Performance evaluation and improvement of congestion control of the constrained application protocol for the Internet-of-things

Nabil Makarem

Sorbonne University - Lebanese University
07 December 2021

Performance Evaluation and Improvement of Congestion Control of the Constrained Application Protocol for the Internet-of-Things

Presented
by
Nabil Makarem

for obtaining
In Partial Fulfillment of the requirements for the degree of
Doctor of Philosophy in Computer Science

Committee:

Abdallah Kassem	Reviewer	Professor, Notre Dame University
Nathalie Mitton	Reviewer	Research Director, INRIA
Noël Crespi	Examiner	Professor, Institut Polytechnique de Paris
Abdulmotaleb El Saddik	Examiner	Professor, University of Ottawa
Anne Fladenmuller	Examiner	Professor, Sorbonne Université
Naceur Malouch	Supervisor	Associate Professor (HDR), Sorbonne Université
Imad Mougharbel	Supervisor	Professor, Lebanese University
Wafaa Bou Diab	Co-Supervisor	Assistant Professor, Lebanese University

"On ne pourra bien dessiner le simple qu'après
une étude approfondie du complexe"
– Gaston Bachelard

Acknowledgments

A life time dream of mine became a reality with the kind support of many people around me. I don't know how to express my sincere gratitude to everyone in a few lines, but a heart full of gratefulness is indebted to everyone that was there for me during my challenging times. Foremost, let me offer my endeavor to the creator of all miracles, God for the wisdom HE bestowed upon me, the strength, the patience and peace of mind to finish this research.

"In all your ways acknowledge HIM and HE will make your paths straight." – Proverbs 3:6

I would like to express my sincere gratitude to my supervisor and professor Naceur Malouch who was a fatherly figure to me during this PhD. It was a great pleasure to do my research under his direction. Thank you for all the guidance, constant supervision, and moral support in completing this endeavor.

I would like also to express my gratitude to my supervisor Professor Imad Mogharbil and Co-supervisor Dr. Wafaa Abou Diab. I would like to express my warmest thanks to Professors Nathalie Mitton, and Abdallah Kassem for their valuable time evaluating and reporting this thesis. I would also like to thank my committee members who honored me in being part of this research.

A special thanks to my wife who was my rock , to my beloved family who served as my inspiration to pursue this mission. A special thanks to Lady Samira and Mr. Bu Dargham who sustained me this far with their kind thoughts and prayers.

The completion of this study couldn't have been possible without professor Nasser Ballani's guidance and Mr. Ayoub Saab who was there for me in the hardest of times. Thank you!

I dedicate this thesis with a million bows of respect to a special soul that left us early our dear Nour Nouaihid and to cancer children around the world who are the greatest heroes and the youngest philosophers. I was honored to know many of them and they touched my heart and soul in every aspect giving me the will and power to proceed.

I hope this PhD is the start of a journey full of knowledge, learning, and achievements. I aspire to be as helpful to others as those kind people who have helped me reach where I am today and to contribute to a better future for people around the world.

Abstract

The Constrained Application Protocol (CoAP) is a lightweight core protocol designed by the Internet Engineering Task Force (IETF) used for communication between devices in the Internet of Things (IoT). Congestion control is substantial to overcome the limitations of such devices and the restrictions imposed by the connecting networks. The CoAP standard defines a simple congestion control mechanism based mainly on retransmissions after timeouts and a binary exponential back-off procedure. However, this simple protocol with its default parameters is not sufficient and can dramatically affect the efficiency of CoAP. Several advanced mechanisms for CoAP were suggested by the literature. Some considered improving retransmission timeout estimation whereas others focused on augmenting the retransmission procedure. The improvements in the second part are twofold: The first is Backoff-based such as CoCoA+ and pCoCoA, while the second is rate-based such as BDP-CoAP. We will perform critical analysis of these works and highlight their shortcomings and pitfalls, then present and evaluate our approach in different uses cases and network scenarios.

In this research work, we propose new exact mathematical models to analyze the performance of CoAP in lossy IoT networks. This study provides insights about improving CoAP congestion control in such networks and highlights the properties – including the limitations – of CoAP. Besides, we show that the simple control mechanism reduces significantly CoAP performance especially in terms of bandwidth utilization since it prevents the protocol from acting efficiently during congestion periods. We then propose new improvements in order to enhance the trade-off between reliability and goodput while keeping the algorithms reasonably simple for constrained devices. First, we optimize further the estimation procedure of the retransmission timeout in order to enhance congestion detection. Timeouts are the only indicator used in CoAP to detect losses, and losses are used as a strong indicator to detect congestion. Second, we replace the backoff algorithm by ”real” congestion control algorithms inspired from the well-known Additive Increase Multiplicative Decrease technique and a recent measurement-based congestion control called BBR. Our analysis using both our simulator and Contiki/Cooja environment show that the rate-based approach outperforms the backoff-based approach. Moreover, all the results show that our algorithms achieve a much better tradeoff between goodput, reliability and overhead.

Contents

List of Figures

1 Introduction

1.1 Context and Motivation

Driven by rapid technological advancements, where wired and wireless networks are ubiquitous, society is moving towards an always-connected state and has given birth to the era of the Internet-of-Things (IoT). IoT represents a vision of the world where billions of devices with embedded intelligence, communication means, and sensing capabilities such as washing machines, TVs, coffee machines and tiny objects like sensors are all connected to the Internet [46]. Fig. 1.1 illustrates an overview of the IoT concept in which objects from different domains are connected to the Internet. Moreover, objects in the IoT entity are uniquely identifiable, with a known position, status, and added intelligence services to benefit and expand the entity. Indeed, the integration of these smart objects with ubiquitous network connectivity, microscopic sensors, and cloud storage has created the Internet of Things. Besides integration, data storage, processing, and analysis are fundamental requirements, necessary to enrich the raw IoT data and transform them into useful information. Subsequently, IoT elements are becoming the objects of everyday life characterized by a diverse capacities and capabilities.

Recently, IoT has grabbed huge attention in almost all areas of scientific and industrial fields such as agriculture, manufacturing, homes, health care, disaster management and transportation. IoT applications include but not limited to:

Figure 1.1: The overall picture of the IoT emphasizing different domains

- Smart Home

  Home Automation [40] involves controlling and automating all the home's embedded technology. It is a residence where all its appliances such as air conditioners, washers, dryers, refrigerators, freezers, and camera systems are able to communicate with each other and remotely controlled via Internet. Using smart products provides time, energy and money savings. For instance, smart lightening systems adjust the lightening based on room occupation and daylight availability. Smart thermostats remotely monitor and control room temperatures. Smart security cameras using motion sensors detect different activities and can notify authorities if needed. One important key feature is the flexibility in infrastructure so that devices from different vendors can be easily integrated. Smart homes provide many benefits like comfort, energy efficiency, low operating costs, and less human efforts.

- Healthcare

  IoT as envisioned in healthcare systems is geared towards treatment of patients, in particular patients with chronic issues, disabled, and elderly people. To meet this end, many IoT applications have been proposed over the years in the healthcare field. Healthcare-based IoT is considered as an effective method aiming to enhance existing features of healthcare, increase convenience and the quality of life, and reduce costs. This aim is envisioned via the use of Remote Patient Monitoring Systems (RPMS), management of health and fitness programs, supervision of chronic diseases, children and elderly care, blood pressure and body temperature monitoring, monitoring of glucose and heart functioning, and medication management. For instance, RMPS [41] allows for real-time monitoring of patients and also provides data for management purposes (information, medical emergency management, etc.). Ambient Assisted Living (AAL) [58] addresses the health care of aging and incapacitated individuals. In addition, fitness accessories are now

suitable for smart devices to help individuals achieve their best shape. Thus, Healthcare-IoT has the potential of enhancing patient experience, improving medical workflows, and optimizing the use of resources [34].

- Disaster Management

  Disasters can be classified as natural which include volcanoes, forest-fires, floods, and earthquakes whereas the man-made mainly include terrorist attacks, urban and industrial disasters. Disasters have great impact on human life and industry. Being aware is better than being cured after a disaster, and IoT has proven to be capable to provide significant and efficient solutions to different problems for disaster management. Existing IoT approaches aggregate many events related to disasters such as notifications, data analysis, locating victims, and remote monitoring. For instance, the existing IoT applications explored in research deal with major aspects of various disasters including but not limited to managerial, monitoring, predictive and analysis. Although there are approaches that handle major issues in disaster incidents, further improvement and enhancement are still necessary in technological and design perspectives [51].

- Smart Grid

  Smart Grids are developed to replace traditional power grids for reliable and efficient energy service. This technology leverages distributed energy generators for many reasons, including the improvement of energy-utilization in these generators, the reduction of CO_2 emissions, and connecting customers with utility supplies via bi-directional networks. Integration of smart-grids and IoT technology in houses and/or buildings can achieve the desired connection by installing smart meters [36]. Smart metering is one of the significant applications in smart grids for environmental sustainability and energy issues recently. These advanced energy meters are used for energy consumption measurement, energy monitoring and interaction with supplies to provide real-time data including but not limited to supply and demand of energy. With this data, utility providers and customers get a better idea about energy consumption and can thus improve resource utilization and reduce energy-related costs [44].

- Smart Cities

  Another application of IoT is in the transportation industry, wherein vehicles are part of a communication network that allows for transportation management, control, and computing to obtain an intelligent transportation system. In this system, traffic data and perception data are used and computing is applied to get a higher safety in the transportation [44].

This widespread use of IoT applications in different areas and the growing number of physical objects connected to the Internet motivates international communities and organizations to increase their efforts in developing standards and protocols to respond to the evolution of the IoT. Besides, different

groups – led by Internet Engineering Task Force (IETF), World Wide Web Consortium (W_3C), European Telecommunications Standards Institute (ETSI), and the Institute of Electrical and Electronics Engineers (IEEE) – have been formed to provide and standardize protocols to support the IoT. Fig. 1.2 presents a summary of the most known protocols defined by these groups.

However, it is important to note that not all of these protocols have to packed together in a given IoT application. Moreover, some protocols, based on the type of an IoT application, may fail to support that application.

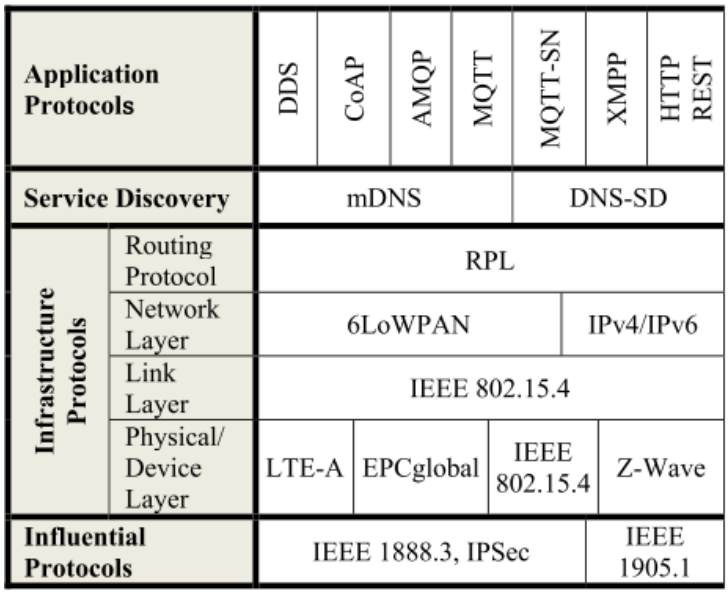


Figure 1.2: Summary of the most known IoT protocols

Application layer protocols are used to exchange data among endpoints of distributed applications. The importance of application layer protocols is that they provide rules for communication between objects, type and format of the data being exchanged, and methods for error notification. Perhaps the most important reason for the widespread adoption of the Internet is the design and development of application layer protocols. In the following, we provide an overview of some of the presented protocols in the application layer and their core functionality.

1. Message Queue Telemetry Transport (MQTT)
   MQTT [49] is a simple lightweight application layer protocol, which is used to collect data from remote sensors and send it to servers. MQTT is characterized by low power consumption, and efficient distribution of information to several receivers. It uses a publish-subscribe architecture. MQTT relies on TCP (Transport Control Protocol) so in network environments with high packet loss, retransmissions reduce the overall application latency [6]. In particular, TCP sends and receives additional packets to establish the connection before the data exchange, which fur-

ther increases the latency. In addition to the increased energy consumption, the incremental size also increases the risk of having errors during the transmission.

2. Advanced Message Queuing Protocol (AMQP)
   AMQP [18] is an open message exchange protocol that supports reliable delivery of messages. It is considered in message-oriented environments and can be seen as the asynchronous complement of HTTP. AMQP implements different architectures such as message distribution, store and forward, and routing. It also relies on TCP for transport so it is highly impacted by high packet loss which causes excessive latency [44].

3. Extensible Messaging and Presence Protocol (XMPP)
   Similarly, XMPP[33] is an instant messaging protocol based on XML streaming protocols that has been adapted for use in IoT applications by means of protocol extensions. It enables real time exchange of structured data between connected entities. It also supports bidirectional communication. The client connects to the server and transmits messages based on XML streaming protocol. XMPP uses XML data format which generates significant overhead. Also, since XMPP relies on both, TCP for transport and XML for encoding the messages, it inherits some limitations that makes it inefficient in certain IoT network environments [44], [30].

4. Constrained Application Protocol (CoAP)
   The IETF nominated the Constrained Application Protocol (CoAP), defined in RFC 7252 [53], as the application-layer protocol for the IoT. CoAP was designed by CORE (IETF Constrained RESTful Environments) working group and became an Internet standard in 2014. CoAP is a lightweight application-layer protocol based on REpresentational State Transfer (REST) model for use with constrained devices and IoT networks [42], [27]. CoAP uses the request and response approach between nodes and supports discovery of resources and services. CoAP can be easily mapped with HTTP (HyperText Transfer Protocol) for integration with the Web. CoAP provides both non-reliable and reliable modes of transmission based on whether packet losses are tolerated or not. Reliability is achieved via retransmissions but the cost to pay is shorter battery life and higher latency. Although low packet losses and low latency are both desired, sometimes one is more sensitive than the other. For example, for weather IoT applications, where the weather reports are based on a fast sequence of sensor data readouts, latency is more important than packet loss; if a packet is lost, it can be ignored and the latest sensor values are read again. On the other hand, for healthcare applications where patient's health is monitored periodically, packet loss is much more important because each sensor data readout represents a sensitive state for the patient.

Due to the limited capabilities of objects in the IoT, the distinction among different protocols is quite foreseeable. Indeed, HTTP and TCP introduce overhead and hence are not suitable in such areas. Also, HTTP is a text-based and TCP is a connection-oriented transport that requires setting up and maintaining the connections, which is burdensome from a communication and a processing perspective. Hence, application layer protocols for the IoT must be selected carefully by taking into account all these constraints. Unlike the aforementioned protocols, CoAP relies on the User Datagram Protocol (UDP) instead of TCP, thus eliminating the cost of establishing and maintaining connections, making it more suitable for the IoT applications. Furthermore, CoAP adjusts some HTTP functionalities to meet the IoT requirements such as low power consumption and reduced operations in the presence of lossy environment and smart tiny objects with limited capabilities. Consequently, CoAP is designed to bring web functionalities to constrained devices that operate in Low-power and Lossy Networks (LLNs).

