TCP in the Internet of Things: from ostracism to prominence

Carles Gomez[1], Andrés Arcia-Moret[2], Jon Crowcroft[2]
[1]Universitat Politècnica de Catalunya (UPC)
[2]University of Cambridge
E-mail: carlesgo@entel.upc.edu, andres.arcia@cl.cam.ac.uk, jon.crowcroft@cl.cam.ac.uk

Keywords- TCP, Internet of Things, evaluation, HTTP, CoAP, MQTT, AMQP.

Abstract

TCP has traditionally been neglected as a transport-layer protocol for the Internet of Things (IoT). However, recent trends and industry needs are favoring TCP presence in IoT environments. In this paper, we first motivate and describe the main IoT scenarios where TCP will be used. We then analyze the historically claimed issues of TCP in the IoT context. We argue that, in contrast to generally accepted wisdom, most of those possible issues fall in one of the following categories: i) are also found in well accepted IoT end-to-end reliability mechanisms, ii) can be solved, or iii) are not actual issues. Considering the future prominent role of TCP in the IoT, we provide recommendations for lightweight TCP implementation and suitable operation in such scenarios, based on our IETF standardization work on the topic.

1. Introduction

The Transmission Control Protocol (TCP) was designed four decades ago as a connection-oriented transport-layer protocol that provides end-to-end reliable and ordered data delivery between applications running on Internet hosts [1]. TCP has ever since been the dominant transport-layer protocol on the Internet, as many major applications (e.g. the WWW, e-mail, file transfer, instant messaging, etc.) have benefitted from its service. However, TCP has faced and overcome significant challenges as the Internet has evolved beyond its initial characteristics. For example, despite the underlying assumption of TCP congestion control whereby the Internet is a wired network, and the issues that arise in wireless environments [2], TCP has been successfully used in mobile networks. Optimization techniques have contributed to today's TCP pervasive presence in mobile phones [3].

A new challenge for TCP is the Internet of Things (IoT). In this major networking trend, it is envisioned that tens of billions of inexpensive devices (e.g. sensors, actuators, etc.) attached to daily life objects will be connected to the Internet to enable *smart* scenarios. However, IoT devices typically exhibit significant constraints (regarding memory, processing, and energy), use low rate and error-prone links, and their networks often follow a multihop topology [4]. Due to these harsh networking conditions, TCP has often been severely criticized as a transport-layer protocol for the IoT. In consequence, many initial IP-based IoT deployments resorted to using UDP with application-layer

reliability [5]. The same approach was followed for the Constrained Application Protocol (CoAP), a lightweight RESTful application-layer protocol developed at the IETF for the IoT [6]. Likewise, the IPv6 over Low power Wireless Personal Area Networks (6LoWPAN) activity developed optimizations such as header compression for UDP, while TCP was neglected [7].

However, the need for graceful integration of CoAP with enterprise infrastructure has recently triggered the development of a CoAP over TCP specification [8]. On the other hand, HTTP, which relies on TCP at the transport layer, has been used and is being optimized for IoT environments, leveraging its *mainstream* position [9]. Furthermore, messaging protocols such as Message Queue Telemetry Transport (MQTT) [21] and Advanced Message Queuing Protocol (AMQP) [22], both assuming TCP underneath, have achieved IoT market presence [27]. These recent industry and standardization tendencies suggest that TCP may gain extensive support in IoT scenarios soon. However, it is necessary to study the potential issues of TCP, and determine how it should be used, in IoT environments. This need has motivated our IETF standardization work to provide guidelines on the use of TCP for IoT [10].

In this paper, we focus on issues of and solutions for TCP in the IoT. The remainder of the paper is organized as follows. In section 2, we analyze the IoT end-to-end connectivity scenarios where TCP will be used. In section 3, we discuss a comprehensive list of critiques to TCP in an IoT context. In sections 4 and 5, we describe and evaluate, respectively, simple measures for lightweight TCP implementation and suitable operation in IoT environments. Section 6 discusses implementation challenges, and Section 7 concludes the paper.

2. TCP in the Internet of Things

Connecting *Things* to the Internet allows end-to-end connectivity between IoT devices and other computers on the same network. In this paradigm, cloud backend systems can communicate with IoT devices (e.g. for sensor data centralization, actuator triggering, and device management). In this section, we describe the main protocol and architectural options for end-to-end connectivity with IoT devices where TCP is used. We focus on the scenarios that involve HTTP, CoAP, MQTT and AMQP.

