Received: 30 October 2017 | Revised: 15 December 2017 | Accepted: 8 January 2018

DOI: 10.1002/ett.3287

RESEARCH ARTICLE

WILEY

Encapsulation of real-time IoT CoAP traffic

Rolando Herrero

College of Engineering, Northeastern University, Boston, MA 02115-5000, USA

Correspondence
Rolando Herrero, College of Engineering, Northeastern University, 360 Huntington Avenue, Boston, MA 02115-5000, USA.
Email: r.herrero@northeastern.edu

Abstract

The *Constrained Application Protocol* (CoAP) has been instrumental to the deployment of wireless sensors and actuators in *Internet of Things low-power low-rate networks*. Typically, a large number of such devices interact with a single gateway that routes traffic to servers where complex processing is performed. CoAP provides a highly efficient end-to-end mechanism that relies on the *User Datagram Protocol* for transport, and it is characterized by both low throughput and low latency. Because Internet firewalls typically filter User Datagram Protocol traffic as it traverses from gateways to servers, the use of *Transmission Control Protocol* (TCP) encapsulation becomes a viable alternative. This solution, however, is negatively affected by network packet loss that, due to TCP inherent retransmissions, severely degrades latency, reducing system responsibility. In this paper, we analyze the effect of TCP-encapsulated CoAP and propose a mechanism that overcomes these limitations without having to change network topologies or modifying protocol functionality.

1 | INTRODUCTION

The standardization of the *Constrained Application Protocol* (CoAP) via Request for Comments (RFC)[1] has enabled an efficient *representational state transfer* Application Program Interface (API)-based interaction between applications and sensors. In this context, the transmission relies on connectionless *User Datagram Protocol* (UDP) transport that is highly efficient at delivering frames by minimizing the overall end-to-end latency. Restrictive access networks like the public Internet, however, use firewalls that block this type of traffic and only allow the traversal of web-friendly *Transmission Control Protocol* (TCP) transported frames. Therefore, an alternative to accomplishing a successful transmission of sensor data in an environment where datagrams are dropped is by means of *tunneling*.

Tunneling relies on encapsulating complete sensor data packets, including network layers, on top of a, typically TCP or stream based, firewall-friendly transport that requires no network topology changes. Although it is possible to alternatively use UDP-based encapsulation either by masking UDP as TCP[2] or by using well-known *Domain Name Server* UDP ports,[3] recently developed improvements in *Deep Packet Inspection* implemented at firewalls render these mechanisms useless.

As shown in Figure 1, access side traffic is generated by sensors and transmitted on top of CoAP at *layer 5* (L5). CoAP is, in turn, transported on top of UDP at *layer 4* (L4) and then packetized into IPv6 datagrams that are compressed and fragmented by means of *low-power wireless personal area networks* (6LoWPAN) at *layer 3* (L3). This provides adaptation for transmission over a low-rate wireless IEEE 802.15.4 infrastructure at *layers 1 and 2* (L1/2). Access traffic eventually reaches a gateway or cluster head that provides core connectivity to the public Internet. In order to provide firewall traversal capabilities, sensor packets become encapsulated as inner traffic that is streamed at L5 on top of the outer TCP at L4 over IPv6 at L3. These 2 layers, L3 and L4, are removed when the packets arrive at the tunnel

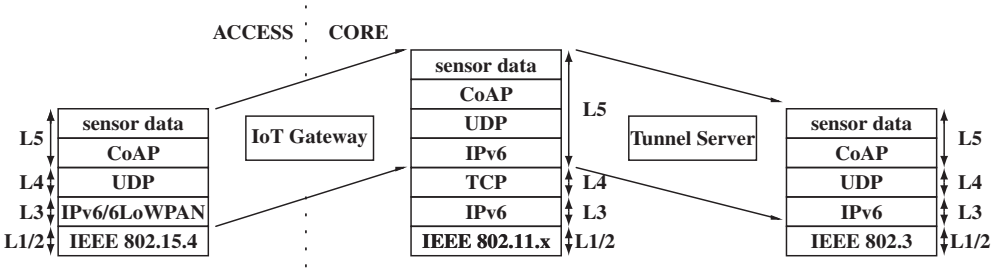

FIGURE 1 Sensor data encapsulation. CoAP, Constrained Application Protocol; IoT, Internet of Things; IPv6/6LoWPAN, IPv6 datagrams compressed and fragmented by low-power wireless personal area networks; L1/2, Layers 1 and 2; L3, Layer 3; L4, Layer 4; L5, Layer 5; TCP, Transmission Control Protocol; UDP, User Datagram Protocol

server that decapsulates the original L3 and L4 layers and provides routing to the CoAP application that consumes the sensor data.

Transporting sensor data over a stream, being TCP a protocol that relies on an *Automatic Repeat reQuest* (ARQ) mechanism, causes extra latency that is introduced by retransmissions that occur when frames are lost. Since these retransmissions are related to the *round-trip time* moving average calculated by taking sample estimates, the nature of the latency is random but typically large enough to disrupt functionality. Specifically, in a *real-time IoT* (RTIoT) environment, servers rely on application-layer *jitter buffers* that drop any packet that arrives too late to be of significance in the decision-making process. Moreover, the effect of latency is not the same for all applications; an application where *unmanned aerial vehicles* have their flightpath dynamically computed based on sensed *hyperspectral images*[4] has different latency requirements than an *urban agriculture* application[5] where excess water is dynamically drained based on rain levels. In this paper, we focus on critical applications where the allowable latency is 150 milliseconds at most. In this scenario, if the TCP-induced latency is above the aforementioned threshold, datagrams are dropped by the application layer, and decision making can be compromised, leading to potential physical loss.