Despite its wide implementation in different areas to gather data from tiny objects and control constrained devices, CoAP reaction to network congestion is considered a great limitation of the protocol that limits its proper functioning and causes performance degradation such as packet losses, increased packet latency and overhead. Even worse, the network may become useless when the periods of congestion are long. Hence, the improvement of IoT protocols' performance is important, and as discussed previously, it is even critical especially in contexts where human lives are involved. Specifically, maintaining a proper functionality while improving the performance includes providing an efficient congestion control mechanism which reduces packet losses and energy consumption, and improves goodput and delays. This motivates us to explore and improve CoAP performance in this research work.

1.2 Research Challenges and Problem statement

From one side, with the vast number of objects connected with IoT, a huge amount of data is collected, transmitted and processed. From the other side, the wireless network and the sensor objects in IoT have limited network bandwidth and energy. Since constrained devices suffer from limited resources and processing capacities, congestion occurs when the node's traffic load exceeds its available capacity, and/or when high traffic is generated during the communication between large number of nodes. Other network related reasons are involved when some mobile networking technologies embrace sudden transmission delay spikes in addition to classic failures or disasters. As we know, congestion negatively affects the performance. Therefore, congestion control plays an important role in satisfying performance requirements. To this regard, it is important to improve performance in IoT networks but from one side, there is a lossy environment and limited network resources in terms of bandwidth and from the other side, devices have limited resources in terms of energy, processing capabilities and mem-

ory. Actually, IoT constrained devices can be too small and this can be noticed when these devices are compared to real life objects as illustrated in Fig. 1.3. This is precisely where the challenges come through.

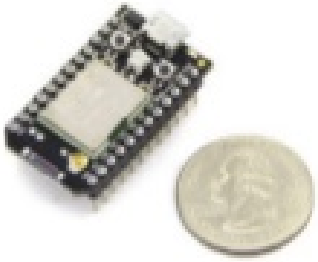


Figure 1.3: Constrained devices compared to real life objects

Another important aspect is that the performance in some IoT applications is critical where sensitive information is exchanged. In this context, healthcare IoT includes tracking and monitoring patients, elderly care, chronic diseases and even remote services. In this kind of applications, researchers usually tend to reduce losses and improve packet delay. For example, if a patient on a wheel chair falls down, packet losses in this case is unbearable.

Also, another important IoT application is disaster management. Disasters have destructive impact on economics and human life. Thanks to IoT which has proven to be capable of providing solutions for disaster management. Here again, performance is critical in terms of packet loss and latency.

Therefore, a bad protocol choice will affect the functioning of the smart object and the overall performance in general. For instance, HTTP and TCP are not suitable for devices in such environments due to the overhead they cause. Also, TCP is a connection-oriented transport protocol and requires setting up and maintaining connections. As a consequence, selecting an application layer protocol must be done carefully to take into account the aforementioned constraints in IoT networks.

For this reason, IETF designed the Constrained Application Protocol (CoAP) for data transmissions between applications of IoT devices. At the same time, CoAP meets the requirements of IoT constrained environments. As a result, CoAP has been widely used in different IoT applications ranging from smart cities, smart grid, smart homes to health care and disaster management.

The standard CoAP provides a basic congestion control mechanism based on retransmissions with a binary exponential backoff in case of packet loss, i.e., when no acknowledgement (ACK) is received [53]. In particular, it uses a fixed retransmission timeout value which is doubled for each packet retransmission. Nevertheless, recent studies demonstrate that it is still necessary to improve the congestion control mechanism of CoAP to improve further its performance in terms of reliability and efficiency [38, 45]. Besides, these studies showed that CoAP is not eligible to be adaptive to different network conditions.

Other related works have analyzed congestion control of CoAP and proposed improvements in calculating the retransmission timeout and/or the backoff procedure [14, 10, 11]. On one hand, the algorithms proposed by the literature outperform the standard CoAP [55], while on the other hand, the improvements come with the cost of performance degradation, more complexity, and increased overhead as we will see in chapter 2.

Now, clearly, the ultimate challenge is the design of a new congestion control algorithm for CoAP that is suitable for constrained devices and at the same time maintains an utmost performance in lossy networks. To do so, first we perform an in-depth analysis of the existing protocol. Second, we identify the problems of previously suggested algorithms regarding improving the congestion control mechanism in CoAP. In general, ensuring the simplicity of any protocol design surrounded by constrained factors is a huge challenge.

1.3 Contributions and Research Approaches

In this thesis, we are looking to study deeply the CoAP protocol in lossy networks and design a better congestion control mechanism. Among the different goals of this thesis, the main one is to propose a new exact analytical model to analyze the performance of CoAP in network environments modeled by Bernoulli loss model, Simple Gilbert loss model and the well-known Gilbert-Elliott two-state Markov process. Therefore, CoAP protocol needs to be analyzed so that we will be able to answer clearly this question: How to improve further the performance of the protocol? Based on our assessment, we found that the default simple congestion control mechanism can significantly reduce CoAP performance especially in networks with high packet loss, and thus preventing the protocol from acting efficiently during congestion periods. Accordingly, we suggest to reduce efficiently the retransmission timeout and supplant the backoff by a rate-based control while still ensuring simplicity. The resulting protocols, named IDC-CoAP and MBC-CoAP, are evaluated against protocols from the literature to show concretely their efficiency. Simulations and experiments in a realistic environment show that our proposed mechanisms reduce protocol losses and overhead, and provide better rate performance compared to CoAP and other previous works that aim to enhance it. The contributions of this research work are summarized as follows:

- Profound analysis and performance evaluation of CoAP through a complete and fast analytical model under the Gilbert-Elliott network loss model without any approximations.
- Profound analysis and performance evaluation of several previous congestion control algorithms, supported by the development of an original simulator dedicated to CoAP testing in a controlled and repeatable environment.
- A new algorithm for retransmission timeout reduction to enhance congestion detection.

- Integration of two new congestion control rate-based algorithms in CoAP instead of the less efficient backoff procedure while keeping the complexity reasonable for constrained devices.
- Performance evaluation results using both our *ad hoc* simulator and the well-known Cooja/Contiki realistic IoT environment show that our rate-based congestion control for CoAP achieves a better tradeoff between reliability, transmission overhead and bandwidth utilization.

Hence, our research work is based on analyzing precisely CoAP performance using the analytical model and we confirm the utility of reducing the retransmission timeout. Besides, the backoff procedure denies the protocol from achieving a better tradeoff between rate performance, reliability and overhead. Accordingly, we propose further substantial improvements to congestion control algorithms for CoAP based on the concept of rate control instead of the backoff in order to overcome previous limitations and enhance the tradeoff between reliability and rate performance.

1.4 Thesis Outline

The remainder of this thesis is organized into 3 parts. In the first part, through chapter 2, we analyze and synthesize the state of the art concerning CoAP performance evaluation and congestion control, then we highlight their shortcomings. In the second part, through chapters 3 and 4, we consider CoAP performance evaluation via modelling. In chapter 3, we present our first analytical models using Bernoulli loss model and Simple Gilbert loss model. There, our performance metrics are presented which will continue with us in chapter 4 where we compute our formulas using the Gilbert-Elliot Markov chain model. The third part is drawn in chapters 5 and 6 where our new design for congestion control mechanism is introduced, in particular, its two main components: congestion detection and congestion counteraction. In chapter 5, we present our algorithm to efficiently compute the retransmission timeout for the congestion detection phase while in chapter 6 we present our algorithm for the congestion counteraction phase. Finally, we conclude this thesis in chapter 7 and draw our future work directions.

2

Analysis and shortcomings of previous performance evaluation and congestion control of CoAP

In this chapter, we explore the state of the art related to CoAP performance evaluation and congestion control algorithms. We introduce different works to evaluate CoAP via modelling and simulations. We then present in-depth analysis of previous main algorithms suggested to improve CoAP congestion control mechanisms. In particular, the two components: Congestion detection and congestion counteraction. For this purpose, we developed a simulator using the well-known programming language Python. Through our simulator, we will be able to perform profound analysis of CoAP algorithms and show their corresponding behavior. The simulator will be presented in details in Chapter 5. Different patterns of Round Trip Time (RTT) are generated to study and analyze the performance of RTO calculation (congestion detection) of different congestion control algorithms. Besides, different network conditions are simulated to evaluate the full congestion control algorithms. Consequently, we highlight the main problems and weaknesses of the suggested approaches which enable us later on to find better alternative solutions. Before concluding the chapter, we provide our insights to tackle and overcome the highlighted problems, and provide more efficient solutions in order to enhance CoAP performance.

CoAP [53] is a lightweight application-layer protocol designed for constrained devices in IoT networks which is based on Representational State Transfer (REST) that supports basic operations like GET, POST, PUT and DELETE. Similar to the position of HTTP for Web, CoAP is dedicated for IoT devices with limited hardware capacities and functions. CoAP adopts a binary message format to reduce overhead. CoAP messages have mandatory fields (Version, Type, Token, Code, Message-ID) and optional fields (CoAP Options and Payload). The design of CoAP messages has been intended not only to keep messages small, but also to make them easy and light to be processed.

CoAP adopts some IoT-oriented features, such as asynchronous message exchange and multicast communication. Fig. 2.1 is an illustration of a CoAP environment where CoAP clients are communicating with different CoAP servers. The figure shows the client-server interaction model which happens within the same network or through the internet. The presented network is geographically distributed over hundreds of meters where communications between CoAP nodes take place via CoAP protocol. In this client-server relationship, CoAP servers host information about the resource (ex. pressure, temperature, humidity, light state, healthcare related such as blood pressure, EEG, oxygen saturation, heart rate, etc..) which will be manipulated by CoAP clients. Clients send requests for specific action (read/update/delete using GET/POST/PUT/DELETE methods) on a resource hosted by the server, and the server sends the response after processing the request. Communication details will be presented in the next figure.

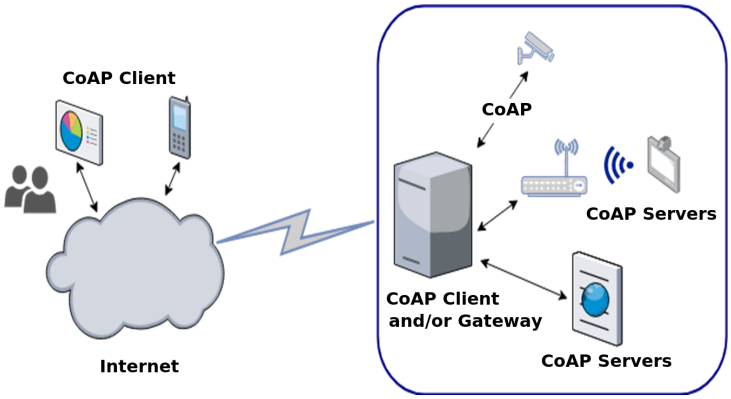


Figure 2.1: An overview of connected devices via CoAP protocol

CoAP operates on top of UDP, which is a lightweight transport protocol because UDP does not require handshaking or end-to-end connection establishment between communicating devices which consequently makes UDP communication very efficient. Also, the connection-less nature of UDP does not require communication and memory overheads which is introduced in the establishment and maintenance of TCP connections. CoAP is based on a request/response communication model and CoAP URIs identify the resources of the application. Representations of resources are exchanged between the

server and the client. A client that is interested in the state of a resource sends a request to the server then the server responds with the current representation of the resource. CoAP URIs use either the coap or coaps (secure CoAP). Coaps uses DTLS as a secure transport, similar to HTTP's use of TLS. The default UDP port for coap is 5683 and 5684 for coaps. As per fig 2.2, the client is requesting the room temperature (temp) and the server is replying with the sensor value.

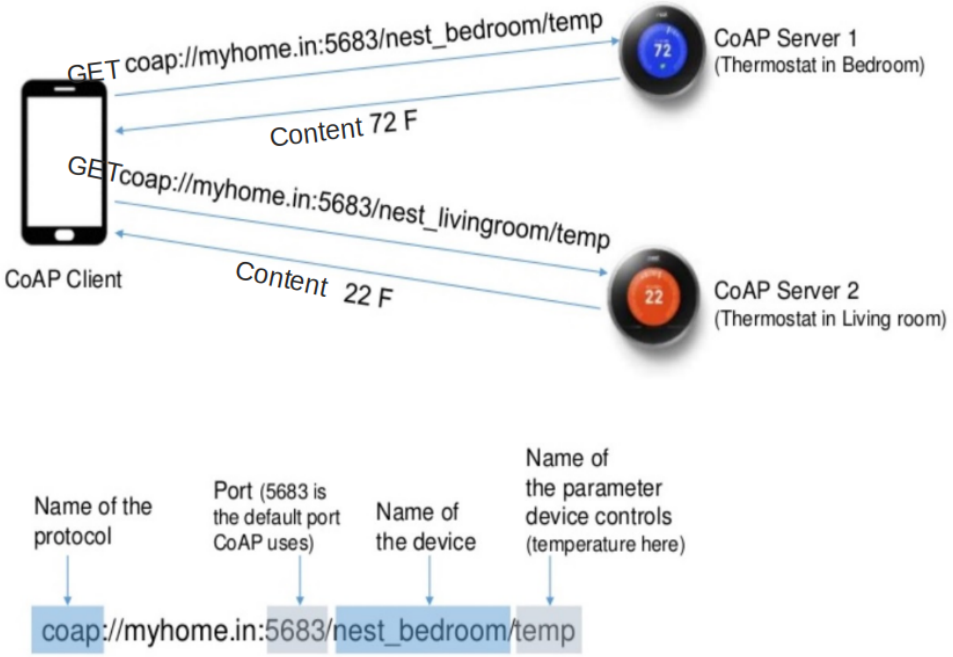


Figure 2.2: Communication between nodes using CoAP protocol

Compared to other protocols used for constrained devices such as Message Queue Telemetry Transport (MQTT) and Advanced Message Queuing Protocol (AMQP) [42], one major difference is that CoAP relies on UDP for communication, consequently removing the load of establishing and maintaining connections, which may be infeasible for smart objects with limited capabilities. Hence, it is suitable for communications in wireless networks that suffer from high packet loss [56] because it removes the overhead and complexity imposed by other transport protocols. However, as UDP is inherently not reliable, CoAP should provide its own reliability mechanism which can be designed to be very simple. CoAP communications can be selected to use either confirmable (CON) or non-confirmable (NON) messages. The standard CoAP [53] uses a simple mechanism for congestion control based on retransmissions with a binary exponential backoff (Fig. 2.3) when reliable communication is required by the application. When a packet containing a CON message is lost, the client re-sends the message at doubled increasing intervals, until an acknowledgement (ACK) is received or the allowed number of attempts is reached. Two parameters control the retransmission process: An initial timeout value RTO_{init} and a maximum retransmission value r. For each new CON message, the first timeout

is set to a value between RTO_{init} and RTO_{init} times a randomization factor f, and the retransmission counter is initialized to 0. If the sender did not receive an ACK for the CON message, CON is retransmitted and the retransmission counter is incremented and the timeout value is doubled. The whole process for each CON is repeated until the counter reaches its limit r or an ACK is received by the sender. This process is illustrated in Fig. 2.3.