2.1. HTTP

HTTP offers several advantages as a protocol for the IoT: it is a free, open standard and, being the mainstream application-layer protocol on the Internet, HTTP development tools are numerous. Moreover, it is the protocol with the highest probability of passing security middleboxes. The recent HTTP/2 is more suitable than HTTP/1.1 in an IoT context. HTTP/2 has a binary, compact header, while pseudo-header fields can be compressed by using a format called HPACK. Currently, an IETF specification is being developed for using HTTP/2 in IoT scenarios [9]. Therefore, HTTP (and thus TCP) is an important candidate for IoT device communication (Figure 1.a)).

2.2. CoAP

CoAP was designed as a lightweight alternative to HTTP/1.1, keeping the basic principles of HTTP such as the REST architecture, albeit with considerably less complexity. To fully exploit its potential, CoAP allows interoperability with HTTP via protocol translation proxies (Figures 1.c) and 1.d)). CoAP was originally designed over UDP, with optional, stop-and-wait reliability. However, deployment experience has shown the need to enable CoAP over TCP (Figure 1.b)) or over WebSockets (WS, Figure 1.c)), in order to overcome connectivity limitations introduced by corporate firewalls [8].

2.3. MQTT

MQTT is an ISO/IEC messaging protocol designed for monitoring applications. It is based on the publish-subscribe paradigm, by which publishers (e.g. sensors) transmit data messages to a broker; the latter delivers such messages to interested entities, called subscribers (e.g. backend systems). This flexible approach places complexity in the broker. Furthermore, MQTT defines a lightweight header format and requires a small code footprint. In MQTT, a TCP connection is established between a publisher or a subscriber and the broker (Figure 1.e)).

2.4. AMQP

AMQP is another ISO/IEC messaging protocol, originally developed for the finance industry. It supports a variety of broker-based architectures, including publish-subscribe. AMQP provides more elaborate mechanisms (e.g. for fine-grained control, queue management and error handling) than MQTT, at the expense of greater implementation complexity and larger message headers. AMQP is also based on TCP (Figure 1.f)).

