Power-Aware Security Protocols for the Internet of Things

Tiago Diogo

Instituto Superior Técnico, Avenida Rovisco Pais 1, Lisboa,
tiago.diogo@tecnico.ulisboa.pt

1 Introduction

The Internet of Things (IoT) can be seen as web of interconnected devices that go from everyday wearable objects into fully deployed sensor networks.Despite the huge variety and characteristics of these devices, one thing that they all have in common in the constrained nature they're built uppon. In order to enable the massive deploy to be expected in the near future [1] *IoT devices must be accessible and affordable, capable of operating under lossy wireless networks while being battery powered.*

2 Main Goals

Given the constraints and limitations of IoT devices described in the previous chapter, the first objective of this work is to identify existing protocols at the OSI application layer that take into account those constrains and are design to allow communication between these devices without consuming an amount of resources that would be appropriate for standard devices but excessive for the IoT ones.

After the analysis of the existing solutions, a baseline of power consumption will be established. Then, the focus will move towards adding confidentiality to the transmitted information by securing the channel. Once both the application level protocols and proper security solutions are defined, experiments will be performed so that the added power consumption cost of adding the security layers can be measured, profiled and documented therefore enabling the finding of the best parameters for a desired level of security.

The work will then proceed towards finding effective counter-measures against a specific group of attacks that targets the IoT devices by intensifying the use of its resources therefore draining the available power and placing the node offline. The ultimate goal is to propose an energy-efficient security mechanism that can resist these power-drain (a.k.a vampire) attacks.

[1] http://blogs.wsj.com/cio/2015/06/02/internet-of-things-market-to-reach-1-7-trillion-by-2020-idc/

3 Related Work

3.1 Protocol Analysis and Selection

*There are many alternatives and some proposed standards when it comes to choose a protocol for IoT communications. The decision must be based on the particularities of the devices to be used and the objective of the application itself, however a thoroughly analysis of the existing solutions is a proper way to unveil the strong and weak points of each protocol providing a good basis for an informed decision. (* maneira boa de referir o paper aqui *) a survey from January 2015 [1] covering the main application and network protocols will be the starting point for the analysis to follow.*

Hypertext Transfer Protocol (HTTP)

HTTP is an application level protocol that works in the request-response model and is the foundation of data communication on the World Wide Web (WWW) It's primarily design to run over Transmission Control Protocol (TCP) which is a problem in lossy and constrained environments due to the delivery assurances and congestion control algorithms it employs. Besides, HTTP is verbose, text-based, and not suited for compact message exchanges. Moreover, the header size required for a message exchange can leave too few payload space in constrained networks like the IEEE 802.15.4-based networks where the Maximum Transmission Unit (MTU) size of the protocol is 127 bytes. These protocol specifications would not raise any issues in standard WWW communications, but when it comes to constrained environments it is clear that the protocol is not adequate to the necessities of IoT devices and networks.

Constrained Application Protocol (CoAP)

CoAP is a document transfer protocol based on REpresentational State Transfer (REST) on top of HTTP functionalities. CoAP objective is to enable tiny constrained devices to use RESTful interactions, where clients and servers expose and consume web services using Universal Resource Identifiers (URIs) together with HTTP get, post, put and delete methods. Unlike REST, CoAP runs over User Datagram Protocol (UDP) instead of TCP which makes it suitable for full IP networking in small micro-controllers. Retries and reordering are implemented at the application stack using a messaging sub-layer that detects duplicated messages and provides reliable communication using different types of messages. Confirmable messages must be acknowledged by the receiver, nonconfirmable one are fire and forget. While being a lightweight protocol, CoAP still provides important features: (não consigo meter um itemize aqui... pls help...) Resource Observation - CoAP can extend the HTTP request model with the ability to observe a resource therefore monitoring resources of interest using a publish/subscribe mechanism. Resource Discovery - CoAP servers provide a list of resources using well-known URIs that allow clients to discover what resources are *provided and their types. Interoperability - since CoAP is based on the REST architecture, a simple proxy enables CoAP to easily interoperate with HTTP. A study that compared CoAP and HTTP using mobile networks concluded that there is no situation where CoAP would consume more resources than HTTP [2]*

Message Queue Telemetry Transport (MQTT)