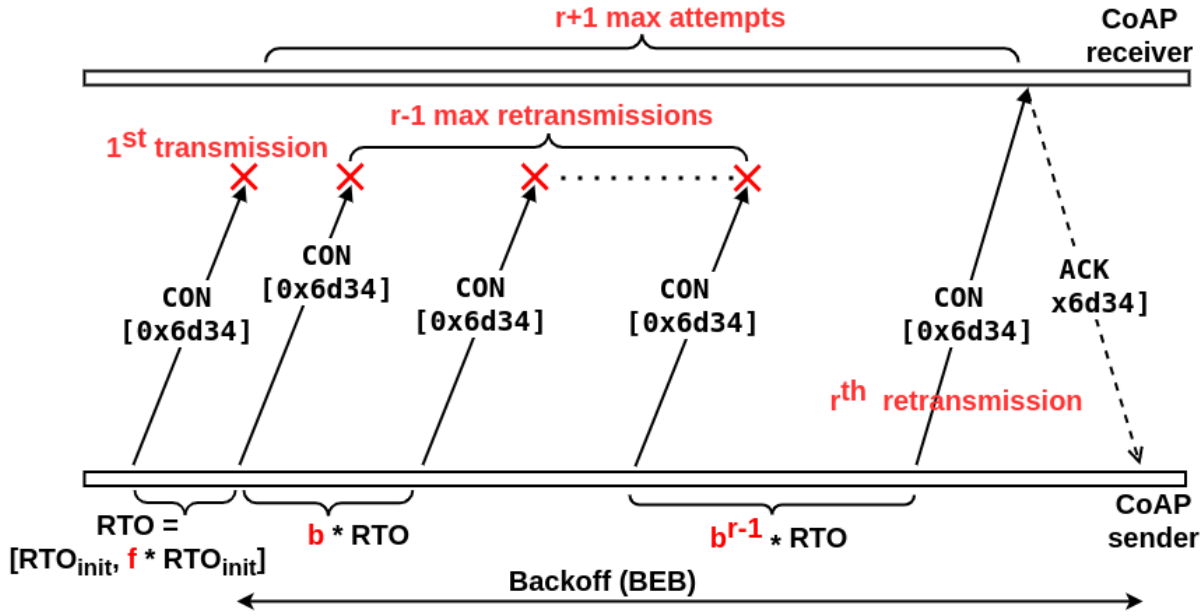


Figure 2.3: CoAP Congestion Control. Retransmission counter: r=4, Retransmission timeout: RTO_{init}=2s, Random factor: f=1.5, Backoff factor: b=2

CoAP literature related to our research is twofold. The first one focuses on analyzing the performance of the default CoAP protocol using modelling and experiments, while the second one includes evaluating different CoAP congestion control algorithms and presenting improvements on CoAP parameters and the congestion control mechanism.

2.2 CoAP Modelling and Performance Evaluation

The downside of default CoAP parameters on performance can be identified by simulations and/or modeling. Although some previous research works have focused on evaluating CoAP based on mathematical models, some of these works did not provide models for different performance metrics while other works are not precise as we will see next. For instance, some analytical models were developed for CoAP performance in [30] and [31]. However, they are not adequate because they use the *steady state* probabilities of the Gilbert-Elliott (GE) Markov chain to analyze *successive* transmitted packets that are correlated. In other words, the GE model is used as if it is equivalent to a Geometric distribution (Bernoulli) with the probability being one of the steady state probabilities of GE. As a consequence, these models do not even apply for the simple Gilbert case, i.e. $k = 1, h = 0$. Besides, these works

consider that GE is applied to data packets and also to ack packets. Normally, GE model should be used as an *observed* network model by a given node, not as a model that applies losses to all packets coming from different nodes.

CoAP hires non-reliable and reliable operation modes [53] depending on whether packet losses are tolerated or not, respectively. The author in [29] presented an analytical model of CoAP packet loss for each CoAP transport mode. Afterwards, CoAP transport mode is dynamically selected based on the packet loss goal as an input in order to preserve battery life. The model is used to estimate the packet loss at the application layer and is built in accordance with the Markov chain but not extended to include other performance metrics such as latency, goodput and overhead. Also, the model is not used to improve CoAP performance in reliable mode but to switch between different modes.

The authors in [24] carried out quantitative evaluation of CoAP performance in dynamic network using an emulator (CORE) versus HTTP transmission over TCP. They also introduced briefly three formulas for success rate, overall delay and overhead. The delay is formulated by $D_O + \sum_{i=1}^{R-1} P_S(1 - P_S)^i t_i$ where D_O is the end-to-end one-way delay, P_S is the success rate probability of one transmission attempt, t_i is the waiting time of the i^{th} retransmission attempt, and R is the total number of transmission attempts. Although they introduced three performance metrics, their probabilities are not accurate and the presented delay formula is *wrong* because it is based on the geometric distribution while we must use the truncated geometric distribution. CoAP was not modelled in different network loss models such as the Simple Gilbert or Gilbert-Elliot. Moreover, the presented formulas were not validated with respect to the experimental results.

In the literature, the Gilbert-Elliot (GE) model is commonly used in several applications. Many previous works use the GE model as a tool to study the performance of some network mechanisms for instance [13, 39, 23]. These works can not be applied to compute the performance of CoAP such as the theoretical loss probability under a given traffic pattern. For instance, GE is usually used to challenge and evaluate Automatic Repeat Request (ARQ) protocols. In general, ARQ protocol follows three basic approaches: stop-and-wait (SW), go-back-N (GBN), and selective-repeat (SR) [39]. In SW ARQ, the sender receives the ACK of a packet before sending a new packet. In GBN ARQ, packets are sent continuously without waiting for ACKs/NACKs. The sender retransmits the negatively acknowledged packet and all subsequent packets if a NACK is received. In SR ARQ, packets are sent as in GBN ARQ, but only negatively acknowledged packets are retransmitted. Although Gilbert-Elliot model is used in [39], the work considers SR ARQ with Negative ACK without the use of timeouts, while CoAP considers Stop-and-Wait behavior which is based on timeouts. The authors have pointed the important difference between these two behaviors. The analysis is completely different for CoAP. Also, the congestion control behavior in CoAP is absent in ARQ studies for link layers. Their work applies to link layer reliability protocols in wireless channels.

On the other hand, previous Gilbert-Elliot models in wireless links are also not applicable. For instance, the authors in [7] presented a model to evaluate the performance of packet loss in wireless environment based on two-state Markov process channel mode. They analyzed the probability of success packets transmission within a network suffering from bursty losses. They differentiated between two cases. In the first case, packets are duplicated and the duplicated packet is sent after the transmission of the original packet while in the second case a newly generated packet is transmitted couple of times. Although they provided strategies for setting packet delay to optimize performance and reduce the effect of packet losses in mobile environment, they studied a transmission mechanism which is different from CoAP mechanism which is a Send-and-Wait with the involvement of XORing or duplicating packets.

In [23], the authors study frame aggregation and block acknowledgments employed in IEEE 802.11n standard. The proposed model calculates an expected number of retransmissions by estimating the amount of sub-frames to be retransmitted under Gilbert-Elliot channel model. With their Markov model, they are able to formulate the throughput at MAC layer as a function of other parameters such as physical data rate, error rate, and path length. Although the proposed model can describe some characteristics, such as fluctuations and burstiness, the studied behavior of aggregation and block acknowledgment is different from the behavior of CoAP. The authors do not consider timeouts. As many other related works, the authors assume that before each transmission attempt, the Gilbert-Elliot model is in steady state which is not suitable for our CoAP analysis in this thesis.

The authors in [13] tried to characterize the quality of wireless links using the Gilbert-Elliot model. They considered the reception of packets as a sequence of bits: 1 for the successful reception and 0 for the lost packet. This formulation leads to the possibility of computing the stationary probabilities of being in the Good or Bad state in the Gilbert-Elliot model, however, they do not compute the theoretical loss probability under a given traffic pattern. They analyze the packet reception ratio based on real measurements and Gilbert-Elliot model is used only to analyze some network scenarios but not to model the performance metrics.

As we have seen, some previous research work considered using modelling to evaluate CoAP, but they fail to correctly compute the performance metrics. Some used the steady-state probabilities of the Gilbert-Elliot model which is inadequate for CoAP protocol. Other related literature performed CoAP evaluation using experiments [54, 25, 19, 56]. However, evaluating CoAP and choosing its parameters based on few experiments is insufficient.

2.3 Congestion Detection: RTO Calculation

Recently, there have been works conducted to improve CoAP performance and particularly its congestion control mechanism. According to our perception, two main procedures should form this mechanism: Congestion detection and congestion counteraction. In this section, we will present previous

works related to different RTO calculation approaches proposed to enhance congestion detection. As a matter of fact, timeouts are the only indicator to detect packet losses in CoAP. These losses are an important indicator for congestion. After presenting the different works related to estimating the retransmision timeout (RTO), we will analyze the shortcomings of the main proposals.

In [20], the authors suggested adjusting CoAP parameters to handle high traffic and high loss probability. They experimentally adjusted the constant value of the retransmission timeout (RTO) and the random factor f, and showed that the results achieved by tuning CoAP parameters overtakes MQTT results in terms of throughput and latency. They adopted new CoAP congestion control parameters based on experiments which is not sufficient since experiments are limited and usually do not cover different network conditions. Also, using small values of the backoff factors will make the sender to transmit faster without waiting for the proper time during congestion periods which induces high packet losses.

The work in [43] proposed a congestion control scheme based on round trip time (RTT) measurements through a counter using the option field of the CoAP message. The sender recognizes the origin of an ACK and calculates the correct RTT to update the retransmission timeout (RTO) without smoothing. Therefore, instead of using the default RTO CoAP parameter (2 sec) for the retransmissions when ACK is not received, the authors use the new calculated RTO for the retransmissions.

An adaptive mechanism for handling congestion in CoAP called Fast-Slow RTO (FASOR) is proposed in [37]. RTO computation is composed of fast and slow RTO calculation. Fast RTO is based on TCP retransmission timer and updated with unambiguous RTT samples (where ACK messages matches CON messages) while slow RTO is measured from the original transmission of a packet till the arrival of its ACK regardless of the required number of retransmissions. Fast RTO is used as a sign of link error (interference) whereas slow RTO is a sign of heavy congestion. The authors evaluated their algorithm among CoAP and an advanced version of CoAP called CoCoA+, which will be explored next, using Netem [28] as network emulator and libcoap library [8]. The authors tried to deduce whether a packet loss is due to link disruption or due to congestion by examining RTT samples (ambiguous or not), however, RTT analysis can hardly be used to detect the reasons behind a packet loss. Also, when the network is lossy, the backoff mechanism will lead to a behavior similar to CoAP due to the exponential backoff mechanism. They also set the retransmission counter to 20 instead of 4 which may cause long packet delays and a very low throughput which might be useful only when packet delivery must be guaranteed on the behalf of all other performance metrics.

CoAP Simple Congestion Control Advanced CoCoA [15] specifies RTO calculation based on TCP RTO computation algorithm where RTT is used to update the RTO value automatically. CoCoA uses two RTO estimators: A strong and a weak RTO estimator that are updated when measuring strong RTTs and weak RTTs respectively. When an ACK is received after the first transmission of a packet, strong RTTs are measured while weak RTTs are measured when an ACK is received after at least one retrans-

mission of the packet. Strong RTTs are used to estimate an average of strong RTTs ($RTTVAR_{strong}$) which is in turn used to update the strong RTO estimate. Similarly, an estimated average of weak RTTs ($RTTVAR_{weak}$) is used to update the weak RTO estimate. When a new RTT value is obtained, the following formulas are calculated:

$$RTTVAR_X = (1 - \beta)RTTVAR_X + \beta|RTT_X - RTT_Xnew| \tag{2.1}$$

$$RTT_X = (1 - \alpha)RTT_X + \alpha RTT_Xnew \tag{2.2}$$

where X represents either strong or weak with $\alpha = 1/4$ and $\beta = 1/8$. Then RTO_X and $RTO_{overall}$ are updated accordingly.

$$RTO_X = RTT_X + 4 * RTTVAR_X \tag{2.3}$$

$$RTO_{overall} = 0.5 * RTO_X + 0.5 * RTO_{overall} \tag{2.4}$$

The new RTO estimation is used as the initial RTO (RTO_{init}) for the next packet transmission. The authors showed that the performance of CoCoA is better than CoAP in congested networks but different studies [10] show that it has some limitations with bursty traffic.

The authors in [9] evaluated another version of CoCoA named CoCoA-S using only the strong estimator. They employed alternative algorithms developed for TCP which are shown to be inadequate for CoAP. CoCoA-S is conservative in lossy networks which results in low throughput. The main algorithm CoCoA delivers better performance in comparison to CoCoA-S.

CoCoA+ [10] has adopted the same RTO estimation mechanism presented in [15] with minor updates on the weak estimator weights. They also introduced new variables in equation (2.3) to update RTO estimation. The equation becomes $RTO_X = RTT_X + K_X * RTTVAR_X$ where K_X represents either K_{weak} or K_{strong} with values 1 and 4 respectively. Strong K is used to update the strong RTO estimate while weak K is used to update the weak RTO estimate. In addition, they added an ageing mechanism to set RTO values if RTT is not updated for a certain time. The impact of the weak estimator is reduced by reducing its weight but according to [14], this countermeasure works well only in steady conditions and when the network load is constant.

pCoCoA [14] has suggested different modifications to CoCoA+ RTO estimation in order to reduce spurious transmissions which refers to packets that are retransmitted because of incorrect estimation of RTO. Although different research works suggested improvements to CoCoA RTO calculation algorithm and various modifications have been implemented over the past years, pCoCoA proved to overcome many limitations suggested by other works. As a consequence, in the following, we will explore

the latest version, pCoCoA [14]. The retransmission timeout calculation is based on TCP Linux. Evaluation showed an improvement of the RTO calculation using several weights to set the round-trip time variance. However, many instructions are used which increases the processing overhead in constrained devices. The algorithm consists of the following main aspects:

- Introducing an option to match each ACK message with its relevant CON message
- A mechanism to calculate the retransmission timeout (RTO) based on linux TCP
- A variable backoff mechanism to set the RTO for retransmissions adopted from CoCoA+ [10] which is presented in the next section.

1. **ACK-CON matching:** In CoAP and CoCoA+, there is no mechanism to match a CON message with its corresponding ACK. pCoCoA adopts the transmission counter (TC) option to link each ACK message with its relevant CON message even when the CON is retransmitted. In particular, TC is set, incremented for each retransmission and echoed in the ACK. ACK-CON matching is based on the TC value. This mechanism is used for round-trip time estimation as well as detecting spurious retransmissions. A retransmission is considered to be spurious when the CON message is falsely retransmitted just because the retransmission timeout value was insufficient.

2. **pCoCoA approach for RTO calculation:** The RTO_{init} calculation in pCoCoA follows almost the same algorithm as the one implemented for TCP in the Linux Kernel [52] with additional instructions. RTO_{init} value is updated according to RTT measurements. RTO_{init} calculation in pCoCoA tries to handle two problems: First, when RTT increases suddenly and causes RTO_{init} overestimation. Second, when RTT variance (RTTVAR) drops to a small value leading to spurious retransmissions. pCoCoA algorithm introduces the maximum mean deviation (mdev_{max}) for the RTO. The parameters are initialized when the first corresponding RTT value R is received as follows: $SRTT \leftarrow R$, $RTTVAR \leftarrow R/2$, $mdev_{max} \leftarrow max(R/2, 250ms)$, and $RTO \leftarrow SRTT + mdev_{max}$.

   When a new value R is computed, RTO_{init} is updated for the future transmissions. SRTT and RTTVAR are updated using different weights to slow down the sudden decrease when fluctuations happen (block 1 in Algorithm 1 pCoCoA). When RTTVAR is less than the difference between SRTT and R, the weight $\beta = 1/4$ is used to update RTTVAR value, otherwise the parameter $\alpha = 1/8$ is used.

   When there is a sudden increase in the network delay, $mdev_{max}$ with aging mechanism is introduced to increase RTO value accordingly (block 2).

   If spurious transmission is detected, k is set to 6, otherwise, it is set to 4 (block 3).