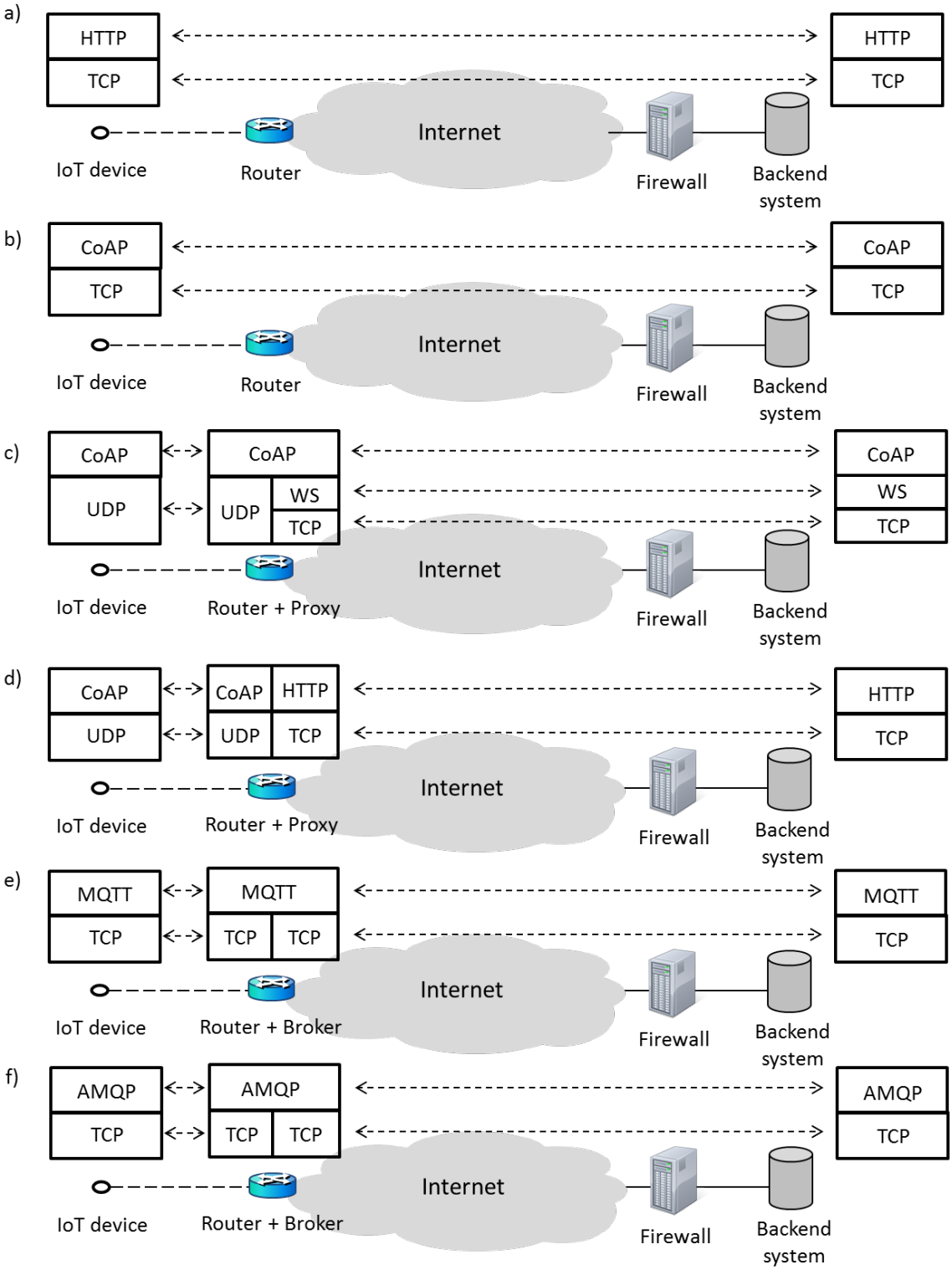

Figure 1. Main options for backend-to-IoT device communication: a) CoAP/TCP; b) HTTP/TCP; c) CoAP/UDP to CoAP/WS/TCP proxy; d) CoAP/UDP to HTTP/TCP proxy; e) MQTT/TCP; f) AMQP/TCP

3. Analysis of claimed TCP issues in Internet of Things scenarios

TCP has been criticized as a protocol for the IoT. In this section we review the main potential problems, and the faults indicated by the networking community, of using TCP in IoT scenarios. Table 1 summarizes the claimed issues, the outcome of our discussion, solutions available (if any), and relevant use cases and application domains.

Claimed issue	Discussion	Solution(s)	Relevant use case(s)	Relevant application domain(s)
Congestion control activation after non-congestion losses	Not specific to TCP (inability to distinguish the cause of packet loss)	Experimental	Remote control, switch, alarm	Home, building
			Control and critical monitoring	Health, industry
			Firmware update	Potentially all
Link layer interaction	Not specific to TCP (common for end-to-end ARQ)	Experimental	Remote control, switch, alarm	Home, building
			Control and critical monitoring	Health, industry
			Firmware update	Potentially all
Header overhead	At least 12 bytes greater size than UDP header	TCP header compression (to be developed)	Battery-enabled sensors/actuators	City, agriculture, forest, home, health, industry
Long TCP connection infeasible due to sleep periods	False claim (if properly configured RDC in use)	Not needed	Battery-enabled sensors/actuators	City, agriculture, forest, home, health, industry
Lack of transport service flexibility	TCP always offers reliable service	No	Non-critical monitoring	City, forest, agriculture, home
High latency	Due to three-way handshake	i) Long-lived connections ii) TFO	Remote control, switch, alarm	Home, building
			Control	Industry
Multicast incompatibility	TCP is a unicast protocol	No	Lighting applications, scenes	Home, building
RTO algorithm issues	RTO not designed for IoT scenarios	Use of CoCoA (see next section)	All scenarios, except for high quality wired links in building automation [19]	
High complexity	False claim (TCP was designed for computers similar to today's IoT devices)	Not needed	All scenarios, except for higher performing IoT devices (e.g. Raspberry Pi)	

Table 1. Summary of claimed issues, discussion, solutions, relevant use cases and application domains for TCP in an IoT context

3.1. Congestion control and packet loss

TCP congestion control was designed in the late eighties when Internet links were mostly wired. It was assumed that packet losses were due to congestion since corruption in a wired link was unlikely [15]. Congestion control mechanisms were designed to avoid congestion-induced network collapse. When a packet loss is detected, a TCP sender reduces its segment rate. However, in IoT environments, packet losses may occur for several reasons besides congestion. First, most IoT link technologies are wireless (or use wired noisy media, as in power-line communication), thus being typically error-prone. Secondly, many IoT networks follow a mesh topology. Route changes, e.g. due to node mobility or temporary link quality degradation, lead to connectivity gaps and may also induce packet reordering (which TCP may treat as packet loss).

Suboptimal performance of TCP under non-congestion losses is a well-known problem [3]. Proposed solutions, such as Explicit Loss Notification, require distinguishing the reason of packet losses, which is not always possible, and neither have been standardized nor widely deployed. However, inability to determine the reason of packet losses is not specific to TCP. In fact, any other well designed Automatic Repeat reQuest (ARQ)-based mechanism for end-to-end reliability (e.g. the one in CoAP) will trigger congestion control measures after a packet loss, incurring underperformance like TCP [18].