In order to better understand the problem, we first introduce a mathematical model of TCP-encapsulated CoAP traffic to estimate the overall application-layer packet loss measured at the output of the jitter buffer. We then propose and mathematically model a solution that relies on dynamic encapsulation over multiple streams to minimize latency. The performance and efficiency of the proposed mechanism is then evaluated through an experimental framework. Note that the novel analytical model provides closed-form expressions that can be used to dynamically estimate the quality affecting the application.

The remainder of this paper is organized as follows. A review of relevant related works as well as a description of the motivation behind this paper are presented in Section 2. Details of the analytical model and a description of the proposed mechanism are introduced in Section 3. In Section 4, a description of the evaluation framework as well as comparative results obtained by applying network impairments and computing quality scores are detailed. Conclusions and future work are provided in Section 5.

2 | RELATED WORK AND MOTIVATION

The performance of CoAP has been studied from both the experimental and theoretical perspectives. Specifically in the work of Thombre et al,[6] the authors measure latency and loss in a CoAP-based wireless sensor network where multihop routing is taken into account. In the work of Collina et al,[7] the authors compare the congestion control mechanisms introduced by CoAP as well as other *Internet of Things* (IoT) protocols like *Message Queueing Telemetry Transport* (MQTT) and measure latency and loss and throughput for different network conditions. In the work of Chen and Kunz,[8] latency, loss, and throughput are measured in a highly degraded wireless network where both CoAP and MQTT performances are compared. In the work of Slabicki and Grochla,[9] CoAP is compared against other main stream data management protocols like the *Simple Network Management Protocol* to experimentally measure their latencies by means of histograms.

Encapsulation of IoT traffic is a fairly recent topic that has been addressed in the work of Esaki and Nakamura,[10] where an overall analysis of tunneling in the context of IoT is presented. In addition, details of encapsulation of CoAP and MQTT connected health sensor data are analyzed in the work of Karamitsios and Orphanoudakis.[11] In general, RTIoT has similar quality demands as those of *real-time communications* (RTC); hence, an understanding of encapsulation in the context

of RTC is also relevant. The use of TCP for RTC has been addressed mostly by the proposal of proprietary methods that provide framing to media over TCP. Packetization of speech under TCP for several traffic types is evaluated in the work of Sanchez-Iborra et al.[12] In the work of Brosh et al,[13] the authors model latency as a function of network loss, and they validate the model through experimental analysis. In the work of Hwang et al,[14] both TCP and UDP transport of different traffic types. The effects of TCP transport for real-time media are experimentally analyzed in the work of Cocker et al[15] with a special focus on buffer sizes. In the work of Satoda et al,[16] the authors propose an experimental method, that relying on multiplexing packets over several TCP connections, improves traffic performance. This mechanism is simulated for multiple traffic conditions such that values of packet loss and latency are obtained.

The material introduced in this paper extends the analysis of encapsulation in the context of IoT by presenting a novel mathematical model that can be used to analyze the effect of the number of streams on the application-layer *quality of service* (QoS). Specifically, by measuring the network-layer packet loss, it is possible to estimate the more complex behavior of encapsulation at the application layer in order to dynamically adjust the configuration of the tunnel and guarantee QoS goals. Indeed, the main motivation for this approach is to define a mechanism that minimizes both application packet loss and latency of encapsulated traffic and, therefore, improves the reliability of sensor and actuation data transmission over low-power low-rate IoT networks.

3 | THEORETICAL FRAMEWORK

One of the characteristics of wireless RTIoT networks is that they are affected by *dynamic multipath fading* that is typically modeled by means of a 2-state Markov process.[17-20] Essentially, the channel is either in a *low loss* (L) or in a *high loss* (H) state, as shown in Figure 2. As expected, when comparing packet loss in both states, that in the low-loss state is much lower than that in the high-loss state.

Four parameters characterize the Markov process, specifically: (1) p, the transition probability from the low-loss to the high-loss state; (2) α, the probability that the channel remains in the high-loss state; (3) β, the packet loss probability when the channel is in the low-loss state; and (4) γ, the packet loss probability when the channel is in the high-loss state.

3.1 | CoAP encapsulation

In this paper, we focus on traditional *nonconfirmable* CoAP traffic that, not relying on individual frame acknowledgments, is more efficient in real-time scenarios. When encapsulated, CoAP frame transmission over TCP relies on making sure that the TCP window is negotiated to guarantee that each packet is transported on top of a single TCP frame. In this case, retransmissions typically result, as shown in Figure 3, from loss introduced by network impairments.

Since the minimum TCP *retransmission timeout*, as indicated in the work of Psaras and Tsaoussidis,[21] is about $\Delta = 300$ ms, retransmissions are responsible for the application-layer packet. Figure 4 shows an example where a sensor produces 7 periodic CoAP frames that are encapsulated in order to traverse a firewall. Specifically, network packet loss causes frames #2 and #3 to be dropped while the transmission of frame #1 is still in progress. The resulting application-layer packet loss is therefore $\frac{2}{7}$ or about 28%.