Algorithm 1 pCoCoA RTO calculation: block 1

$SRTT = (1 - \alpha)SRTT + \alpha R$
if $R < (SRTT -$ RTTVAR$)$ **then**
  RTTVAR $= (1 - \gamma)$RTTVAR $+ \gamma|SRTT - R|$
else
  if $|SRTT - R| >$ RTTVAR **then**
    RTTVAR $= (1 - \beta)$RTTVAR $+ \beta|SRTT - R|$
  else
    RTTVAR $= (1 - \alpha)$RTTVAR $+ \alpha|SRTT - R|$
  end if
end if

Algorithm 1 pCoCoA RTO calculation: block 2

if $R > SRTT$ **then**
  if RTTVAR $> mdev_{max}\ for\ 3\ consecutive\ times$ **then**
    $mdev_{max} = average\ of\ the\ last\ 3$ RTTVAR
  else if RTTVAR $> mdev_{max}\ for\ 8\ consecutive\ times$ **then**
    $mdev_{max} = (1 - \beta)mdev_{max} + \beta$RTTVAR
  end if
end if

Algorithm 1 pCoCoA RTO calculation: block 3

if $(spurious)$ **then**
  $k = 6$
else
  $k = 4$
end if

Lastly, RTO_{init} is updated using another smoothed variable SRTO. SRTO calculation is based on the spurious flag k in order to limit the minimum SRTO values. Then, RTO_{init} is computed using another weighted sum that combines SRTO and previous RTO_{init} as per block 4.

Algorithm 1 pCoCoA RTO calculation: block 4

$SRTO = SRTT + max(k * \text{RTTVAR}, mdev_{max})$
$RTO_{init} = (1 - \delta)SRTO + \delta RTO_{init}$

2.4 Analysis and Shortcomings of Previous RTO Estimation Algorithms

In CoAP, RTO is fixed and its default value is $2s$, the initial RTO before the first retransmission attempt is chosen from a fixed interval $[2, 1.5 * 2]$ according to CoAP RFC [53]. The main difference in the improved algorithms of RTO calculation compared to CoAP is the use of Round-Trip Time (RTT) to set RTO. Then, similar to CoAP, a dithering technique is implemented to the estimated RTO value and hence the initial value of RTO is chosen from the interval $[RTO_{init}, 1.5 * RTO_{init}]$ [10]. In the following, we will present the shortcomings of previous RTO calculation algorithms.

2.4.1 Inefficient weak and strong estimators

As already mentioned, CoCoA and CoCoA+ use two RTO estimators, weak and strong. A weak RTO estimator is updated when measuring weak RTTs and strong RTO estimator is updated when measuring strong RTTs. Weak RTT is sampled from transactions that require at least one retransmission of a CoAP message while strong RTT is sampled from transactions that do not require any retransmission. However, subsequent updates of the RTO strong estimator when RTT is not oscillating can cause the new value of RTO_{init} to increase enormously if RTT decreases suddenly. In particular, after some stable measurements, the variance of RTT (RTTVAR) becomes too small, hence, any sudden fluctuation in RTT increases RTTVAR and consequently increases the RTO_{init} because it is multiplied by 4 as in equation (2.3). Therefore, their design leads to undesired effects on the calculation of RTO values.

In CoCoA+, they kept the weights of the strong estimator which imposed the same problem. This is illustrated in Fig. 2.4, where RTT is varied continuously according to a normal distribution with mean 2000ms and a standard deviation of 50ms during 50 times, then a sudden change to a normal distribution with mean 3000ms and a standard deviation of 50ms during 50 times, then a sudden return to the previous distribution and so on and so forth. The green plot represents the value of RTO which is estimated according to RTT samples (red plot). The x-axis represents the number of samples over time. As shown in the figure, RTO_{init} increases when RTT decreases and it looks like RTO values are inversely estimated (x-points at sequence numbers 400 - 415). On the other hand, CoCoA+ authors tried to minimize the effect of the weak estimator by using 0.25 and 0.75 weights instead of the 0.5

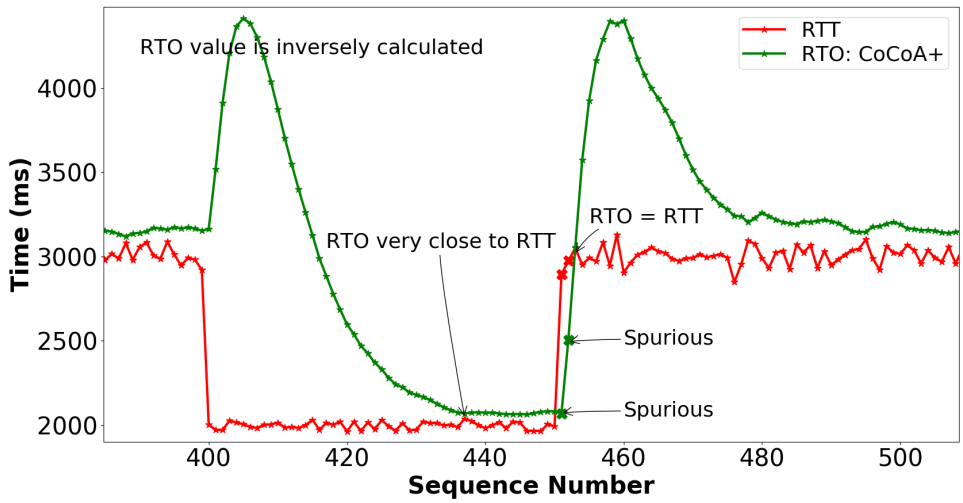


Figure 2.4: CoCoA+ RTO estimation

weight in their estimation: $RTO_{init} = 0.25 \times RTO_{weak} + 0.75 \times RTO_{init}$. Also, they reduced the value of K_{weak} from 4 to 1. Although this change might reduce the impact of the weak estimator in some network scenarios, however, it will fail to increase the RTO value efficiently when RTT increases as per Fig. 2.4 (x-points at sequence numbers 450 - 452). After x-point 452 and till x-point 460, RTO keeps converging to a very high value. So here again RTO does not converge properly. Hence, not only this new update is unable to overcome the shortcomings of CoCoA but will also lead to a sequence of spurious transmissions.

2.4.2 Spurious Transmissions

In CoCoA+, the RTTVAR variable, that calculates the mean variance of RTT, is reduced when RTT samples are similar. Hence, RTO_{init} values become close to the sampled RTT. The problem occurs when RTT values fluctuates after certain stable behavior of the network which usually happens in case of interference, low signal strength or intermittently dropping the connection with other nodes. This sudden change after some stable RTT measurements increases the probability of having spurious transmissions as can be observed from Fig. 2.4.

In pCoCoA, when RTT increases suddenly to a higher value, RTO increases because of the smoothed RTT (SRTT) value which is based on the average of the measured RTTs. However, after few transactions with lower RTTs, RTO decreases very closely to RTT which leads to spurious transmissions. In Fig. 2.5, we simulated such scenario where we plot the evolution of RTT values and RTO values over time. We used the same RTT samples presented in Fig. 2.4. When RTT decreases suddenly, pCoCoA RTO converges quickly and this is risky because the sudden decrease might be followed by a sudden increase which leads to spurious transmissions as shown in the figure (bold green x-points at sequence numbers

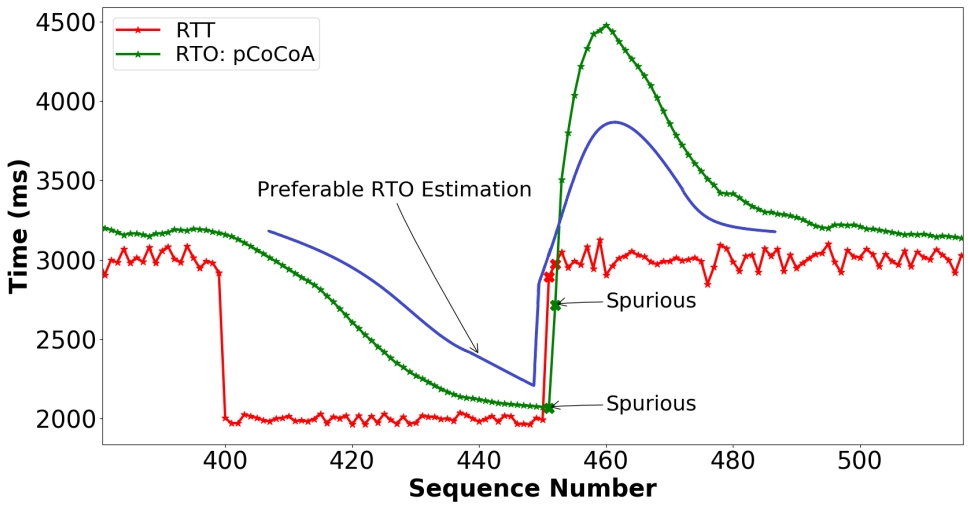


Figure 2.5: pCoCoA RTO estimation vs preferable RTO Estimation

451 and 452). A better design should handle RTO convergence gradually when RTT decreases as per the blue plot in the figure.

2.4.3 Large RTO Estimations

It could be simple to estimate RTO much greater than RTT to avoid spurious transmissions. However, the larger RTO, the lower the goodput and the longer the transmission delay. Also, the reaction to congestion may take more unnecessary time. Ideally, RTO should be as close as possible to RTT values without triggering spurious transmissions. Particularly, when RTT increases suddenly as simulated in Fig. 2.5 and Fig. 2.4, RTO is calculated by CoCoA+ and pCoCoA after the sudden increase of RTT but not fastly enough to avoid spurious transmissions. In such case, RTO must converge quickly as per the blue plot of Fig. 2.5 to avoid spurious transmissions but without exceeding that much RTT values as CoCoA+ and pCoCoA are behaving. Note that RTT values can increase suddenly due to severe congestion or other factors such as burst connection arrivals or handoffs in wireless networks.

2.4.4 Complexity

As per our observations in different network scenarios, the analysis of the RTO values computed by CoCoA+ has shown very high values of RTO when RTT decreases. We have examined this behavior in almost all simulations even the ones with few fluctuations in RTT samples before the sudden decrease. In Fig. 2.6, the RTO estimators could not prevent the unexpected increase of RTO values (x-points at sequence numbers 200, 300, 400, 500). Also, as presented previously, the weak estimator leads to

spurious transmissions. Therefore, the efforts of declaring and maintaining the variables (Weak - Strong for each variable name) are doubled without making a substantial impact on RTO convergence.

pCoCoA adopts RTO calculation mechanism similar to the one implemented in Linux TCP [52]. Although Linux TCP is used by many network applications in the internet, its complexity makes it not efficient for constrained devices that are limited in storage and processing capabilities. Especially, our simulations show that the block for calculating $mdev_{max}$ in Algorithm 1 pCoCoA block 2 is being executed up to 70% in each simulation but not being used in RTO_{init} final calculation except in few cases as shown in Fig. 2.7. We tested 29 different RTT network scenarios (x-axis) which are detailed later in Table 5.1 (Chapter 5 - Section 6.2). The blue bars show the calculation for $mdev_{max}$ if RTTVAR is greater than mdev for 3 consecutive times, while the red bars present the calculation $mdev_{max}$ when RTTVAR is less than mdev for 8 consecutive times. The yellow bars present the usage of $mdev_{max}$ when calculating SRTO. Although it is calculated in all the transactions, $mdev_{max}$ is rarely used. This overhead might seem negligible with normal machines but it increases energy consumption and overhead computation in a constrained IoT environment for a negligible benefit.

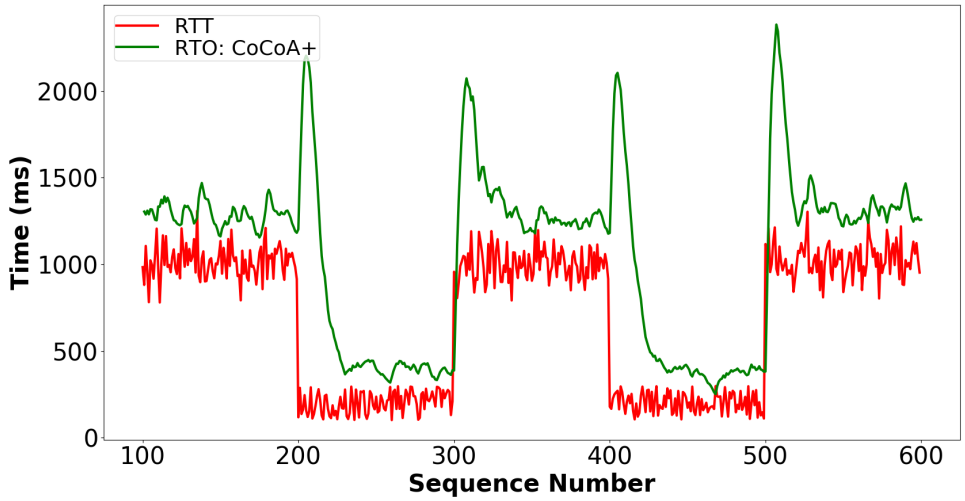


Figure 2.6: CoCoA+ RTO behavior vs RTT fluctuations

2.4.5 Dithering technique

For each new CoAP message, RTO_{init} is set to a random duration using a random factor. As per CoAP RFC, this random factor should have a value that is different from 1 to protect from synchronization effect. The default value used by the original CoAP is 1.5. The same procedure is implemented in the improved and most recent versions of RTO calculation mechanisms, CoCoA+, 4-state and pCoCoA. However, applying this dithering technique to RTO values which are calculated using RTT measurements and a couple of instructions to tune the moving averages will have a bad impact on the waiting

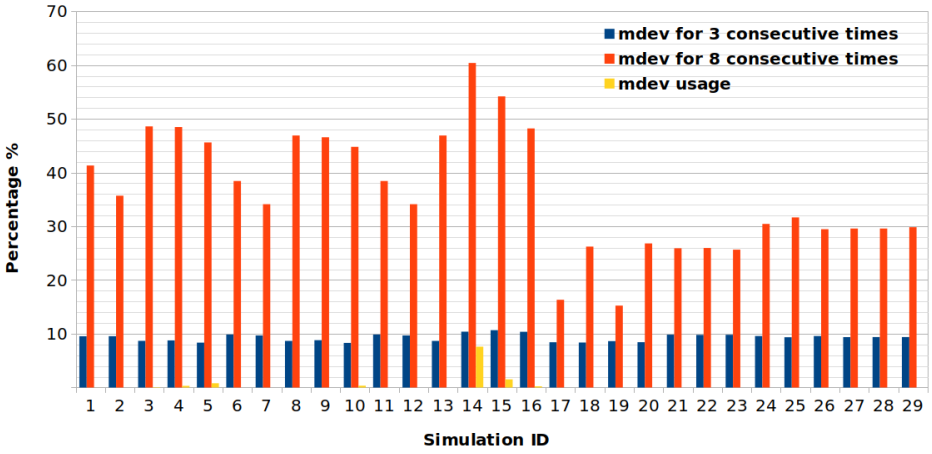


Figure 2.7: $mdev_{max}$ usage by pCoCoA

time for the next retransmissions. The main reason is that such technique will increase the initial RTO value by around 25% up to 50%, therefore, all the efforts done to reduce and optimize RTO$_{init}$ are vanished.

2.5 Congestion Counteraction

In this part, we present the literature and in-depth discussion related to the complementary part of congestion control mechanism: The congestion counteraction. We dig up two approaches to handle congestion counteraction: Backoff-based and Rate-based. From the analysis of both approaches and through the obtained results, the shortcomings of these mechanisms are identified afterwards.