3.2. Link layer interaction

Many IoT link-layer protocols use ARQ. Examples include IEEE 802.15.4, ITU-T G.9959 (Z-Wave), Bluetooth Low Energy or IEEE 802.11ah. Link-layer ARQ may increase TCP performance, as link Round Trip Time (RTT) is expected to be lower than end-to-end path RTT, allowing local recovery of a lost packet before the TCP sender triggers a retransmission (and congestion control). However, if a link suffers quality degradation, the link-layer ARQ mechanism may perform retries, increasing end-to-end latency, and sometimes leading to spurious TCP retransmissions [14]. This problem is not TCP-specific, and will happen when two RTO mechanisms run in parallel at two different layers, as with CoAP (over UDP) RTO over any ARQ-based link layer.

3.3. Header overhead

The TCP header has a minimum size of 20 bytes, which is greater than the 8-byte UDP header. Furthermore, 6LoWPAN header compression allows to encode the UDP header efficiently (typically, reducing it to 4 bytes), but not the TCP one [7]. Note that RFC 1144 TCP header compression [29] is not suitable for lossy links, and Robust Header Compression (ROHC), which addresses issues of the former, is too heavy for IoT devices. In fact, ROHC ranks among the most complex IETF protocols, thus unsuitable for constrained devices which often have 8- or 16-bit microprocessors and a RAM of ~10 to ~50 kB [4]. TCP header compression was once proposed for 6LoWPAN

(achieving a 6-byte TCP header with 95% probability), but it was neither completed nor standardized. Therefore, TCP header compression for IoT scenarios is an open issue.

3.4. TCP connection maintenance

IoT devices may run on a limited energy source, e.g. a battery. Communication, and in particular, idle listening, is the main energy-consuming component in such devices. In order to save energy, many IoT devices use radio duty cycling (RDC), by which the radio interface is kept in off state by default, and is turned on for communication under certain conditions [11]. It has been claimed that "devices may frequently go into sleep mode, thus it is infeasible to maintain a long-lived connection in IoT applications" [12]. However, RDC techniques allow the exchange of packets between an energy-constrained device and another device, at the expense of increased delay and buffering requirements [11]. With appropriately configured RDC mechanisms, and since storing state consumes a low amount of energy, we argue that long TCP connections are perfectly feasible.

3.5. Latency

Certain IoT applications require low latency, such as alarm activation (e.g. after fire detection), and human-triggered interaction between controls and actuators (e.g. lightbulbs, appliances, etc.). The requirement for short-lived TCP connections for IoT environments expressed by some researchers (see the previous subsection) may translate into opening a new connection every time new data has to be sent, increasing delay due to connection establishment [12]. In contrast, delay is minimized in a long-lived connection, which is only established once and may subsequently be reused for data exchanges. An alternative is TCP Fast Open (TFO), which allows embedding data in SYN and SYN-ACK packets, thus saving one RTT compared to the traditional approach whereby the three-way handshake precedes data exchange [13]. For security reasons, TFO requires the negotiation of a cookie, which is included in SYN packets.

3.6. Always-reliable service

Monitoring applications in IoT often tolerate a fraction of lost sensor readings. This can be exploited to save energy and bandwidth by using unacknowledged transmission. For instance, CoAP (over UDP) supports optional non-confirmable transmission. However, if TCP is used, all upper layer messages will be acknowledged at the transport layer, precluding the application developer from considering the lighter, unreliable approach.

3.7. Multicast

There exist IoT applications that involve communication between a sender and a group of receivers. Examples include controlling a specific group of lights (e.g. in smart homes or smart cities) and group firmware updates. Such applications benefit from the packet economy of multicast to save energy and bandwidth. For instance, CoAP group

communication uses IP multicast [16]. However, TCP is a unicast protocol: therefore it is not suitable as a transport-layer protocol for multicast. (This is one of the reasons that originally favored UDP as a transport-layer protocol for CoAP.)

3.8. RTO algorithm

TCP uses its well-known RTO algorithm, which adaptively determines the RTO by applying an EWMA smoothing scheme on RTT samples [17]. However, this algorithm was not designed considering IoT scenarios. Problems of TCP RTO in this context are summarized next [18].

Karn algorithm

By the Karn algorithm, only *strong* RTTs (i.e. RTTs for which the sender has not performed retransmissions) are considered for RTO computation. However, IoT scenarios are lossy; therefore it is also necessary to use *weak* RTTs (for which the sender has run into retransmission), to avoid long periods without any input to the RTO computation, even if weak RTTs provide ambiguous information.

Lack of aging

Network conditions may change over time in IoT environments, from temporary congestion due to message bursts (e.g. when several sensors detect and communicate a global event) to low offered load periods. The RTO algorithm may thus converge to a value adapted to an outdated situation. The TCP RTO lacks an *aging* mechanism, allowing to decay the RTO estimate towards the default RTO value after a long period without RTT samples.