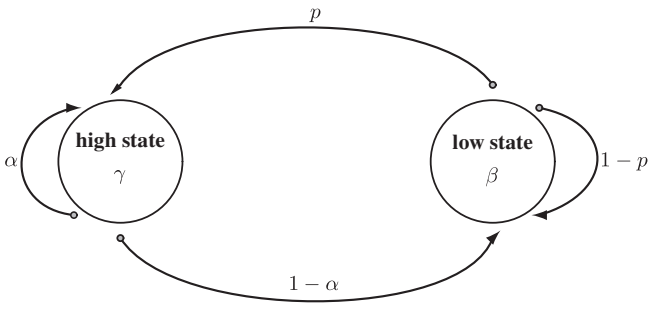

FIGURE 2 Two-state Markov model

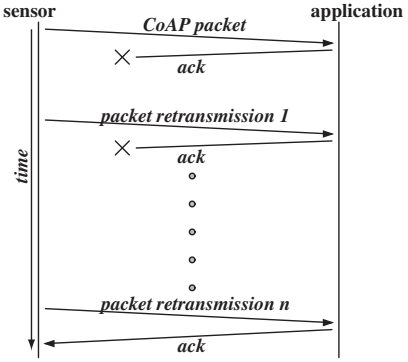

FIGURE 3 Encapsulated Constrained Application Protocol (CoAP) frame transmission

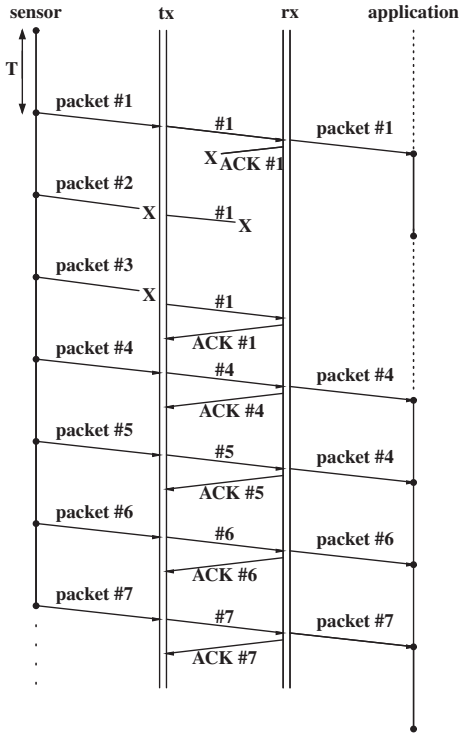

FIGURE 4 Encapsulated Constrained Application Protocol traffic

In this paper, we apply an intuitive approach that consists in multiplexing encapsulated traffic over multiple streams. By parallelizing traffic, as shown in Figures 5 and 6 for multiplexing and demultiplexing, respectively, it is possible to minimize the effect of the delay introduced by network-layer loss. Specifically, the Figures show 9 CoAP frames encapsulated over 3 streams where multiplexing occurs at the transmitter and demultiplexing at the receiver for a given direction.

A single-stream transport can be modeled as an ARQ mechanism where sent frames are acknowledged on an individual basis and retransmissions occur at a fixed period of time whenever a previous transaction fails due to loss, as shown in Figure 3. In general, the probability of a successful transaction results from combining a successful transmission with a successful acknowledgment. Specifically, this probability is defined as $P_{\text{ack,trans}}$ and given by

$$P_{\text{ack,trans}} = P_{\text{ack|trans,low}} P_{\text{trans|low}} P_{\text{low}} + P_{\text{ack|trans,high}} P_{\text{trans|high}} P_{\text{high}}, \tag{1}$$

where $P_{\text{ack|trans,low}}$ and $P_{\text{ack|trans,high}}$ are the conditional probabilities of successful acknowledgment given a successful transmission for a channel in the low-loss and high-loss states, respectively; $P_{\text{trans|low}}$ and $P_{\text{trans|high}}$ are the conditional probabilities of successful transmission for a channel in the low-loss and high-loss states, respectively; and P_{low} and P_{high} indicate the state probabilities of the channel.

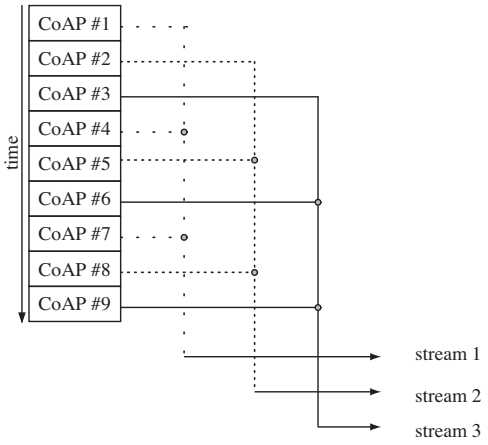

FIGURE 5 Multistream encapsulation. CoAP, Constrained Application Protocol

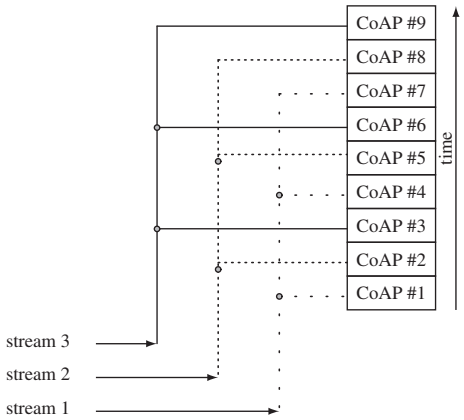

FIGURE 6 Multistream decapsulation. CoAP, Constrained Application Protocol

Assuming behavioral symmetry and independence on both directions, ie, the probability of successful acknowledgment equal to the probability of successful transmission for a given channel state, the conditional probabilities of successful acknowledgment given a successful transmission are as follows:

$$P_{\text{ack|trans,low}} = P_{\text{trans|low}} P_{\text{low}\to\text{low}} + P_{\text{trans|high}} P_{\text{low}\to\text{high}} \tag{2}$$

and

$$P_{\text{ack|trans,high}} = P_{\text{trans|low}} P_{\text{high}\to\text{low}} + P_{\text{trans|high}} P_{\text{high}\to\text{high}}, \tag{3}$$

where $P_{\text{low}\to\text{low}}$, $P_{\text{low}\to\text{high}}$, $P_{\text{high}\to\text{low}}$, and $P_{\text{high}\to\text{high}}$ are all possible transition probabilities of the channel switching from one loss state to another.