2.5.1 Backoff-based approach

Here, we present the different works suggested by the literature that follow the backoff-based approach. The successive retransmission timeouts in the backoff mechanism of CoCoA+ and pCoCoA are based on the maintained RTO$_{init}$ (presented in the previous section) and also on several values of backoff factors used to multiply the timeouts instead of 2 (doubling) in case of successive losses. The authors of CoCoA+ introduced a backoff policy to set the timeout for the retransmissions named Variable Backoff Factor (VBF) to replace the binary exponential mechanism used by default in CoAP. Although pCoCoA presented a full approach for calculating RTO value, they adopt the same CoCoA+ backoff mechanism for setting retransmission timeout values in the backoff period. The value of VBF is chosen from a list $[1.5, 2, 2.5]$ according to RTO_{init} value. After the publication of CoCoA+, the authors continue adapting VBF values. For instance, they used $[1.3, 2, 3]$ and $[1.5, 2, 3]$ and these suggestions were based on experiments. That means some values are better in some cases than others. This presents one of the weaknesses of CoCoA+ since some values might fit some network scenarios and not other scenarios.

The RTO for the backoff period is defined by:

$$RTO_{backoff} = RTO_{init} \times VBF, \quad \text{where} \tag{2.5}$$

$$VBF = \begin{cases} 2.5, & RTO_{init} < 1 \text{ sec} \\ 2, & 1 \leq RTO_{init} < 3 \text{ sec} \\ 1.5, & RTO_{init} \geqslant 3 \text{ sec} \end{cases} \tag{2.6}$$

Using small backoff factors does not increase the time between retransmissions when RTO_{init} is larger than 3 seconds, while using large backoff factors when RTO_{init} is small avoids quick retransmissions in a short time which may cause further congestion. For RTO values smaller than 1 sec, new RTO value grows faster with VBF than the default BEB CoAP mechanism whereas it behaves in a similar way in both mechanisms for RTO values between 1 and 3 sec as presented in [10].

In the literature, some research mainly focused on improving RTO estimation only and adopted the backoff concept from other works while others focused on improving the backoff concept and adopted the RTO estimation from previous research. Indeed, CoCoA-4-State-Strong [11] adopted the strong estimator for RTO calculation from CoCoA and introduced an improvement for the backoff concept that uses a 4-state estimator for variable backoff factors. They differentiated between four states and each state was given a weight to be used when a loss is detected. Based on the number of packet retransmissions, each transaction is considered to be in one of four states (1-2-3-4). Each time a packet is retransmitted, its state in increased by 1. When the packet is transmitted and acknowledged, its state is decreased by one. For each transaction, there are four different backoff factors corresponding to four different states (VBF1, VBF2, VBF3 and VBF4). This is summarized by:

$$VBF = \begin{cases} 1.1, & \text{VBF1 for state } 1 \text{ (No retransmissions)} \\ 1.3, & \text{VBF2 for state } 2 \text{ (1 retransmission)} \\ 1.7, & \text{VBF3 for state } 3 \text{ (2 retransmissions)} \\ 2.5, & \text{VBF4 for state } 4 \text{ (} \geqslant 3 \text{ retransmissions)} \end{cases} \tag{2.7}$$

The RTO for the backoff period is computed similar to equation (2.5). Using small weights for the backoff factor will improve the throughput because transmitting will be faster but packet losses will increase. In addition, and according to [50], more retransmissions is more likely to occur than CoCoA which leads to more overhead and consumption of battery life.

2.5.2 Rate-based approach

The previous works mentioned above follow what we call the backoff-based approach since they act on the retransmission timeout and its evolution during the backoff period. Few researches suggested alternative approaches to deal with congestion. The authors of BDP-CoAP [5] implemented the congestion control of TCP BBR protocol [17] which follows a rate-based measurement-based approach. Instead of using mainly packet loss (three duplicate ACK reception) to infer the congestion, TCP BBR estimates the Bandwidth-Delay Product and determines the maximum number of packets in flight to not exceed in order to prevent losses. The Bandwidth-Delay Product is computed by estimating the round trip propagation delay and the available bandwidth through several measurements. In particular, an available bandwidth measurement is obtained at the reception of every ACK. A max filter is used to stabilize the estimated available bandwidth over a sliding time window.

$$\widehat{AvaiBw} = max\left(MeasBW_t\right) \quad \forall t \in [T - W_B, T]$$

where W_B is a time window, and T is the current time

TCP BBR stops sending packets when the number of packets in flight, i.e. packets that have not received yet their acknowledgements, is larger than the Bandwidth-Delay Product so that the bottleneck queue does not grow up more and thus buffer overflow is prevented.

The estimated max-filtered bandwidth is also used to control the sending rate through an eight-phase cycle with the use of pacing gains. Each phase corresponds to a packet transmission as shown in Fig. 2.8. At each phase, the sending rate is set to the estimated bandwidth multiplied by the pacing gain. In the first six phases of the cycle, the pacing gain is equal to 1. Then in the seventh phase it is set to 5/4 to increase the sending rate and probe for the available bandwidth. However, if in this phase, some losses had occurred as per Fig. 2.9, then the pacing gain is set to 3/4 to reduce the sending rate. In the eighth phase, the pacing gain is set to 3/4 in a preventive approach in case the probing of the bandwidth is not successful and also to empty any resulting queue. The values 5/4 and 3/4 are chosen so that the average sending rate during the two probing and preventive phases does not change from other phases: $\left(\frac{5}{4} + \frac{3}{4}\right)/2 = \frac{8}{4}/2 = 1$. Hence, BBR cycles through the following values of pacing_gain: 5/4, 3/4, 1, 1, 1, 1, 1, 1. This approach allows to probe for more bandwidth by increasing the sending rate using pacing gain factor of $5/4$ and then immediately reducing it again using $3/4$ pacing gain factor. According to the authors of TCP BBR [17], this cycling scheme, allows BBR flows to achieve high throughput, low queuing delay, and convergence to a fair share of bandwidth.

According to the authors of [32], BBR does not reflect the perspective of an individual sender but the aggregate behavior of all flows which leads to a sustained overload. Also, BBR ignores packet loss as a main congestion signal which also may produce massive packet loss and increases further the conges-

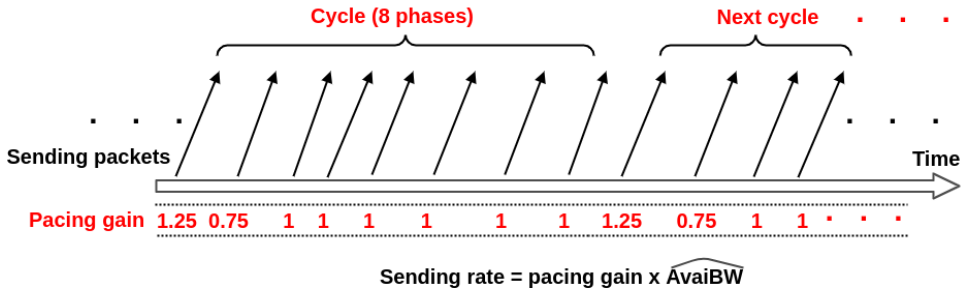


Figure 2.8: The eight-phase cycle scheme in TCP BBR

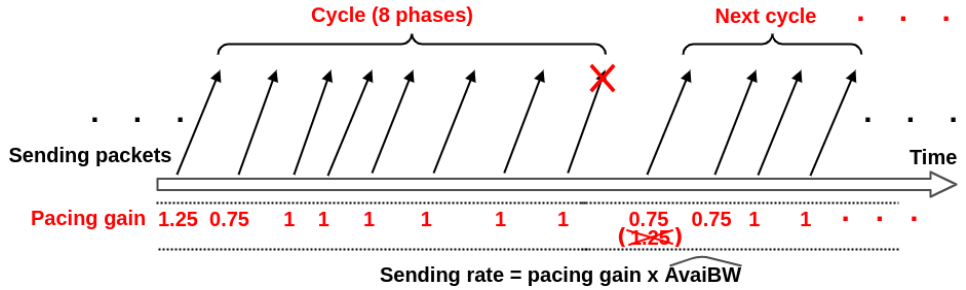


Figure 2.9: Packet loss in the seventh phase cycle scheme of TCP BBR

tion. On the other hand, the authors of C2TCP [3] showed that although the concept of BBR might work in a wired network where the bottleneck link bandwidth does not change very fast, in a highly dynamic environment, it does not perform well.

BDP-CoAP implements all the components proposed by BBR for TCP congestion avoidance with additional differences. First, BDP-CoAP computes the pacing gain factor through a ten-phase cycle instead of eight. Second, the pacing gain values of the probing phase and the preventive phase are 1.2 and 0.8 instead of 1.25 and 0.75 respectively. Other pacing gains are equal to 1 as in TCP BBR. Third, neither the pacing gain nor the estimated bandwidth are updated in case of retransmission. Indeed, when an ACK of a retranssmitted CON is received, the bandwidth measurement that can be done using this ACK is canceled and the estimation function is not called. Fourth, the filter used to compute the estimated bandwidth uses, in addition to the maximum of previous measurements, the minimum of these previous measurements. The min and the max are combined together with a weighted sum. Fifth, the time window used to compute the filtered estimated bandwidth is removed. Instead, the filter considers the last 10 measurements done at the last 10 packet sending instants. Naturally, losses are still detected via RTO_{init} expiration but its value is not multiplied by a backoff factor in case of successive losses. RTO_{init} is estimated exactly as in CoCoA+. In summary, in addition to BBR components, BDP-CoAP incorporates some modifications in an attempt to enhance fairness and to adapt to CoAP and CoCoA+ constraints. Not only the resulting protocol is very complex, but we also show that BDP can outperform previous backoff-based protocols in few cases where the network conditions are prosperous.

2.6 Analysis and Shortcomings of Backoff-based Congestion Control

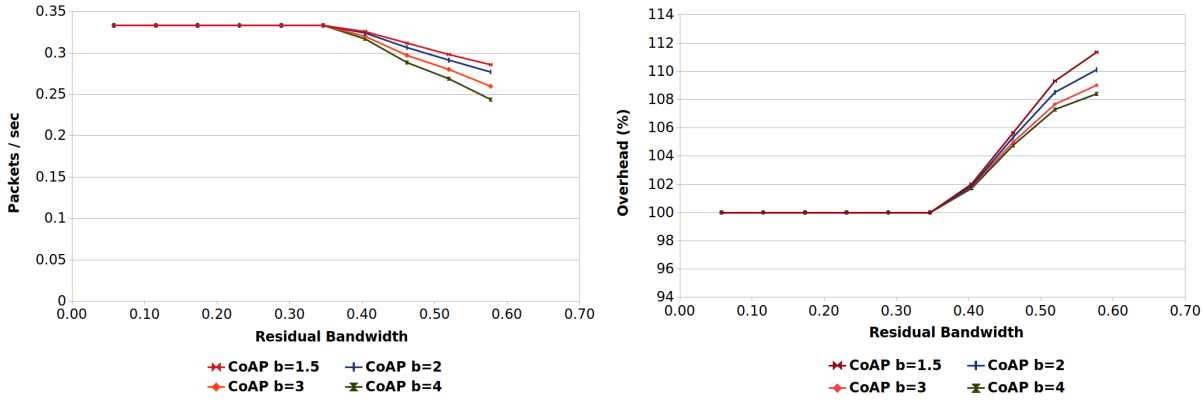
In the following, we present the issues of the main previous proposals to enhance the backoff-based mechanisms. In backoff-based Congestion Control algorithms, RTO_{init} is either calculated dynamically or set statically. Then a backoff mechanism is applied for the remaining retransmissions. In such mechanisms, the backoff factor can change depending on the previous value of RTO_{init} as in CoCoA+ and pCoCoA, or according to some state as in 4-state, or based on an exponential increase of the RTO value as per the default CoAP. Default CoAP does not use any information to adjust the backoff factor, thus it acts the same in any network condition regardless of the level of congestion, the number of forwarding nodes, the number of senders, or any other network related matters.

2.6.1 Inadequate backoff factor

We show here the analysis of CoAP backoff factors with a challenging scenario where the average residual bandwidth is fixed to 1 packet per second and we increment the standard deviation of the residual bandwidth which is varied according to a uniform distribution. The residual bandwidth changes every bad period whose length is 5 seconds. The detailed network parameters are summarized in Chapter 6 - Table 6.1. In Figure 2.10, we vary CoAP backoff factor b from 1.5 to 3 and we compute the goodput and overhead using the default CoAP mechanism suggested in [53]. The goodput is defined as the total amount of successfully received data in a given time interval while the overhead is the total amount of lost packets in the network over total amount of packets sent successfully. We notice that changing the b has no effect on goodput and overhead when the standard deviation is less than 0.4 packets per second. In such cases, the value of the backoff factor does not matter. As can be seen from Fig. 2.10a, the lower the backoff factor, the better the goodput when the bandwidth variation is greater than 0.4 packets per second. However, an opposite effect on overhead is observed when the backoff decreases. As a matter of fact, reducing the backoff will impose fast packet transmissions and retransmissions and this has a bad impact on the overhead due to packet losses. This proves that varying the backoff factor will not improve the tradeoff between performance metrics when the bandwidth is not highly fluctuating. On the other hand, it will not lead to an acceptable tradeoff between goodput and overhead when the bandwidth oscillates. We have also seen a similar propertiy using our models in chapters 3 and 4.

2.6.2 Inaccurate Variable Backoff

Remind that VBF is used by pCoCoA and CoCoA+. It is sufficient to highlight that the backoff factor should be normally related to the state of the network such as losses or goodput decreases, and not to the value of RTO as proposed by VBF. For instance, if RTO_{init} value is 1.5 seconds which might be close to RTT, and the ACK message was not received, in this case the backoff factor will be 2 according



(a) Goodput **(b)** Overhead

Figure 2.10: Simulation results - Varying CoAP backoff factor b

to CoCoA+ backoff algorithm, then the sender will unnecessarily wait for 3 seconds. A large RTT, and thus a large RTO does not signify a congestion. Besides, when RTT values are less than 1 second in the network, then if a packet is lost, the backoff factor that will be used is 3. Hence, RTO grows faster with the VBF than with the binary exponential mechanism. The waiting time will be a lot increased which affects the performance in terms of goodput and delay. The negative impact is even worse if the loss is due to interference or short congestion state. Actually, all the efforts done to reduce and optimize RTO_{init} are vanished when we use large backoff factors for no justified reasons.

2.6.3 Inappropriate Dynamic Backoff factors

4-state algorithm uses the same method for RTO calculation as CoCoA+ but differs in the backoff mechanism where 4 different backoff factors are used. The authors try to increase the throughput of their algorithm by reducing the values of the VBF and providing a different value to each state. Their results show that 4-state achieves better goodput than CoCoA+, however, it leads to more retransmissions. On one hand, the packet losses and overhead are not reduced. On the other hand, due to the limited hardware capacities of constrained IoT nodes, using many variables will require extra storage capacities.

Another important remark is that when the load increases on the nodes in any network topology, the number of retransmissions increases as well. In this case, the use of BEB and VBF to schedule retransmissions negatively affects the performance of the nodes that are far from the data collector node (or sink node). This is because the RTO estimation depends on the RTT values that are usually larger for far nodes. The problem is exacerbated when the load on the nodes and the number of retransmissions increase which results in a longer waiting time before next packet transmissions for the far nodes, thus reducing the corresponding goodput.