Constant backoff factor

Many IoT networks follow the multihop topology and are connected to the Internet through a border router. In these scenarios, nodes that are close to the border router will be favored by a constant backoff factor of 2, as their RTO will converge to low values, allowing quick retries. However, remote nodes will converge to high RTO values, leading to high retry delay, creating a fairness problem that degrades network performance.

Synchronization issues

Due to the periodic nature of sensor transmissions in many IoT applications, such transmissions may synchronize, leading to collisions and losses. Dithering allows avoiding such effects; however, it is not available in TCP RTO.

3.9. Protocol complexity

The IoT community has often regarded TCP as a *complex* protocol [23, 24]. While TCP has evolved over time, it is backwards compatible with its RFC 793 specification. In the

early eighties, TCP was running on computers with very limited processing and memory characteristics. Those computers would be categorized today as constrained devices [4].

4. A TCP profile for IoT devices

As aforementioned, TCP presence in IoT scenarios is foreseen to increase dramatically. On the other hand, some of the potential issues of TCP for IoT environments may be addressed or mitigated by adequately configuring TCP. For these reasons, a specification is being developed in the IETF LWIG working group (WG), in cooperation with the IETF CoRE and TCPM WGs, intended to offer simple measures for lightweight and suitable TCP operation in IoT environments [10]. This section summarizes the main recommendations from the specification.

4.1. Maximum Segment Size

In many IoT scenarios, an adaptation layer based on 6LoWPAN is needed to enable IPv6 over the lower layers [19]. Such adaptation layer may not grant support for packets larger than 1280 bytes. Therefore, in order to avoid IP-layer fragmentation when TCP is used, the Maximum Segment Size (MSS) must be set appropriately.

4.2. Window size

TCP has often been criticized as a *complex* protocol by the IoT community [23, 24]. This claim is partly due to its sliding window management mechanisms, often optimized for high bandwidth scenarios [25]. However, in IoT networks, traffic patterns are typically transactional, e.g., sensors sending short data messages infrequently. In fact, the reliability mechanism in CoAP provides stop-and-wait operation, which is a lightweight approach well accepted by the IoT community [23]. TCP flow control can provide stop-and-wait functionality by using a single-MSS window. This approach simplifies TCP implementation and operation in several ways. Firstly, buffer space and buffer management requirements are reduced [26]. Secondly, segment reordering is avoided, as a new segment cannot be sent until the previous one has been acknowledged.

Nevertheless, software updates is a use case where a window greater than one MSS may be beneficial, in order to reduce transfer time. An IoT device might also benefit from sending several segments consecutively (e.g. a batch of accumulated readings), as the energy cost of radio warm-up and cool-down transitions before and after communication, respectively, could be amortized. Therefore, we recommend using stop-and-wait, while allowing implementers choose the approach that better suits their needs.

4.3. RTO algorithm

An enhanced RTO algorithm is being standardized by the IETF CoRE WG for CoAP, as part of the CoAP Congestion Control/Advanced (CoCoA) specification [18]. CoCoA has been designed for IoT scenarios. It is based on the TCP RTO algorithm [17]; however, it also uses weak RTTs, an aging mechanism, a variable backoff factor, and dithering. CoCoA outperforms state-of-art TCP RTO variants such as Linux RTO or Peak Hopper, at the expense of a complexity increase that does not pose a problem for IoT devices [18]. Therefore, we recommend using CoCoA for TCP.

4.4. TCP connection establishment and maintenance

In order to save energy and bandwidth resources in IoT scenarios, it is fundamental to minimize communication overhead. The penalty of TCP connection establishment becomes asymptotically negligible as TCP connection lifetime increases. Therefore, we recommend maintaining long-lived TCP connections whenever possible (e.g. for AMQP- or MQTT-based approaches). Otherwise, if the TCP connection needs to traverse a middlebox (e.g. a firewall, a NAT, etc.), it may face the issue that many middleboxes silently remove connection state after a few minutes of inactivity [20]. This behavior forces the two communicating TCP endpoints to establish a new TCP connection for transmitting new data. An alternative is TFO, which is more efficient than creating a new TCP connection per notification if the TFO cookie update rate is at least one order of magnitude below the notification rate.

4.5. TCP options

Several TCP options such as Timestamps, Window Scale, and Selective Acknowledgments (SACK) were designed to increase TCP performance over high bandwidth-delay product and high-speed paths, where using a high TCP window size is critical to achieving good performance. If stop-and-wait is used, these TCP options cannot provide benefits and thus their support can be avoided. However, if a window size greater than a single MSS is used in often lossy IoT environments, SACK is recommendable. With this option, a sender will not retransmit data unnecessarily.

5. Evaluation

This section evaluates performance of TCP in an IoT context, considering different settings, in terms of energy efficiency, delay, and throughput.