From the Markov process, $P_{\text{trans|low}} = 1-\beta$, $P_{\text{trans|high}} = 1-\gamma$, $P_{\text{low}\to\text{low}} = 1-p$, $P_{\text{low}\to\text{high}} = p$, $P_{\text{high}\to\text{low}} = 1-\alpha$, and $P_{\text{high}\to\text{high}} = \alpha$, and the steady-state probabilities[22] are given by $P_{\text{low}} = \frac{1-\alpha}{1-\alpha+p}$ and $P_{\text{high}} = \frac{p}{1-\alpha+p}$.

When replacing all these expressions in Equation 1, we then have

$$P_{\text{ack,trans}} = \frac{1-\alpha}{1-\alpha+p}\left[(1-\beta)\left[(1-\beta)(1-p)+(1-\gamma)p\right]\right] + \frac{p}{1-\alpha+p}\left[(1-\gamma)\left[(1-\gamma)\alpha+(1-\beta)(1-\alpha)\right]\right]. \tag{4}$$

In general, retransmissions occur until frames are successfully transmitted and acknowledged such that the overall probability of success after $R = k$ retransmissions of a single frame follows a *geometric distribution* with a *probability mass function* given by

$$P_{\text{suc}}(R=k) = P_{\text{ack,trans}}(1-P_{\text{ack,trans}})^k. \tag{5}$$

Similarly, the expected number of retransmissions per single frame results from calculating the expectation of the geometric distribution by means of

$$E(R) = \sum_{k=0}^{\infty} kP_{\text{suc}}(R = k) = \frac{1 - P_{\text{ack,trans}}}{P_{\text{ack,trans}}}.$$

Figure 4 shows how CoAP frames, sensed at a rate of 1 every T seconds, are transmitted. If a frame is sent while the retransmission of a previous one is still in progress, the frame is dropped. This means that if the average time between retransmissions is Δ, the number of lost frames due to the retransmission of a frame is $E(R)\frac{\Delta}{T}$.

Now, if the loss probability of a single frame is given by $P_{\text{frame}}(\text{loss})$, then, when retransmissions are put into consideration, the overall application-layer packet loss probability for transport over a single tunnel defined as $P_{\text{encap}}(\text{loss})$ and measured at the output of the jitter buffer is given by

$$P_{\text{encap}}(\text{loss}) \leq \min\left(P_{\text{frame}}(\text{loss})\left[1 + E(R)\frac{\Delta}{T}\right], 1\right), \tag{6}$$

where the upper bound results from the fact that losses occurring during retransmissions may overlap with single-frame losses. Note that the unidirectional single-frame probability of loss, $P_{\text{frame}}(\text{loss})$, can also be obtained out of the model and results as

$$P_{\text{frame}}(\text{loss}) = 1 - \left[P_{\text{trans|low}}P_{\text{low}} + P_{\text{trans|high}}P_{\text{high}}\right], \tag{7}$$

which, when replacing with the corresponding quantities obtained from the Markov model, is further simplified as

$$P_{\text{frame}}(\text{loss}) = 1 - (1 - \beta)\frac{1 - \alpha}{1 - \alpha + p} - (1 - \gamma)\frac{p}{1 - \alpha + p}, \tag{8}$$

where for the extreme case of no network impairments such that $p = 0$ and $\beta = 0$, then $P_{\text{frame}}(\text{loss}) = 0$ and $P_{\text{str}}(\text{loss}) = 0$.

3.2 | Multiple CoAP encapsulation

The main problem with single-stream transport is that CoAP frames are dropped if a previous frame transmission is still in progress. The multiplexing scheme shown in Figure 5 attempts to solve this problem by relying on simultaneous transmission over multiple streams. For example, Figure 7 shows 3 connections ($N = 3$), such that frames #1, #4, and #7, frames #2 and #5, and frames #3 and #6 are sent over the first, second, and third streams, respectively. As in the single-stream case shown in Figure 4, frame #1 fails to be acknowledged right away due to network-layer impairments, but

