do *Transport Layer Security* (TLS), entretanto, há um atraso maior de entrega devido ao tempo de propagação e operações criptográficas. O fluxo de funcionamento do módulo de sincronismo é apresentado na Figura 3.

Inicialmente o dispositivo ingressa na rede DHT contatando um dos participantes.

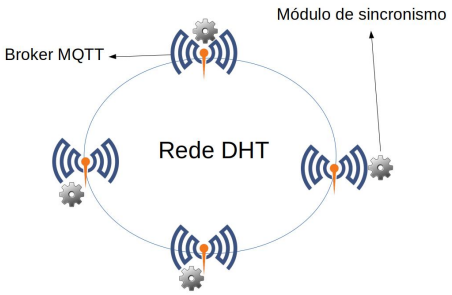

Figura 2. Visão geral dos componentes da arquitetura proposta

No caso de ele ser o primeiro nó, este é denominado nó de *bootstrap*. Caso contrário, o nó ingressante e o nó de *bootstrap* utilizam o mecanismo de autenticação proposto em [Garcia et al. 2020]. Primeiramente é estabelecido um canal de comunicação seguro (*Diffie-Hellman*), em que a chave resultante é usada para identificar a sessão da comunicação, e assim evitando ataques de replicação [Garcia et al. 2020]. Na sequência são trocados os certificados digitais de ambas entidades, para que seja realizada a autenticação mútua. No caso de sucesso, o módulo de sincronismo conecta-se ao *broker* MQTT.

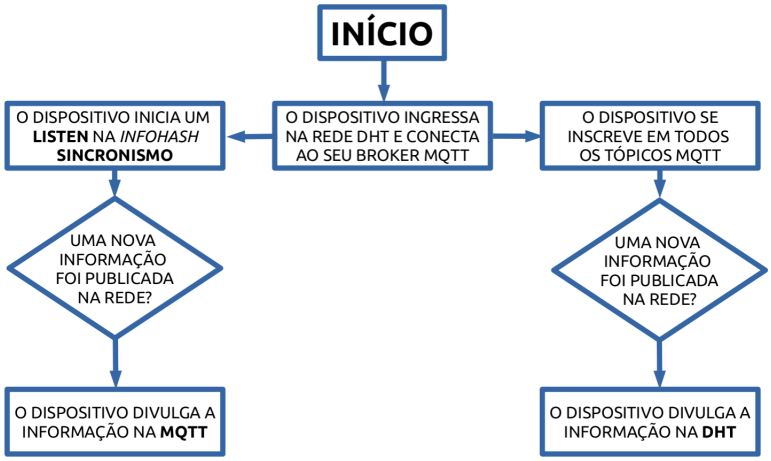

Figura 3. Fluxograma de funcionamento do módulo de sincronismo

Cada vez que o *broker* receber um novo tópico, ele repassa para o módulo de sincronismo. Esse, monta um pacote com seu certificado digital, tópico, mensagem recebida e assinatura digital desses dados, de forma concatenada. Então esse pacote é publicado na rede DHT. Quando uma nova informação é recebida na rede DHT, o módulo de sincronismo verifica o certificado digital do emissor e na sequência a assinatura digital da mensagem. Se forem válidos, os clientes publicam no seu respectivo *broker*. Para evitar que a mesma informação seja recebida mais de uma vez na rede DHT, um sequenciador é utilizado. Uma vantagem deste trabalho é que a proposta é genérica e pode ser aplicada a qualquer tipo de *broker* MQTT. Não há necessidade de modificação de código-fonte dos *broker*s utilizados. A seguir, na Seção 5 são apresentados os testes e resultados alcançados.

5. Resultados e Discussões

Para validar o trabalho proposto, foi utilizado o *Common Open Research Emulator* (CORE), que permite a criação de uma rede de simulação [Ahrenholz 2010]. O *broker* MQTT utilizado foi o `mosquitto` [Light 2017]. Além disso, o módulo de sincronismo foi implementado na linguagem de programação C++, utilizando as bibliotecas *Opendht* (versão modificada) e *Paho-MQTT*.

Foram realizadas 10 rodadas de testes com 5, 10, 15, 20 e 25 *brokers* e módulo de sincronismo. Em cada teste, um (1) cliente MQTT publica 100 vezes uma informação em um determinado tópico no intervalo de 1 segundo. Foi recuperado o pior tempo de sincronismo de cada rodada, portanto, o maior tempo e feita uma média entre eles. O tempo de recebimento considerado foi após validação do certificado do emissor e da assinatura da mensagem, portanto, considerando o processamento do pacote recebido por meio da DHT.

O resultado é apresentado na Figura 4 com escala de tempo em nanosegundos. O tempo alcançado com 5, 10, 15 e 20 dispositivos foi razoável. Com 25 dispositivos, a capacidade de processamento do hospedeiro pode ter afetado o resultado. Foi utilizada uma máquina com processador Intel® Core® i3, *Random Access Memory* de 4 GigaBytes e sistema operacional Ubuntu 18.04.