5.1. Energy efficiency

We investigate the energy efficiency of UDP- and TCP-based architectures and options for communication between a battery-enabled IoT device and the backend. The following conditions are assumed for the IoT device: the CC2530 IEEE 802.15.4 radio chip [28], RDC with a poll rate of 0.1 Hz, MAC-layer acknowledgments, a CR2032 button-cell battery, 6LoWPAN header compression for IPv6 and UDP (with 11-byte and 4-byte headers, respectively), and periodic notification of 25-byte sensor readings.

Figure 2 shows the theoretical lifetime of the considered IoT device. Results illustrate that differences in the evaluated architectures are significant for notification periods lower than 10^4 seconds as energy consumption due to RDC and sleep intervals becomes dominant. Applications that do not require reliability benefit from CoAP non-confirmable transmission over UDP. However, common security middleboxes only allow TCP traffic.

Among the TCP-based approaches, an always-open TCP connection yields the best performance. MQTT and AMQP exploit this approach, by leveraging a TCP connection with a broker in the IoT device subnet. HTTP/2 or CoAP will need to use TFO if traversed firewalls' state timers are set to values lower than the notification period. Small differences in application-layer protocol header size lead to IoT device lifetime differences below 1% (not shown in Figure 2 for clarity) due to the overhead incurred by the layers below TCP. Similarly, compressing the TCP header (see subsection 3.3) only achieves a minor IoT device lifetime increase (e.g. 3% and 1% for notification periods of 10 s and 100 s, respectively). For TCP payloads between 80 and 93 bytes, TCP header compression avoids fragmentation, yielding greater lifetime increase (up to ~6%). Establishing a new TCP connection for each notification degrades IoT device lifetime for high notification rates.

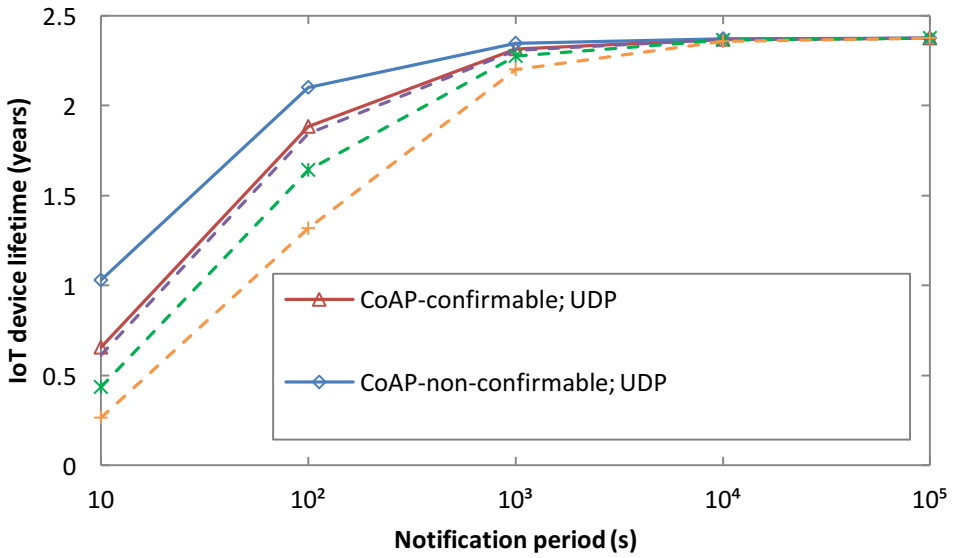

Figure 2. IoT device lifetime of UDP- and TCP-based approaches. Server push is assumed for HTTP/2

5.2. Delay

Figure 3 illustrates the theoretical delay for a notification sent by an IoT device under the conditions assumed in the previous subsection, over a single link. TFO yields slightly greater delay than using an always-open connection, due to addition of the cookie in the SYN packet. Opening a new TCP connection per notification increases

delay significantly, since a three-way handshake precedes each notification. Finally, note that TCP header compression has a minor impact on delay, except for the range of TCP payload values where it avoids fragmentation.