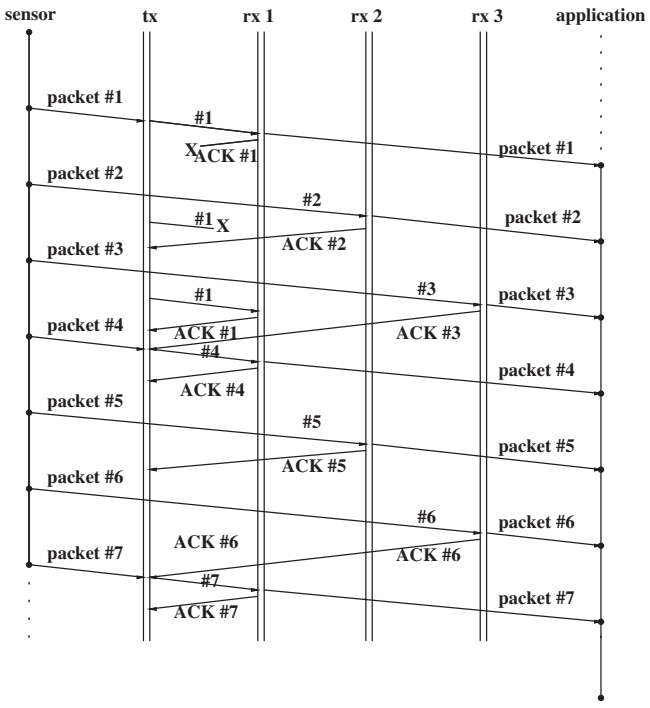

FIGURE 7 Multistream encapsulated Constrained Application Protocol traffic

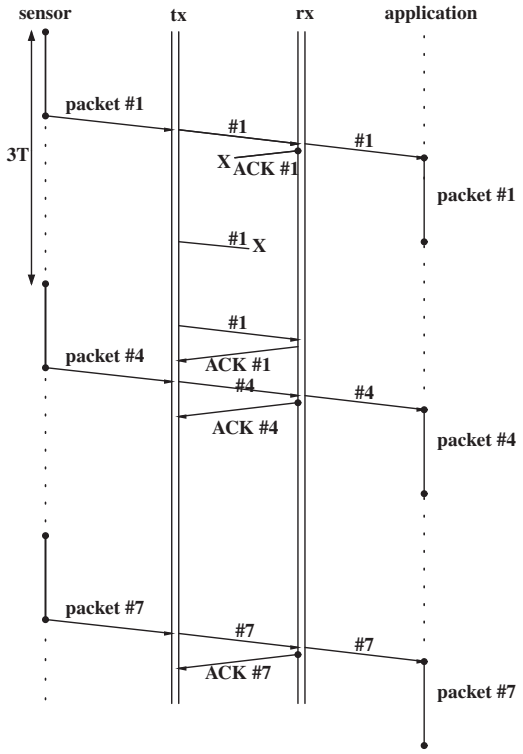

FIGURE 8 Multistream encapsulated Constrained Application Protocol traffic (single stream)

in this scenario, this delay has no effect on frames #2 and #3 that successfully arrive at the destination. Consequently and when relying on multiple-stream encapsulation, network conditions that would otherwise cause application-layer loss have no impact on quality. Specifically, as opposed to the single-stream case, no application-layer packet loss is experienced under this scheme.

Figure 8 shows the transmission over the first stream alone, where it can be seen that CoAP frames are effectively transmitted every NT seconds. Because the effect of relying on N streams for multiplexed transport is analogous to increasing the interframe periodicity from T to NT, Equation (6) can be modified accordingly. Specifically, the application packet loss defined as P_{mencap} is given by

$$P_{\text{mencap}}(\text{loss}) \leq \min\left(P_{\text{frame}}(\text{loss})\left[1 + E(R)\frac{\Delta}{NT}\right], 1\right) \tag{9}$$

when multiple streams are used for transport.

The Markov model can be further simplified by assuming that all packets are dropped at the high-loss state and no packets are dropped at the low-loss state. In this case, $\beta = 0$ and $\gamma = 1$, and therefore, Equation (8) becomes

$$P_{\text{frame}}(\text{loss}) = \frac{p}{1 - \alpha + p}, \tag{10}$$

and consequently, Equation (9) results as

$$P_{\text{mencap}}(\text{loss}) \leq \min\left(\frac{p}{1 - \alpha + p}\left[1 + \left(\frac{2p - \alpha p}{1 - \alpha - p + \alpha p}\frac{\Delta}{NT}\right)\right], 1\right). \tag{11}$$

Note that creating an infinite number of streams ($N \to \infty$) is equivalent to creating a stream per frame to be transmitted, and therefore, the probability $P_{\text{mencap}}(\text{loss})$ is given by

$$P_{\text{mencap}}(\text{loss}) \leq \lim_{N\to\infty} \min\left(P_{\text{frame}}(\text{loss})\left[1 + E(R)\frac{\Delta}{NT}\right], 1\right) \leq \min\left(P_{\text{frame}}(\text{loss}), 1\right) = P_{\text{frame}}(\text{loss}), \tag{12}$$

implying that stream transport can have loss probability that is as low as the loss probability of datagram transport as long as the number of streams is sufficiently large. Specifically, for a CoAP frame size of $T = 20$ ms, Figure 9 shows the application packet loss as a function of the number of streams N and the network loss probability parameters p and α. These plots assume $\Delta = 300$ ms, a value that complies with the minimum TCP RTO.[21]

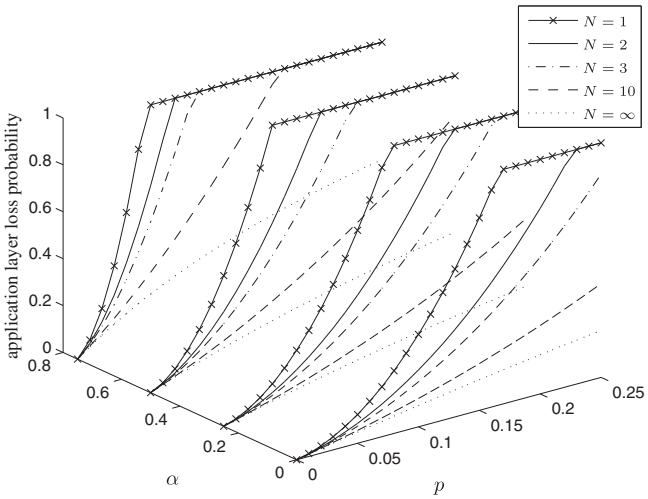

FIGURE 9 Loss probability vs p vs α

4 | EXPERIMENTAL FRAMEWORK

Figure 10 shows the experimental framework that is used to evaluate the performance of CoAP encapsulation that relies on transport by means of multiple streams. Specifically, 20 independent temperature sensors ($K = 20$) transmit temperature readouts at a rate of 50 per second. The sensors are initialized with a 1-millisecond delay difference that guarantees that not all sensors transmit simultaneously in a busty fashion. The mechanism relies on the Internet Engineering Task Force RFC 7641 "Observing Resources in the Constrained Application Protocol,"[23] which allows CoAP to support unidirectional sensor observation. Each experiment is executed for 300 seconds, and packet loss computation is performed at the output of the jitter buffer on the application. The protocol stacks for access and core networks are as shown in Figure 1. All network elements in the framework (sensors, gateway, and application) are emulated by means of *Visual ProtoStack*.[24] This emulator is also used to introduce controlled network packet loss on the core network in accordance with the channel model presented in Section 3. For quick reference, Table 1 shows a description of each of the variable parameters involved in this experimental framework.