MQTT is a pusblish/subscribe messagins protocol designed for lightweight Machine to Machine (M2M) communications. It employs a client/server model and consists of three components, the publisher, the subscriber and a broker. Subscribers register their interest for a specific topic and then get informed by the broker when a publisher generates data regarding that topic. Every message is a discrete chuck of data, opaque to the broker. The broker, on his side, achieves security by checking authorization of the publishers and subscribers. MQTT supports three Application Level Quality of Service (QoS) levels:

- At Most Once (Fire and Forget): A message won't be acknowledged by the receiver or stored and redelivered by the sender.
- At Least Once: It is guaranteed that the message will be delivered to the receiver, but more that one can reach the destination. The sender stores the message until it gets an acknowledge from the receiver.
- Exactly Once: A four-way handshake mechanism is used to guarantee that the message will be received exactly once by the counterpart.

MQTT has support for persistence messages stored on the broker, where the most recent message will be sent to a client that subscribes that topic. Clients can register a custom message to be sent to the broker on disconnect enabling other subscribers to know when a device disconnects. MQTT runs on TCP which in some cases causes drawbacks in performace. A performace evaluation of MQTT and CoAP [3] provides comparisons on several protocol facets:

- Influence of Packet Loss on Delay: With low values of packet loss, MQTT experienced lower delays, but as the packet loss increased CoAP performed better. This is due to the greater TCP overheads involved in the retransmissions of messages when compared to UDP
- Influence of Packet Loss on Data Transfer: CoAP generated less data for each packet loss versus all the MQTT QoS levels.
- Overheads for Message Sizes: When packet loss rate is low, CoAP generates less overhead than MQTT for all message sizes, but as message size grows, the reverse is true. This happens because when the message size is is large, the probability that UDP looses the message is higher than TCP which causes CoAP to retransmit the whole message more often than MQTT

referir que foi desenvolvido o Message Queue Telemetry Transport for Sensor Networks (MQTT-SN) [4] especifico para sensor networks Datagram Transport Layer Security (DTLS) Transport Layer Security (TLS)

Table 1. IoT Application Protocols Comparison

Application Protocol	Publish/ Subscribe	RESTful	Request/ Response	QoS	Transport	Security
HTTP	✗	✓	✓	✗	TCP	TLS
CoAP	✓	✓	✓	✓	UDP	DTLS
MQTT	✓	✗	✗	✓	TCP	TLS
MQTT-SN	✓	✗	✗	✓	UDP	DTLS

Table 2. Protocol Stack Comparison Overview

Web	IoT
HTTP	CoAP/MQTT
TLS	DTLS
TCP	UDP
IPv6	6LoWPAN

mais coisas aqui

→ *colocar aqui o desenho bonito das layers web vs layers IoT* ←

3.2 Security and Improvements

TLS

This is the resume about TLS afoebfoewufbewoçubewfwefwfe

DTLS

This is the resume about DTLS paper que diz mal do DTLS [5] e diz que outra solução é necessária

3.3 Attack Analysis, Detection and Prevention

3.3.1 Internet Attacks

do some work identifying threats to the web in general

3.3.2 IoT Attacks

focus on the power-drain attacks alguns papers ainda não filtrados: [6, 7]

4 Proposed Solution

5 Work Evaluation

6 Work Planning

7 Conclusion

References

1. Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., Ayyash, M.: Internet of Things: A Survey on Enabling Technologies, Protocols and Applications. IEEE Communications Surveys & Tutorials **PP**(99) (2015) 1–1
2. Savolainen, T., Javed, N., Silverajan, B.: Measuring Energy Consumption for RESTful Interactions in 3GPP IoT Nodes. (2014) 1–8
3. Ma, X., Valera, A., Tan, H.x., Tan, C.K.y.: Performance Evaluation of MQTT and CoAP via a Common Middleware. (April) (2014) 21–24
4. Ibm: MQTT For Sensor Networks (MQTT-SN) Protocol Specification. (2013) 28
5. Alghamdi, T.a., Lasebae, A., Aiash, M.: Security analysis of the constrained application protocol in the Internet of Things. 2nd International Conference on Future Generation Communication Technologies, FGCT 2013 (2013) 163–168
6. Vasserman, E.Y., Hopper, N.: Vampire attacks: Draining life from wireless ad Hoc sensor networks. IEEE Transactions on Mobile Computing **12**(2) (2013) 318–332
7. Vanitha, K., Dhivya, V.: A Valuable Secure Protocol to Prevent Vampire Attacks In Wireless Ad Hoc Sensor Networks. **3**(3) (2014)