As a conclusion, the backoff mechansim is inefficient and the backoff factor does not have a good impact on the performance in many cases. Even though the backoff can be set depending on the congestion state of the network, it is not sufficiently fine-grained to tune precisely the retransmission timeouts during the backoff. Furthermore, the retransmission timeout is useful to detect congestion but it is hard to make it perform other roles in parallel and in particular reacting correctly to the detected congestion. Indeed, retransmission timeouts were not designed originally to control the sending rate which is supposed to be the more efficient way to control congestion.

2.7 Analysis and Shortcomings of Previous Rate-based Algorithms

The congestion avoidance algorithm proposed in TCP BBR for congestion control follows a measurement-based strategy to detect congestion and to set the sending rate adequately, instead of a more classic loss-based strategy. In this regard, the BBR congestion control can be very efficient since it aims at equating the sending rate to the available bandwidth which is somewhat the ultimate objective of any congestion control. However, it must be judicially adapted to be incorporated in the CoAP protocol which has specific properties and is destined to specific devices and network environments. It turns out that the adaptations proposed by BDP-CoAP have several shortcomings presented in the following.

2.7.1 Bandwidth Sampling Inaccuracy

BBR is designed for TCP Congestion Avoidance periods where usually the bandwidth is very high and the number of packets sent and ACK received, is very high as well, resulting in a lot of measurement samples to estimate the available bandwidth quickly and precisely. In CoAP, the sending rate is 1 message per RTT or lower and hence the number of bandwidth measurement samples is very low. As a consequence, in contrast to what is proposed by BDP-CoAP, each sample must be considered in the estimation especially those obtained at the reception of an ACK of a retransmitted CON message. These ACKs reflect also successful transmissions and bandwidth availability and must be considered. Besides, after several successive retransmissions, which means losses, the first ACK received will provide an actual measurement on the new decreased available bandwidth that causes the losses. Ignoring samples from retransmitted packets will lead to inaccurate or nonexistent bandwidth estimation if there are losses in the network. Fig. 2.11 simulates a case of bandwidth sudden decrease showing the inability of BDP-CoAP to decrease its sending rate due to successive losses despite that many ACKs are received (green line).

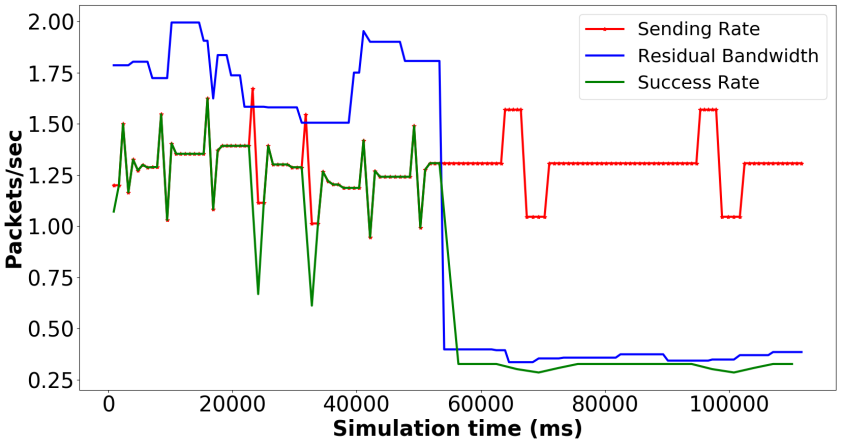


Figure 2.11: BDP-CoAP inefficiency in case of sudden bandwidth decrease

2.7.2 Inadequacy of the Bandwidth Filter Time Window

For the same reason, sliding the bandwidth measurement window over *time* is not adequate to filter the measured samples because after several losses and/or sending rate reduction, the time window will not be able to cover enough number of samples and this number can even be drawn to zero which blocks totally the protocol. BDP-CoAP uses a window that slides on the instants of sending attempts instead of time. However, this procedure does not solve the problem because in case of losses, the attempts continue and make the window sliding further, which yet removes past measurements from the filter but without adding new ones.

2.7.3 Bandwidth Delay Product Inapplicability

Similar to TCP BBR, BDP-CoAP computes the Bandwidth Delay Product and uses it to control the number of CoAP CON messages to send without waiting for their acknowledgements during an RTT. However, the CoAP concept is based on sending only one packet per RTT (NSTART=1 [53]) to keep its operation simple and avoid using a sending window and all algorithms for its management as TCP. With this constraint, packets inflight is either 0 or 1, and the Bandwidth Delay Product is always between 0 and 1. Even, if we allow NSTART to be more than one, the Bandwidth Delay Product might still be small in IoT environments due to low link data rates and small buffers.

2.7.4 Bandwidth Estimation filter Degradation

Including the minimum of the bandwidth measurement samples in the estimation filter is not adequate in terms of goodput maximization especially if the bandwidth is variable. The minimum was introduced as an attempt to improve fairness, however, the impact on the goodput is very harmful. Indeed, the

minimum will slow the convergence to the maximum available bandwidth. Furthermore, in the filter, the minimum is associated with a weight that is related to the number of retransmissions. The more the retransmissions, the higher the weight, the slower the convergence. Fig. 2.12 shows indeed the inability of BDP-COAP to converge reasonably when the available bandwidth increases suddenly. The wastage of the bandwidth is huge.

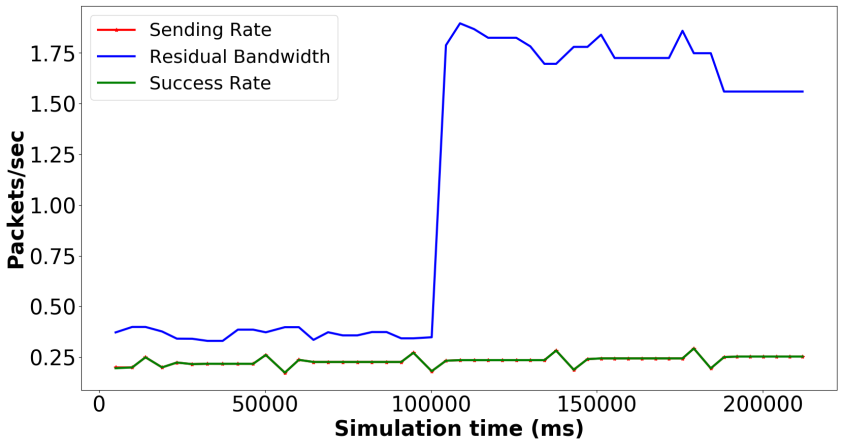


Figure 2.12: BDP-CoAP inefficiency in case of sudden bandwidth increase

2.7.5 Complexity

Even more than TCP BBR, BDP-CoAP uses many variables, instructions and function calls in order to perform bandwidth measurements and processing them. Hence, the algorithm becomes too complex. As a matter of fact, all simulations done in the BDP-CoAP work [5] have used a non-constrained type of devices while the employed simulator was designed especially for constrained devices. Again, the objective of having a good compromise between efficiency and complexity can not be ignored in IoT environments. This complexity can be reduced by removing unnecessary components that were designed for TCP and not really useful for CoAP. Also, one can use a different approach of congestion control other than a measurement-based.

2.8 Synthesis

Improving CoAP congestion control for IoT devices is clearly a challenging issue. From one side, the algorithm should be simple enough in order to be incorporated in these devices that are limited in memory and processing capacity. From the other side, the algorithm should operate efficiently to increase the transmission rate and to reduce retransmissions as much as possible to prolong the battery life. In this regard, it is important to analyze CoAP deeply and understand the effect of its parameters on performance. Accordingly, improvements on the existing approaches can be offered and new approaches can

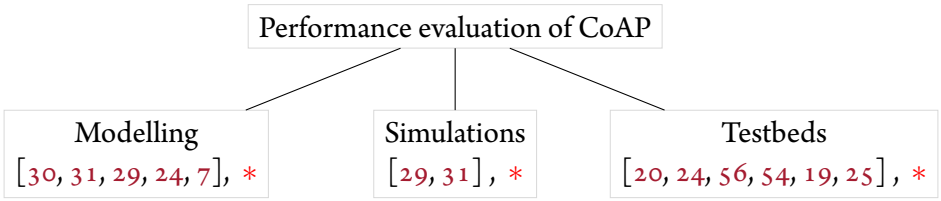


Figure 2.13: Performance evaluation of CoAP. Our contributions: *

be proposed. Deep analysis and tackling the shortcomings of previous congestion control algorithms will pave the path for introducing an improved approach for congestion control. Recently, there have been a number of research efforts related to the evaluation and improvement of CoAP. We split CoAP performance evaluation into three categories sketched in Fig. 2.13. As illustrated, different methods are considered to evaluate CoAP performance.

Some previous works tried to improve CoAP RTO calculation and its corresponding backoff mechanism. On the other hand, few works tried to propose other approaches. For instance, the authors of BDP-CoAP [5] adopted the idea of TCP BBR protocol [17] in order to control the rate of CoAP transmissions. Now, how to improve correctly the performance? How to find more efficient alternative solutions? To answer these questions, our first step is to develop a new precise analytical model in order to analyze the performance of CoAP in lossy network environments and understand thoroughly its behavior. Accordingly, we can evaluate and highlight its weaknesses. Besides, improvements for CoAP congestion control mechanisms suggested by the literature were evaluated to highlight their corresponding shortcomings.

There are two components in the congestion control algorithm designed for CoAP: RTO_{init} calculation for the first retransmission or for loss detection and a congestion counteraction mechanism for the remaining retransmissions and possibly all next transmissions. The anatomy of CoAP congestion control is presented in Fig. 2.14. For RTO calculation, if RTO is less than the Round Trip Time RTT, then the packet is falsely retransmitted due to incorrect RTO estimation causing a spurious transmission. If RTO is much larger than RTT, then the sender will wait unnecessarily causing a degradation in terms of goodput and delays of packets delivery. Therefore, the challenge in RTO estimation is to reach a good tradeoff between reliability and goodput. For congestion counteraction, based on our assessment, we conclude that the solution is to replace the backoff procedure by a rate-based control. Hence, the challenge in this part is to tune adequately the sending rate and improve the tradeoff between losses and efficiency. In addition to ensuring simplicity, another challenge while working with IoT constrained devices is reducing the energy consumption. Considering thousands of connected devices deployed in IoT network, recharging them over long periods is practically impossible. Hence, prolonging the lifetime of power resources of these devices is an achievement by itself. Furthermore, prolonging the nodes lifetime have a positive impact on the network lifetime. Therefore, the objective

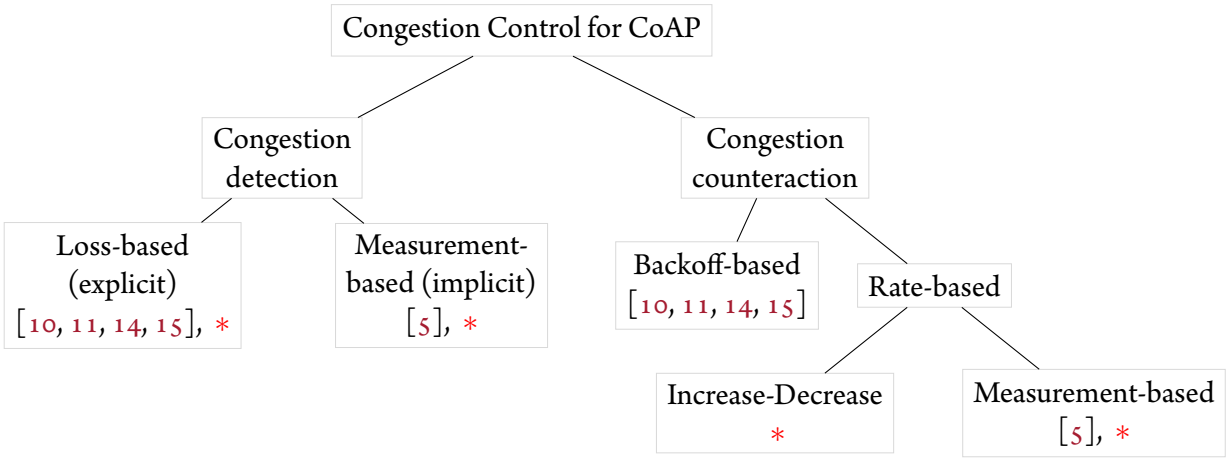


Figure 2.14: Anatomy of Congestion Control for CoAP. Our contributions: *

of our work is to provide a simple algorithm for controlling congestion that tunes the sending rate of packet transmissions and retransmissions which helps in reducing the energy wastage without impacting the overall performance. In general, a well designed algorithm should reduce energy consumption and packet losses, and improve reliability and goodput.

3

CoAP Modelling: Bernoulli and Simple Gilbert Network Loss Models

One significant aspect of CoAP is that it relies on User Datagram Protocol (UDP) but introduces a reliable mode of operation when packet loss is not tolerated. There is a cost to pay when high reliability is imposed reflected by shorter battery life and higher latency due to retransmissions. Both, low latency and low packet loss are preferred but one can be more critical than the other. For instance, in weather application, where the last measurement of the temperature reflects the current state, latency is more important than packet loss. On the other hand, for health care applications where each sensitive readout might represent critical situation of the patient, packet loss is much more important than latency. How to find a good trade-off between latency and packet lost? Another important aspect is that some IoT applications might need to impose a desired goodput in order to preserve battery life. How to ensure such aspects? The aforementioned necessitate a thorough analysis of CoAP which will be accomplished via modelling. In this chapter, we propose new exact mathematical models to study the reliability and the overall performance of CoAP in lossy networks. This study provides useful insights in order to tune CoAP parameters and also highlights CoAP properties and limitations so that better mechanisms can be designed adequately. Our analysis was carried out first in two different types of network loss models:

- Bernoulli Network Loss Model

- Simple Gilbert Network Loss Model

For each network scenario, we compute the following performance metrics:

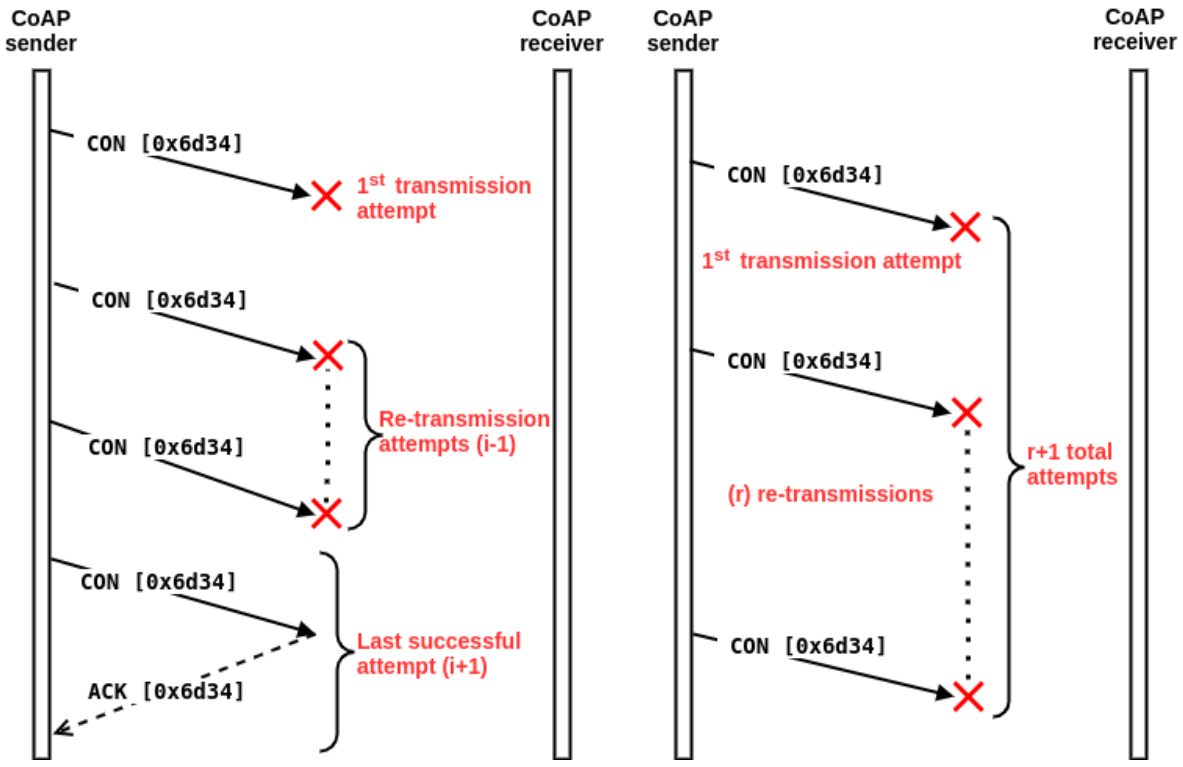
- Loss ratio
- Delay
- Goodput
- Overhead for a successful transmission

Although it is usually more practical to employ modeling tools such as Markov chains, which are used in the next chapter, we model here CoAP through "direct" computations that follow and reflect the behavior of CoAP in the presence of losses so that we understand better the protocol and find clearly the correspondence between the protocol parameters and the closed-form analytical expressions of the different performance metrics. Besides, the results obtained in this chapter will be confronted with those obtained in the next chapter with Markov chains to ensure definitely their correctness and completeness. As a matter of fact, we were so confident regarding the analytical results then when we found differences between our models and experiments done with the CoAP implementation in the Contiki operating system, we use the model to find precisely the mistake in the source code and correct it. Also, we highlight in this chapter one of the mistakes done in previous works with direct computations and provide the correct way to derive the analytical expressions of the performance metrics.