Application-layer packet losses, experimentally obtained at the output of the dynamic jitter buffer as well as theoretical ones, are shown in Figures 11 and 12 for low ($\alpha = 0.3$) and high ($\alpha = 0.9$) network bursty packet losses, respectively. The plots show results for single-stream ($N = 1$) and multiple-stream encapsulation ($N = 3, 10$) when the CoAP interpacket transmission period is 20 milliseconds long. Note that in all cases, as the number of streams increases, packet loss as seen

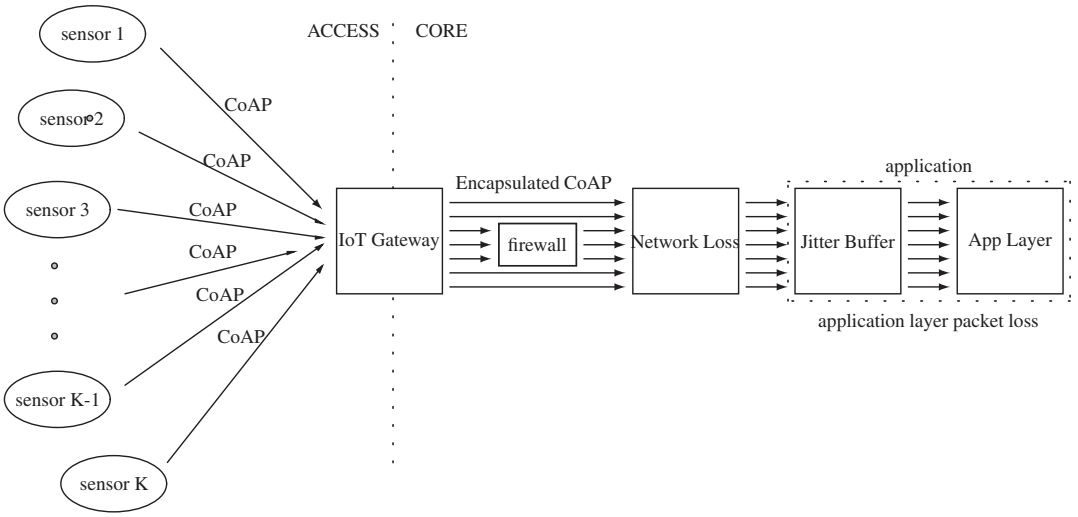

FIGURE 10 Experimental framework. CoAP, Constrained Application Protocol; IoT, Internet of Things

TABLE 1 Summary of parameters

Parameter	Description
α	Network packet loss burstiness
β	Network packet loss probability
K	Number of sensors
N	Number of encapsulation streams

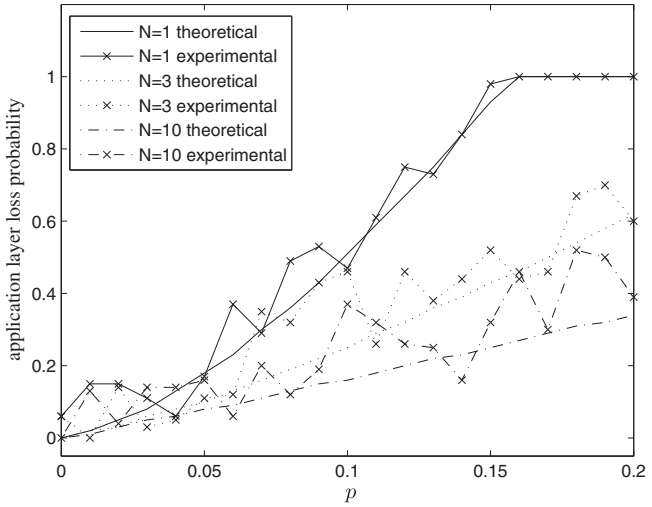

FIGURE 11 Application loss probability ($\alpha = 0.3$)

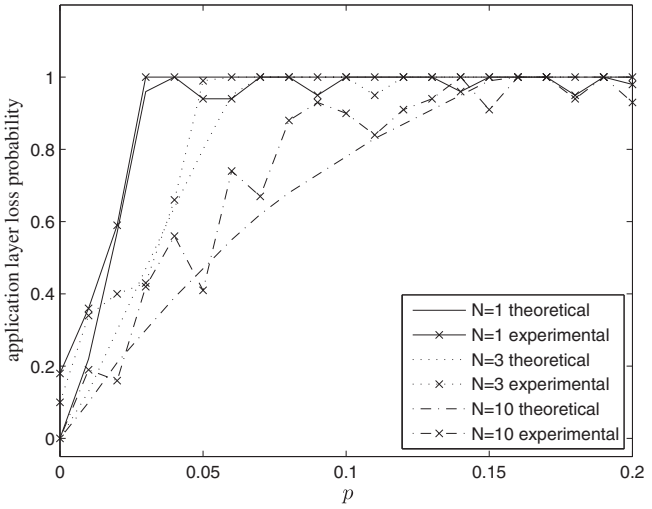

FIGURE 12 Application loss probability ($\alpha = 0.9$)

by the application lowers accordingly as now there are multiple paths for packets to traverse. Moreover, this improvement is present regardless of the burstiness of the packet loss experienced at the network layer.

Similarly, Figures 13 and 14 show the experimental mean latency for low ($\alpha = 0.3$) and high ($\alpha = 0.9$) network bursty packet losses, respectively. Again, each plot shows the delay for regular ($N = 1$) as well as multiple-stream encapsulation ($N = 3, 10$). When more streams are used for a single session, fewer packets are sent on every single stream, leading to less congestion and, therefore, lower latency.