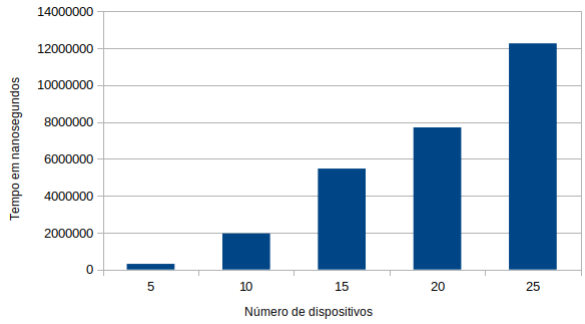

Figura 4. Tempo de sincronismo de dados

6. Conclusão

A arquitetura proposta apresentou um tempo de sincronismo de dados relativamente baixo. Em torno de 12 milissegundos (ms) com uso de 25 nós. Entretanto, para redundância de dispositivos, não é usual tal quantidade. A proposta é genérica para qualquer tipo de *broker*, sem a necessidade de modificação do código-fonte das aplicações atuais. Além disso, o respectivo trabalho alcança prerrogativas como autenticidade, integridade e não-repúdio com a utilização de certificado e assinatura digital das informações trocadas na rede DHT.

Como trabalho futuro, testes mais elaborados devem ser realizados sobre a arquitetura. Ainda, a utilização de um mecanismo para diminuir o processamento em *brokers* MQTT pode ser explorada em conjunto. Adicionalmente, protocolos como *Virtual Router Redundancy Protocol* (VRRP) podem ser utilizados com o trabalho proposto para quando o *broker* principal falhar, outro assuma seu lugar utilizando o mesmo endereço *Internet Protocol* (IP).

Referências

Ahrenholz, J. (2010). Comparison of core network emulation platforms. In *2010-Milcom 2010 Military Communications Conference*, pages 166–171. IEEE.

Firdous, S. N., Baig, Z., Valli, C., and Ibrahim, A. (2017). Modelling and evaluation of malicious attacks against the iot mqtt protocol. In *2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pages 748–755. IEEE.

Garcia, J. T., Dias, L. V., and Rizzetti, T. A. (2020). Um serviço para prover autenticação e revogação de nós na rede dht. *Revista Eletrônica Argentina-Brasil de Tecnologias da Informação e da Comunicação*, 3(1).

Jutadhamakorn, P., Pillavas, T., Visoottiviseth, V., Takano, R., Haga, J., and Kobayashi, D. (2017). A scalable and low-cost mqtt broker clustering system. In *2017 2nd International Conference on Information Technology (INCIT)*, pages 1–5. IEEE.

Kashyap, M., Sharma, V., and Gupta, N. (2018). Taking mqtt and nodemcu to iot: Communication in internet of things. *Procedia computer science*, 132:1611–1618.

Kotak, J., Shah, A., and Rajdev, P. (2019). A comparative analysis on security of mqtt brokers.

Light, R. A. (2017). Mosquitto: server and client implementation of the mqtt protocol. *Journal of Open Source Software*, 2(13):265.

Niruntasukrat, A., Issariyapat, C., Pongpaibool, P., Meesublak, K., Aiumsupucgul, P., and Panya, A. (2016). Authorization mechanism for mqtt-based internet of things. In *2016 IEEE International Conference on Communications Workshops (ICC)*, pages 290–295. IEEE.

Quincozes, S., Emilio, T., and Kazienko, J. (2019). Mqtt protocol: Fundamentals, tools and future directions. *IEEE Latin America Transactions*, 17(09):1439–1448.

Rahimi, N., Sinha, K., Gupta, B., Rahimi, S., and Debnath, N. C. (2016). Ldepth: A low diameter hierarchical p2p network architecture. In *2016 IEEE 14th International Conference on Industrial Informatics (INDIN)*, pages 832–837. IEEE.

Sen, S. and Balasubramanian, A. (2018). A highly resilient and scalable broker architecture for iot applications. In *2018 10th International Conference on Communication Systems & Networks (COMSNETS)*, pages 336–341. IEEE.

Sethi, P. and Sarangi, S. R. (2017). Internet of things: architectures, protocols, and applications. *Journal of Electrical and Computer Engineering*, 2017.

Soni, D. and Makwana, A. (2017). A survey on mqtt: a protocol of internet of things (iot). In *International Conference On Telecommunication, Power Analysis And Computing Techniques (ICTPACT-2017)*.

Srinivasan, A. and Aldharrab, H. (2019). Xtra—extended bit-torrent protocol for authenticated covert peer communication. *Peer-to-Peer Networking and Applications*, 12(1):143–157.