3.1 Bernoulli Network Loss Model

In a lossy network environment, the chance of a successful data packet delivery is determined by the network loss ratio. Denote p as the loss probability of a transmission attempt, $1 - p$ is then the probability of a successful transmission, r as the maximum counter of CoAP retransmission attempts. There are two scenarios when transmitting a packet:

- Successful transmission of a packet after one or more attempts presented in Fig. 3.1a. The overall probability is formalized as: $p^i(1-p), 1 \leq i \leq r$.
- Sending trials of a packet ending with failure. This is presented in Fig 3.1b. and the overall probability is: p^{r+1}.



(a) Scenario 1: Successful packet transmission

(b) Scenario 2: Packet transmission ending with failure

Figure 3.1: Different scenarios for packet transmission

3.1.1 Observed Losses

The observed losses correspond to the case where all retransmission attempts are lost and ACK is not received (Fig. 3.1b). The mathematical formula for CoAP observed loss ratio P_L in Bernoulli Loss Network Model is simply:

$$P_L = p^{r+1} \tag{3.1}$$

3.1.2 Delay

Let D_0 be the one-way delay between CoAP sender and receiver. The initial timeout value RTO denoted by T_0 is multiplied by a random factor f so that the sender waits for a time in the range $[RTO, RTO*f]$ before retransmitting the packet. The average initial timeout value is then: $T = \frac{(f+1)}{2} RTO = T_0 \frac{(f+1)}{2}$.

The delay is calculated starting from the first transmission of a new packet till it is received successfully. The duration of i successive retransmissions is defined as the time from the first transmission until the expiration of the i^{th} one while the timeout is doubled in each attempt which is computed as $T(2^i - 1), 1 \leq i \leq r$.

Let D define a random variable describing the delay of a CoAP successful packet. When a packet is successfully delivered with zero retransmissions (Fig. 3.2), the delay is denoted as d_0 and is equal

to: $d_0 = D_0 + (2^0 - 1)T_0\frac{(f+1)}{2}$, with a probability $Pr(D = d_0) = (1 - p)$. If the packet is lost

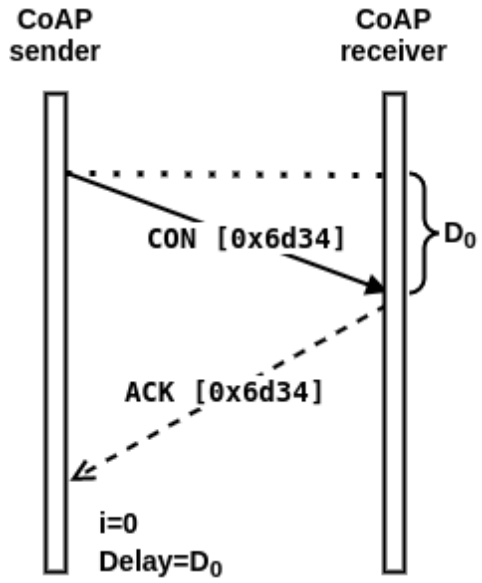


Figure 3.2: Delay without retransmissions

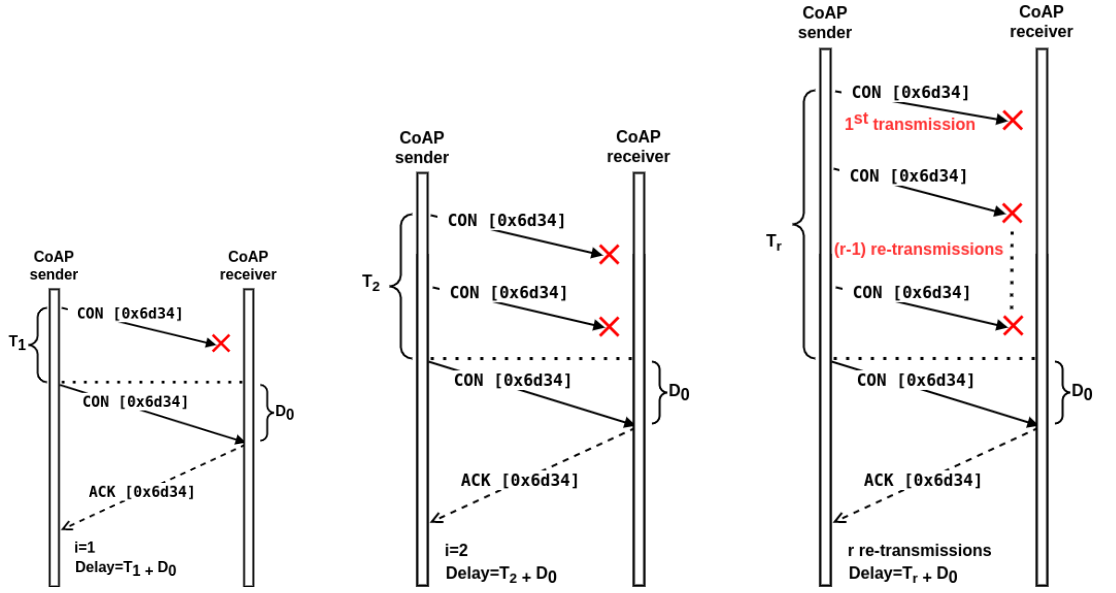
and a single retransmission is needed (Fig. 3.3a), the delay is denoted by d_1 and is equal to: $d_1 = D_0 + (2^1 - 1)T_0\frac{(f+1)}{2}$, with a probability $Pr(D = d_1) = p(1-p)$. For r retransmissions (Fig. 3.3b and Fig. 3.3c), the same applies and the delay is given by: $d_r = D_0 + (2^r - 1)T_0\frac{(f+1)}{2}$, and $Pr(D = d_r) = p^r(1 - p)$. Accordingly, the *preliminary* CoAP mathematical description for the average overall delay in the Bernoulli Loss Network Model is:

$$\bar{D} = \sum_{i=0}^{r} Pr(D = d_i) \times d_i = \sum_{i=0}^{r} p^i(1 - p)[D_0 + (2^i - 1)T_0\frac{(f + 1)}{2}] \tag{3.2}$$

The above equation ((3.2)) is similar to the one presented in [24] which is *wrong* because it is based on the geometric distribution while we must use the truncated geometric distribution. Indeed, in the CoAP context, there is a remaining scenario which is the event of a packet lost despite r retransmissions. This event can be denoted by F (failure) and then we denote its complement, i.e., the event of a successful transmission by A, with $Pr(F) = p^{r+1}$ and $Pr(A) = 1 - Pr(F) = 1 - p^{r+1}$. Consequently, $Pr(D = d_i)$ for $i = 0, 1, \ldots, r$ in equation (3.2) represent *unconditional* probabilities, and therefore should be conditioned on the event A to obtain a valid probability distribution. Normalization by the use of Bayes' Rule gives

$$Pr(D = d_i|\text{ successful transmission}) = Pr(D = d_i|A) \tag{3.3}$$

$$= \frac{Pr(\{D = d_i\} \cap A)}{Pr(A)} = \frac{Pr(D = d_i)}{Pr(A)} = \frac{p^i(1 - p)}{1 - p^{r+1}}, \tag{3.4}$$



(a) Delay with 1 retransmission **(b)** Delay with 2 retransmissions **(c)** Delay with r retransmissions

Figure 3.3: Different scenarios for packet transmission

and we obtain the probability mass function for the desired delay:

$$p_{D|A}(d_i) = \begin{cases} \frac{p^i(1-p)}{1-p^{r+1}}, & \text{for } i \in \{0, 1, \ldots, r\} \\ \\ 0, & \text{otherwise} \end{cases} \tag{3.5}$$

where $D|A$ is the truncated geometric distribution of retransmission attempts.
Now, we can check that (3.5) is valid since $\sum_{i=0}^{r} P(D = d_i|A) = \frac{\sum_{i=0}^{r} p^i(1-p)}{1-p^{r+1}} = \frac{1-p^{r+1}}{1-p^{r+1}} = 1$. Thus, our proposed CoAP model for *successful* transmission delay is now computed as follows:

$$\begin{aligned}
\bar{D}_s &= E\left[D|A\right] && (3.6)\\
&= \sum_{i=0}^{r} Pr(D = d_i|A) \times d_i && (3.7)\\
&= \sum_{i=0}^{r} \frac{p^i(1-p)}{(1-p^{r+1})} \times d_i && (3.8)\\
&= \sum_{i=0}^{r} \frac{p^i(1-p)}{(1-p^{r+1})} [D_0 + (2^i - 1)T_0 \frac{(f+1)}{2}] && (3.9)\\
&= \frac{1-p}{1-p^{r+1}} \sum_{i=0}^{r} p^i [D_0 + (2^i - 1)T] && (3.10)\\
&= \frac{\bar{D}}{1-p^{r+1}}, \text{ where } T = \frac{T_0(f+1)}{2}. && (3.11)
\end{aligned}$$

Simplifying Eq. (3.9), we obtain the closed form:

$$\bar{D}_s = \frac{1-p}{1-p^{r+1}} \sum_{i=0}^{r} p^i [D_0 + (2^i - 1)T] \tag{3.12}$$

$$= D_0 + (\frac{1-p}{1-p^{r+1}} \times \sum_{i=0}^{r} 2^i p^i \times T) - T \tag{3.13}$$

$$= D_0 + T \left[\frac{1-p}{1-p^{r+1}} \times \left(\frac{1-(2p)^{r+1}}{1-2p} \right) - 1 \right] \tag{3.14}$$

The closed form of delay can be also represented as:

$$\bar{D}_s = D_0 + \left[\frac{p[2(2^{r-1}-1)p^r - (2^r-1)p^{r-1} + 1]}{(1-2p)(1-p^r)} \right] T \tag{3.15}$$

$$= D_0 + \frac{p}{1-p^r} \left(\frac{1-(2p)^{r-1}}{1-2p} - (2^{r-1}-1)p^{r-1} \right) T \tag{3.16}$$

To analyze $\bar{D}_s$, we start by noting that it is undefined for $p = 1$ and $\lim_{p \to 1} \bar{D}_s = \infty$, since $p = 1$ means all packet transmissions will fail. Additionally, the closed form expression for $\bar{D}_s$ in (3.14) is inapplicable for $p = 0.5$, due to the ratio of the term $(1 - (2p)^{r+1})$ over $(1 - 2p)$. At this point, we refer back to (3.13) and note that for $p = 0.5 = 2^{-1}$, $\sum_{i=0}^{r} 2^i p^i = \sum_{i=0}^{r} 2^i 2^{-i} = \sum_{i=0}^{r} 1 = (r+1)$. Finally, we can rewrite the closed form expression as follows:

$$\bar{D}_s = \begin{cases} D_0 + T \left(\frac{1-p}{1-p^{r+1}} \times (r+1) - 1 \right), & \text{for } p = 0.5 \\ D_0 + T \left[\frac{1-p}{1-p^{r+1}} \times \left(\frac{1-(2p)^{r+1}}{1-2p} \right) - 1 \right], & \text{otherwise} \end{cases} \tag{3.17}$$

3.1.3 Goodput

The Goodput (GP) is defined as the total number of packets received by the CoAP endpoint during a period of time (in seconds). It can be calculated through the multiplication of the Packet Sending Rate (PSR) by the probability of successful transmission, previously denoted by $P(A)$. So, $GP = PSR \times Pr(A)$, and we remind the reader that

$$P\left(A = \{A_0 \cup A_1 \cup \cdots A_r\}\right) = \sum_{i=0}^{r} P(A_r) = \sum_{i=0}^{r} p^i (1-p) = 1 - p^{r+1}, \tag{3.18}$$

where A_i denotes the event of successful packet transmission with i required retransmissions.[1] Let D_{app} be the sending duration of 1 successful packet. Therefore, and as illustrated in Fig. 3.4, we have:

$$GP = \frac{Pr(A)}{D_{app}} \tag{3.19}$$

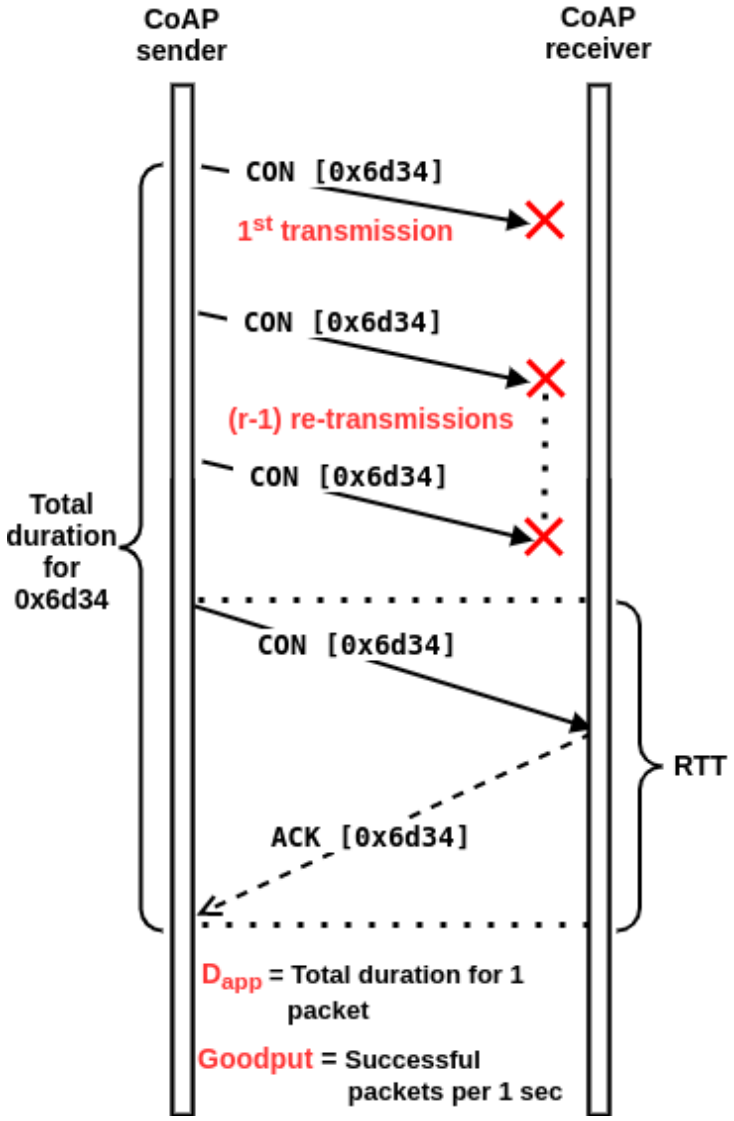


Figure 3.4: Goodput illustration

D_{app} can be considered as a random variable like the delay from the previous section, but with two main differences. Firstly, the round trip delay between CoAP sender and receiver R must be used instead of the one-way delay D_0. For instance, the duration of a successful packet transmission with k

[1] This notation will be useful later for Overhead too.

retransmission attempts is given by:

$$g_k = T + 2T + 2^2T + \cdots + 2^{k-1}T + R \tag{3.20}$$

$$= \sum_{i=0}^{k-1} 2^i T + R \tag{3.21}$$

$$= (2^k - 1)T + R, \tag{3.22}$$

where $T = T_0\frac{(f+1)}{2}$ and $Pr(D_{app} = g_k) = p^k(1-p)$ for $k = 0, 1, 2, \ldots, r$, as seen in section 3.1.2. Secondly, D_{app} should include the case when the last retransmission attempt is not successful, which corresponds to a delay $D_{app} = g_{r+1} = (2^{r+1} - 1)T$ with $Pr(D_{app} = g_{r+1}) = p^{r+1}$.