When considering different sensor loads, Table 2 shows the average application packet loss when encapsulating traffic over 10 streams ($N = 10$) for low bursty network packet loss. Two parameters are taken into account, ie, the sensor population K, now ranging between 5 and 100 sensors, and the transmission rate fixed at 50 *samples per second* (sps) or 10 sps. For different sensor populations, different transmission rates are selected on a proportion given by the ratio

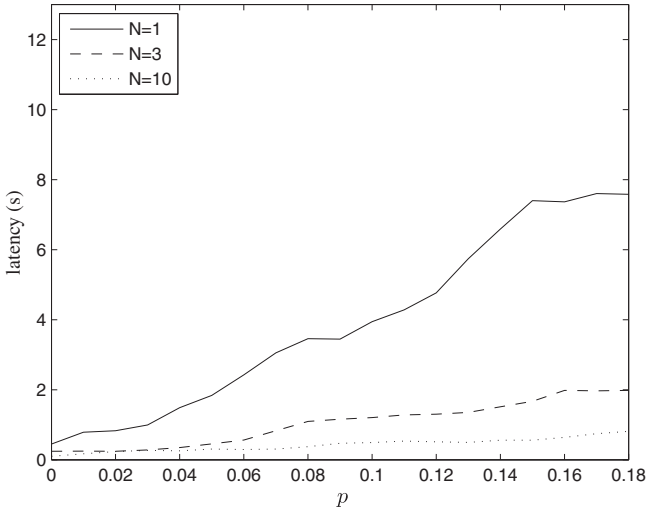

FIGURE 13 Experimental latency ($\alpha = 0.3$)

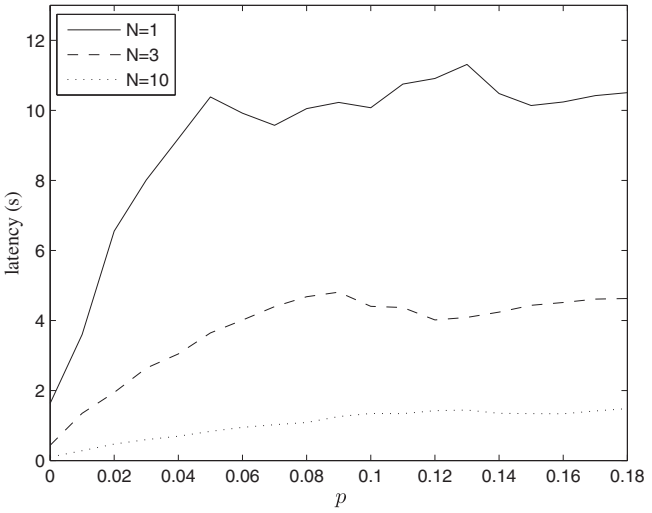

FIGURE 14 Experimental latency ($\alpha = 0.9$)

TABLE 2 Experimental application packet loss vs sensor load

K	Transmission Rate Distribution [50 sps][10 sps]				
	[1][0]	[2/3][1/3]	[1/2][1/2]	[1/3][2/3]	**[0][1]**
5	13.81%	13.76%	13.71%	13.70%	13.59%
20	14.72%	14.51%	14.51%	14.27%	13.78%
50	32.88%	22.17%	22.85%	16.51%	14.16%
100	85.69%	65.18%	40.53%	38.87%	15.10%

between one rate or the other. The limiting factor in all cases is the IEEE 802.15.4 contention that translates into higher application packet loss as population and transmission rates increase. For small sensor populations ($K \leq 20$) and regardless of the sensor readout sample rate, application packet loss is fairly uniform.

Table 3 shows the average absolute error obtained when comparing experimental against theoretical, obtained from the model in Section 3, application-layer packet loss. The error is presented for single- and multiple-stream encapsulated CoAP traffic as well as for low and high network-layer packet losses. It can be seen that in all cases, the estimation error introduced by the model is below 10%. This model, therefore, can be used to estimate the impact of wireless network-layer packet loss at the application layer. The idea is to find the α and p parameters of the Markov process and then predict the

TABLE 3 Application packet loss estimation error

Stream Count	Network Loss	
	Low	High
1	4.71%	3.14%
3	7.91%	3.61%
10	8.42%	7.57%

TABLE 4 Experimental application packet loss and latency reduction

Stream Count	Network Loss		Latency	
	Low	High	Low	High
3	23.23%	7.23%	72.64%	86.79%
10	33.09%	17.14%	59.22%	87.59%

application-layer loss to determine QoS and assess the number of streams under which encapsulation must be performed in order to reach QoS goals.

Table 4 shows the experimental relative application packet loss and latency reduction that results from introducing multiple-stream encapsulation. For both low and high network-layer packet losses, the reduction is about 10% higher when using 10 streams as opposed to 3. Similarly, for latency, its reduction is between 13% and 25% depending on the case. In general and in order to improve the application-layer performance, as indicated in the previous paragraph, the mathematical model can be used to estimate the minimum number of streams needed to reach a specific QoS goal.

5 | CONCLUSION AND FUTURE WORK

CoAP traffic relies on UDP-based transport that minimizes latency, but it is typically blocked by firewalls. In this paper, we have proposed a mechanism that overcomes this problem through TCP encapsulation. This, however, degrades latency as network packet loss results in transport layer retransmissions that delay packet delivery. In order to mitigate this effect, we have extended CoAP encapsulation to perform it over multiple TCP streams. Experimental results show that this mechanism reduces application-layer packet loss by an amount between 7% and 33% depending on network conditions and the number of streams under consideration. This lower application-layer packet loss is tied to lower latency as packets arriving faster tend to survive the delay restrictions imposed by playout buffers. This is particularly important since streams are TCP based and TCP natively converts network packet loss into transport delay due to retransmissions. In addition, the efficiency of this approach is mathematically modeled and can be used, in turn, to assess the application-layer quality. Moreover, based on this model, an application can dynamically select the minimum number of encapsulation streams, and therefore overhead, that would guarantee a given QoS goal. Note that under this approach, however, the analytical model fails to provide a closed-form expression of packet latency that can be used to better understand the effects of network impairments on the application layer. Another limitation of the mathematical model presented in this paper is that it assumes no contention at media access.