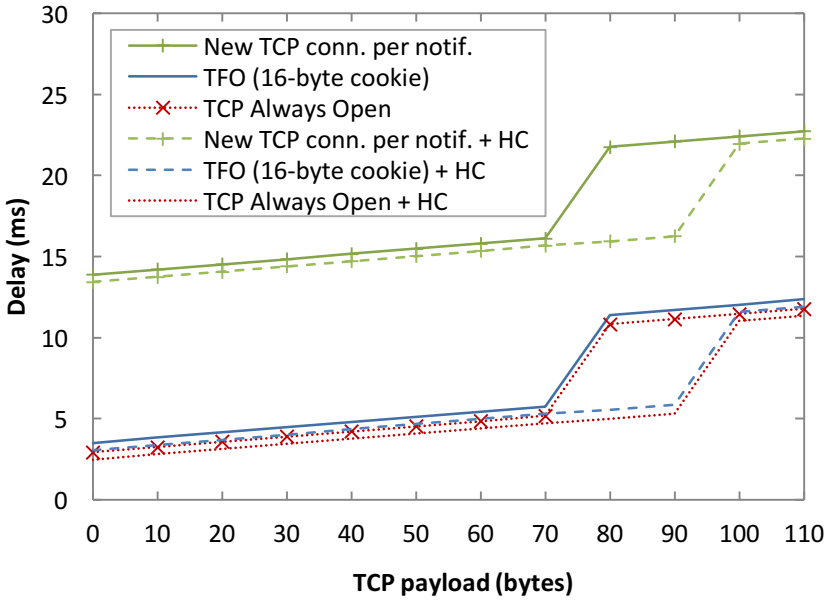

Figure 3. Notification delay as a function of TCP payload and TCP options. HC stands for Header Compression.

5.3. Throughput

In order to evaluate TCP throughput in an IoT scenario, we carried out single-flow experiments over up to 6-hop paths in a grid-shaped indoor testbed comprising 60 TelosB motes. We measured ~35% throughput improvement by increasing the window size from 1 to 3 segments. However, increasing the window size further induces congestion and damages performance.

On the other hand, network-wide throughput is relevant for IoT networks that suffer congestion intervals, e.g. when several sensors detect an event that must be communicated. In such situations, CoCoA RTO outperforms TCP RTO, yielding greater throughput, and lower settling time after message bursts [18].

6. Implementation challenges

Lightweight TCP implementation faces a fundamental challenge: the trade-off between simplicity and performance tunability. A Class 1 IoT device (with ~10 kB of RAM [4]) can run a complete IP-based protocol stack. However, a TCP implementation for such platform will probably need to be very simple (e.g. single-MSS window), with little margin for TCP tuning for a particular application. Less constrained IoT devices (e.g. Class 2 devices, with ~50 kB of RAM [4]) allow greater flexibility. Such platforms may afford higher-performing TCP implementations with higher memory footprint, which may however conflict with the rest of protocols in the stack and with the application

itself. In general, a holistic analysis must be carried out to determine which protocol features may need to be sacrificed across layers. In many scenarios, IoT devices in a subnet will share the same characteristics and application requirements; thus TCP settings may be homogeneous over the subnet.

7. Conclusion

While TCP has traditionally been neglected in IoT network designs, current trends suggest that TCP will gain extensive deployment in IoT scenarios. Particular drawbacks of TCP compared to UDP-based solutions include increased header overhead, lack of flexibility for loss-tolerant applications, and unsuitability for multicast (the latter precludes TCP for group-oriented applications). TCP underperforms UDP-based solutions for non-critical monitoring with relatively frequent sensor reading updates. However, with appropriate configuration, TCP can behave similarly to unicast end-to-end reliability mechanisms well accepted for the IoT, while integrating with middleboxes much better than UDP.

Acknowledgments

Carles Gomez has been funded in part by the Spanish Government and by the ERDF through the Jose Castillejo grant CAS15/00336, and through project TEC2016-79988-P. His contribution to this work has been carried out in part during his stay as a visiting scholar at the Computer Laboratory of the University of Cambridge. Andrés Arcia-Moret has been funded by the project Network as a Service (EP/K031724/2).

References

[1] V. Cerf, R. E. Kahn, “A Protocol for Packet Network Interconnection”, IEEE Transations on Communications, Vol Com-22, No 5, May 1974.

[2] H. Balakrishnan, V. Padmanabhan, S. Seshan, “A Comparison of Mechanisms for Improving TCP Performance over Wireless Links”, IEEE/ACM Transactions on Networking, Vol. 5, No. 6, Dec. 1997.

[3] H. Inamura, G. Montenegro, R. Ludwig, A. Gurtov, F. Khafizov, “TCP over Second (2.5G) and Third (3G) Generation Wireless Networks”, RFC 3481, Feb. 2003.

[4] C. Bormann, M. Ersue, A. Keranen, “Terminology for Constrained-Node Networks”, May 2014.

[5] D. Sturek, Z. Shelby, D. Lohman, S. Ashton, “Smart Energy Requirements for 6LowApp”, IETF Internet Draft, Oct. 2009.

[6] Z. Shelby, K. Hartke, C. Bormann, “The Constrained Application Protocol (CoAP)”, RFC 7252, Jun. 2014.

[7] Z. Shelby, C. Bormann, “6LoWPAN: The Wireless Embedded Internet”, John Wiley & Sons, 2009.