As an area of future work, the mathematical model presented in this paper can be further extended to quantify application QoS through a *Quality Score* that provides to IoT applications what a *Mean Opinion Score* provides to RTC applications. This involves, among other considerations, closed-form expressions for the estimation of latency.

ORCID

Rolando Herrero http://orcid.org/0000-0001-8389-270X

REFERENCES

1. Bormann C, Hartke K, Shelby Z. The constrained application protocol (CoAP). RFC 7252; 2015. https://rfc-editor.org/rfc/rfc7252.txt
2. Herrero R, Katz H, Deng M. Concealed datagram-based tunnel for real-time communications. US patent application 14/657,227. 2016. https://encrypted.google.com/patents/US20160269285

3. Mahalingam M, Dutt D, Duda K, et al. Virtual extensible local area network (VXLAN): a framework for overlaying virtualized layer 2 networks over layer 3 networks. RFC 7348; 2014. https://rfc-editor.org/rfc/rfc7348.txt
4. Herrero R, Cadirola M, Ingle VK. Preprocessing and compression of hyperspectral images captured onboard UAVs. In: *Unmanned/Unattended Sensors and Sensor Networks XI; and Advanced Free-Space Optical Communication Techniques and Applications*. Bellingham, WA: SPIE; 2015. https://doi.org/10.1117/12.2186169
5. Sivamani S, Bae N, Cho Y. A smart service model based on ubiquitous sensor networks using vertical farm ontology. *Int J Distrib Sens Netw*. 2013;9. https://doi.org/10.1155/2013/161495
6. Thombre S, Islam RU, Andersson K, Hossain MS. Performance analysis of an IP based protocol stack for WSNs. Paper presented at: 2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS); 2016; San Francisco, CA.
7. Collina M, Bartolucci M, Vanelli-Coralli A, Corazza GE. Internet of things application layer protocol analysis over error and delay prone links. Paper presented at: 2014 7th Advanced Satellite Multimedia Systems Conference and the 13th Signal Processing for Space Communications Workshop (ASMS/SPSC); 2014; Livorno, Italy.
8. Chen Y, Kunz T. Performance evaluation of IoT protocols under a constrained wireless access network. Paper presented at: 2016 International Conference on Selected Topics in Mobile and Wireless Networking (MoWNeT); 2016; Cairo, Egypt.
9. Slabicki M, Grochla K. Performance evaluation of CoAP, SNMP and NETCONF protocols in fog computing architecture. Paper presented at: IEEE/IFIP Network Operations and Management Symposium; 2016. https://doi.org/10.1109/NOMS.2016.7503010
10. Esaki H, Nakamura R. Overlaying and slicing for IoT era based on internet's end-to-end discipline. Paper presented at: 2017 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN); 2017; Osaka, Japan.
11. Karamitsios K, Orphanoudakis T. Efficient IoT data aggregation for connected health applications. Paper presented at: 2017 IEEE Symposium on Computers and Communications (ISCC); 2017; Heraklion, Greece.
12. Sanchez-Iborra R, Cano MD, Garcia-Haro J. Performance evaluation of QoE in VoIP traffic under fading channels. Paper presented at: 2013 World Congress on Computer and Information Technology (WCCIT); 2013; Sousse, Tunisia.
13. Brosh E, Baset SA, Misra V, Rubenstein D, Schulzrinne H. The delay-friendliness of TCP for real-time traffic. *IEEE/ACM Trans Netw*. 2010;18(5):1478-1491.
14. Hwang H, Yin X, Wang Z, Wang H. The internet measurement of VoIP on different transport layer rotocols. Paper presented at: 2009 International Conference on Information Networking; 2009; Chiang Mai, Thailand.
15. Cocker E, Ghazzi F, Speidel U, Dong MC. Measurement of buffer requirement trends for real time traffic over TCP. Paper presented at: 2014 IEEE 15th International Conference on High Performance Switching and Routing (HPSR); 2014; Vancouver, Canada.
16. Satoda K, Nihei K, Yoshida H. Quality evaluation of voice over multiple TCP connections. Paper presented at: 2014 International Conference on Computing, Networking and Communications (ICNC); 2014; Honolulu, HI.
17. Herrero R, Cadirola M. Effect of FEC mechanisms in the performance of low bit rate codecs in lossy mobile environments. Paper presented at: 2014 Conference on Principles, Systems and Applications of IP Telecommunications (IPTComm'14); 2014; Chicago, IL.
18. Herrero R, St-Pierre C. Analytical model of stream transported media. *IET Networks*. 2017;6(1):14-21.
19. Herrero R. Modeling and comparative analysis of forward error correction in the context of multipath redundancy. *Telecommunication Systems*. 2017;65(4):783-794.
20. Herrero R. Integrating HEC with circuit breakers and multipath RTP to improve RTC media quality. *Telecommunication Systems*. 2017;64(1):211-221.
21. Psaras I, Tsaoussidis V. The TCP minimum RTO revisited. In: Akyildiz IF, Sivakumar R, Ekici E, de Oliveira JC, McNair J, eds. *Networking*. *Lecture Notes in Computer Science*, vol. 4479. Berlin, Germany: Springer; 2007:981-991.
22. Barton M, Lemberg H, Sarraf M, Hamilton C. Performance analysis of packet loss concealment in mobile environments with a two-state loss model. Paper presented at: 2010 IEEE International Workshop Technical Committee on Communications Quality and Reliability (CQR); 2010; Vancouver, Canada.
23. Hartke K. Observing resources in the constrained application protocol (CoAP). RFC 7641; 2015. https://rfc-editor.org/rfc/rfc7641.txt
24. VPS. Visual ProtoStack Protocol Emulator. http://www.vprotostack.com

How to cite this article: Herrero R. Encapsulation of real-time IoT CoAP traffic. *Trans Emerging Tel Tech*. 2018;e3287. https://doi.org/10.1002/ett.3287