[8] C. Bormann, S. Lemay, H. Tschofenig, K. Hartke, B. Silverajan, B. Raymor, “CoAP (Constrained Application Protocol) over TCP, TLS, and WebSockets”, IETF Internet Draft, July 2016.

[9] G. Montenegro, S. Céspedes, S. Loreto, R. Simpson, “H2oT: HTTP/2 for the Internet of Things”, IETF Internet Draft, July 2016.

[10] C. Gomez, J. Crowcroft, “TCP over Constrained-Node Networks”, IETF Internet Draft, Mar. 2017.

[11] C. Gomez, M. Kovatsch, H. Tian, Z. Cao, “Energy-Efficient Features of Internet of Things Protocols”, IETF Internet Draft, Mar. 2017.

[12] Wentao Shang, Yingdi Yu, Ralph Droms, Lixia Zhang, “Challenges in IoT Networking via TCP/IP Architecture”, NDN Technical Report NDN-0038, Feb. 2016.

[13] Y. Chen, J. Chu, S. Radhakrishnan, A. Jain, “TCP Fast Open”, RFC 7413, Dec. 2014.

[14] P. Karn et al, “Advice for Internet Subnetwork Designers”, RFC 3819, July 2004.

[15] V. Jacobson, “Congestion avoidance and control”, in Proceedings of SIGCOMM, 1988.

[16] A. Rahman, E. Dijk, “Group Communication for the Constrained Application Protocol (CoAP)”, RFC 7390, Oct. 2014.

[17] V. Paxson, M. Allman, J. Chu, M. Sargent, “Computing TCP's Retransmission Timer”, RFC 6298, Jun. 2011.

[18] A. Betzler, C. Gomez, I. Demirkol, J. Paradells, “CoAP Congestion Control for the Internet of Things”, IEEE Communications Magazine, pp. 154-160, Jul. 2016.

[19] Y-G. Hong, C. Gomez, Y-H. Choi, D-Y. Ko, “IPv6 over Constrained Node Networks (6lo) Applicability & Use cases”, IETF Internet Draft, Mar. 2017.

[20] S. Hätönen et al, “An Experimental Study of Home Gateway Characteristics”, In Proceedings of SIGCOMM, 2010.

[21] “Information technology — Message Queuing Telemetry Transport (MQTT)” v3.1.1, ISO/IEC 20922:2016, Jun. 2016.

[22] "Information technology -- Advanced Message Queuing Protocol (AMQP) v1.0 specification", ISO/IEC 19464, Apr. 2014.

[23] C. Bormann, A. Castellani, Z. Shelby, "CoAP: an Application Protocol for Billions of Tiny Internet Nodes", IEEE Internet Computing, pp. 62-67 Mar/Apr, 2012.

[24] A. Zanella, et al., "Internet of Things for Smart Cities", IEEE Internet of Things Journal, pp. 22-32, Feb 2014.

[25] D. Borman et al., "TCP extensions for high performance", RFC 7323, Sep. 2014.

[26] A. Dunkels, "Full TCP/IP for 8-bit Architectures", in Proc. of MobiSys'03, May 2003.

[27] "Azure IoT device SDK for C", Sep. 2016. Available online: https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-device-sdk-c-intro

[28] C. Kim, "Measuring Power Consumption of CC2530 With Z-Stack", Application Note AN079, Sep. 2012.

[29] V. Jacobson, "Compressing TCP/IP Headers for Low-Speed Serial Links", RFC 1144, Feb. 1990.

Biographies

Carles Gomez received his Ph.D. degree from Universitat Politècnica de Catalunya in 2007. He is an associate professor at the same university. He is a co-author of numerous technical contributions including papers published in journals and conferences, IETF RFCs, and the book "Sensors Everywhere — Wireless Network Technologies and Solutions". His current research interests focus mainly on the Internet of Things. He serves as Associate Editor of the Journal of Ambient Intelligence and Smart Environments.

Andrés Arcia-Moret received the B.Eng. and M.Sc. degree in Computer Science from the University of Los Andes, Venezuela. He received the PhD degree in Computer Science from the IMT Atlantique, France, in 2009. He has been a fellow researcher in the IRISA/CNRS, Rennes, France (2012), and in the Guglielmo Marconi Laboratory at the ICTP, Trieste, Italy (2013). Since 2015, he is a Research Associate in the Computer Laboratory at the University of Cambridge, UK.

Jon Crowcroft has been the Marconi Professor of Communications Systems in the Computer Laboratory, University of Cambridge since October 2001. He has worked in the area of Internet support for multimedia communications for over 30 years. Three main topics of interest have been scalable multicast routing, practical approaches to

traffic management, and the design of deployable end-to-end protocols. Current active research areas are opportunistic communications, social networks, and scaling infrastructure-free mobile systems.