Thus, the average D_{app} is computed as

$$\bar{D}_{app} = E[D_{app}] = \sum_{i=0}^{r} Pr(D_{app} = g_i)g_i + Pr(D_{app} = g_{r+1})g_{r+1} \tag{3.23}$$

$$= \sum_{i=0}^{r} p^i(1-p)[(2^i - 1)T + R] + p^{r+1}(2^{r+1} - 1)T. \tag{3.24}$$

Equation (3.24) is very similar to (3.17). Taking advantage of the analysis in section 3.1.2, multiplying the **first term** in (3.24) by $1 - p^{r+1}$ in both the numerator and denominator allows for a simplification:

$$\bar{D}_{app} = \frac{1 - p^{r+1}}{1 - p^{r+1}} \sum_{i=0}^{r} p^i(1-p)[(2^i - 1)T + R] + p^{r+1}(2^{r+1} - 1)T, \tag{3.25}$$

giving us the closed form:

$$\bar{D}_{app} = \begin{cases} (1 - p^{r+1})\left[R + \left(\frac{1-p}{1-p^{r+1}} \times (r+1) - 1\right)T\right] + p^{r+1}(2^{r+1} - 1)T, & \text{for } p = 0.5 \\ (1 - p^{r+1})\left[R + \left(\frac{1-p}{1-p^{r+1}} \times \left(\frac{1-(2p)^{r+1}}{1-2p}\right) - 1\right)T\right] + p^{r+1}(2^{r+1} - 1)T, & \text{otherwise} \end{cases} \tag{3.26}$$

which reduces to,

$$\bar{D}_{app} = \begin{cases} (1 - 0.5^{r+1})R + \frac{T}{2}(r+1), & \text{for } p = 0.5 \\ (1 - p^{r+1})R + p\left(\frac{1-(2p)^{r+1}}{1-2p}\right)T, & \text{otherwise} \end{cases} \tag{3.27}$$

$\bar{D}_{app}$ can be calculated using a different method by adding the product of the success probability by

RTT and the loss probability by T:

$$\bar{D}_{app} = (1 - p^{r+1})RTT + T\sum_{i=0}^{r} p^{i+1}2^i. \tag{3.28}$$

In conclusion, we obtain the following formula for Goodput:

$$GP = \frac{P(A)}{E[D_{app}]} = \frac{1 - p^{r+1}}{\bar{D}_{app}} \tag{3.29}$$

$$= \frac{1 - p^{r+1}}{(1 - p^{r+1})R + p\left(\frac{1-(2p)^{r+1}}{1-2p}\right)T} \tag{3.30}$$

$$= \frac{1}{R} + \frac{(1 - 2p)(1 - p^{r+1})}{p(1 - (2p)^{r+1})T} \tag{3.31}$$

3.1.4 Overhead for a successful transmission

We denote by Y the data payload to be sent between the sender and the receiver. The total size of the packet including the headers added at different layers is denoted by Z. The total overhead is expressed as the non-application bits divided by the payload of the packet received Y. Normally, the total overhead depends only on the number of lost packets which means on the observed network loss ratio, in our case, P_L. Let us detail its computation below. For a successful transmission from the first attempt, the overhead for receiving the payload is:

$$\frac{(Z - Y)}{Y}, \text{with } Pr(A_0) = (1 - p)$$

If the first transmission attempt is not successful, then the overhead is:

$$\frac{(Z + Z - Y)}{Y}, \text{with } Pr(A_1) = p(1 - p)$$

When the packet requires r retransmission attempts to have a successful transmission, the fraction will be:

$$\frac{((r + 1)Z - Y)}{Y}, \text{with } Pr(A_r) = p^r(1 - p)$$

Finally, when $r + 1$ transmissions fail, we will have no application bits and thus we have only non-application bits:

$$(r + 1)Z, \text{with } Pr(F) = p^{r+1}$$

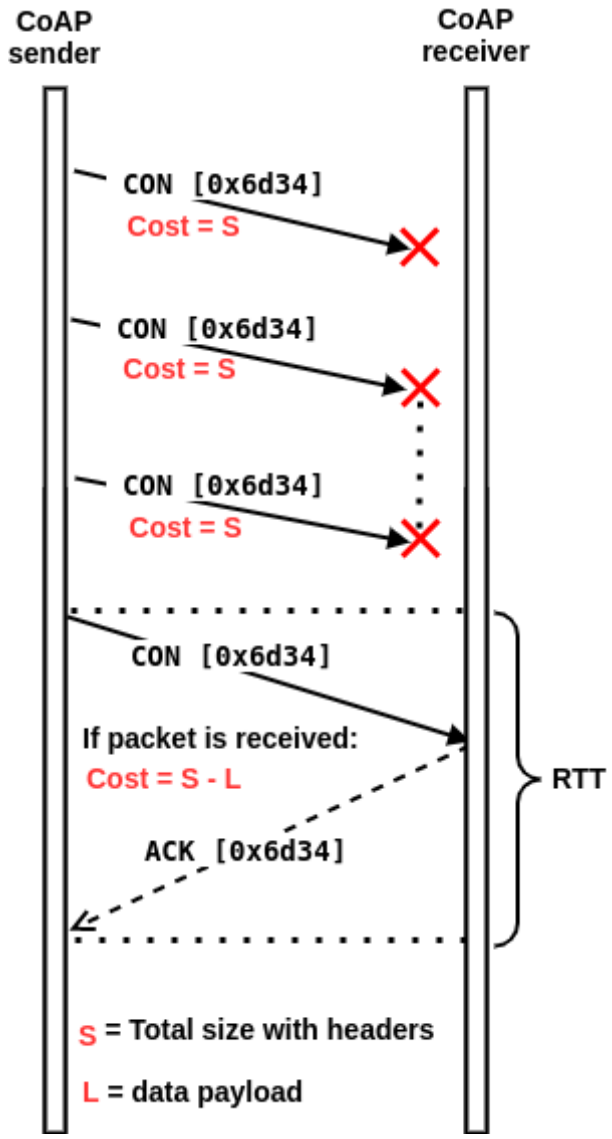


Figure 3.5: Overhead illustration

Taking the ratio of averages, the overall average Overhead expression is given by:

$$\frac{\sum_{i=0}^{r} p^i(1-p)[(1+i)Z - Y] + Z(r+1)p^{r+1}}{\sum_{i=0}^{r} p^i(1-p)Y + 0 \times p^{r+1}}, \tag{3.32}$$

where $\sum_{i=0}^{r} p^i(1-p)[(1+i)Z - Y]$ refers to the retransmission attempts of non-application bits, $Z(r+1)p^{r+1}$ corresponds to the r-th retransmission when an ACK is not received, and $\sum_{i=0}^{r} p^i(1-p)Y$ corresponds to the retransmission attempts of data payload. Since $\sum_{i=0}^{r} p^i(1-p) = 1 - p^{r+1}$, Eq. (3.32) becomes:

$$\frac{\sum_{i=0}^{r} p^i(1-p)[(1+i)Z - Y] + Z(r+1)p^{r+1}}{(1-p^{r+1})Y} \tag{3.33}$$

$$= \frac{\sum_{i=0}^{r} p^i(1-p)(Z-Y)}{(1-p^{r+1})Y} + \frac{\sum_{i=0}^{r} Z \times i \times p^i(1-p) + Z(r+1)p^{r+1}}{(1-p^{r+1})Y} \tag{3.34}$$

$$= \frac{Z-Y}{Y} + \frac{Z}{Y}\left(\frac{(1-p)\sum_{i=0}^{r} i \times p^i + (r+1)p^{r+1}}{(1-p^{r+1})}\right). \tag{3.35}$$

Now, using the following identity:

$$(1-p) \times \sum_{i=0}^{r} i \times p^i = \frac{rp^{r+2} - (r+1)p^{r+1} + p}{(1-p)}, \tag{3.36}$$

replacing (3.36) in (3.35) changes the latter to:

$$\frac{Z-Y}{Y} + \frac{Z}{Y}\left[\frac{rp^{r+2} - (r+1)p^{r+1} + p}{(1-p)(1-p^{r+1})} + \frac{(r+1)p^{r+1}}{(1-p^{r+1})}\right] \tag{3.37}$$

Factoring out $\frac{p^{r+1}}{(1-p^{r+1})}$ in (3.37) we get

$$\frac{Z-Y}{Y} + \frac{Z}{Y}\left[\left(\frac{p^{r+1}}{1-p^{r+1}}\right)\left(\frac{rp - (r+1) + p^{-r}}{1-p} + r + 1\right)\right] \tag{3.38}$$

$$= \frac{Z-Y}{Y} + \frac{Z}{Y}\left[\left(\frac{p^{r+1}}{1-p^{r+1}}\right)\left(\frac{rp - (r+1) + p^{-r} + (r+1)(1-p)}{1-p}\right)\right] \tag{3.39}$$

$$= \frac{Z-Y}{Y} + \frac{Z}{Y}\left[\left(\frac{p^{r+1}}{1-p^{r+1}}\right)\left(\frac{rp - p(r+1) + p^{-r}}{1-p}\right)\right] \tag{3.40}$$

$$= \frac{Z-Y}{Y} + \frac{Z}{Y}\left[\left(\frac{p^{r+1}}{1-p^{r+1}}\right)\left(\frac{-p + p^{-r}}{1-p}\right)\right] \tag{3.41}$$

$$= \frac{Z-Y}{Y} + \frac{Z}{Y}\left[\left(\frac{p^{r+1}}{1-p^{r+1}}\right)\left(\frac{p}{1-p}\right)\left(p^{-(r+1)} - 1\right)\right] \tag{3.42}$$

$$= \frac{Z-Y}{Y} + \frac{Z}{Y}\left[\left(\frac{p^{r+1}}{1-p^{r+1}}\right)\left(\frac{p}{1-p}\right)\left(\frac{1-p^{r+1}}{p^{r+1}}\right)\right] \tag{3.43}$$

$$= \frac{Z-Y}{Y} + \frac{Z}{Y}\left(\frac{p}{1-p}\right) \tag{3.44}$$

$$= \frac{Z-(1-p)Y}{(1-p)Y}. \tag{3.45}$$

As expected, we observe that the overhead depends only on p. However, the overhead for a successful transmission will include a CoAP parameter. In this case, we should normalize in the numerator and denominator by $1-p^{r+1}$ in equation (3.32) to adjust the probability values after removing the factor $Z(r+1)p^{r+1}$ which refers to the last unsuccessful transmission attempt. Thus, we obtain the overhead for a successful transmission O_S:

$$O_S = \frac{\sum_{i=0}^{r} p^i(1-p)[(1+i)Z - Y]}{\sum_{i=0}^{r} p^i(1-p)Y}, \tag{3.46}$$

$$= \frac{\sum_{i=0}^{r} p^i(1-p)(Z-Y)}{(1-p^{r+1})Y} + \frac{Z(1-p)\sum_{i=0}^{r} p^i \times i}{(1-p^{r+1})Y}, \tag{3.47}$$

$$= \frac{Z-Y}{Y} + \frac{Z(1-p)\sum_{i=0}^{r} p^i \times i}{Y(1-p^{r+1})}. \tag{3.48}$$

As before, using the identity (3.36) in (3.48)

$$O_S = \frac{Z-Y}{Y} + \frac{Z}{Y}\left(\frac{rp^{r+2} - (r+1)p^{r+1} + p}{(1-p^{r+1})(1-p)}\right) \tag{3.49}$$

$$= \frac{Z-Y}{Y} + \frac{Z}{Y}\left(\frac{p^{r+1}(rp - (r+1) + p^{-r})}{(1-p^{r+1})(1-p)}\right), \tag{3.50}$$

$$= \frac{Z-Y}{Y} + \frac{Z}{Y}\left(\frac{p^{r+1}}{1-p^{r+1}}\right)\left(\frac{p^{-r} - 1 - (1-p)r}{1-p}\right), \tag{3.51}$$

$$= \frac{Z-Y}{Y} + \frac{Z}{Y}\left(\frac{p^{r+1}}{1-p^{r+1}}\right)\left(\frac{p^{-r} - 1}{1-p} - r\right). \tag{3.52}$$

Given that $p^{-r} - 1 = \frac{1-p^r}{p^r}$, we get the following closed form of overhead for successful transmission:

$$O_S = \frac{Z-Y}{Y} + \frac{Z}{Y}\left(\frac{p^{r+1}}{1-p^{r+1}}\right)\left(\frac{1-p^r}{p^r(1-p)} - r\right), \tag{3.53}$$

$$O_S = \frac{Z-Y}{Y} + \frac{Z}{Y} \times \frac{P(F)}{P(A)} \times \left(\frac{P(A_0 \cup A_1 \cup \ldots \cup A_{r-1})}{P(A_r)} - r\right). \tag{3.54}$$

Let R be a random variable defined as the ratio of overhead by the payload. R takes values from the set $\{\frac{Z-Y}{Y}, \frac{2S-Y}{Y}, \ldots, \frac{(r+1)Z-Y}{Y}\} = \{\frac{(1+i)Z-Y}{Y}\}_{i\in\{0,1,\ldots,r\}}$. Again, because we ignore the case with no ACK reception, the associated probabilities have to be conditioned on the event of successful reception A. Then we have:

$$\sum_{i=0}^{r} \frac{p^i(1-p)}{1-p^{r+1}} \times \frac{(1+i)Z-Y}{Y} \tag{3.55}$$

$$= \sum_{i=0}^{r} \frac{p^i(1-p)}{1-p^{r+1}} \times \frac{Z-Y}{Y} + \frac{Z(1-p)}{Y}\sum_{i=0}^{r} \frac{p^i \times i}{1-p^{r+1}} \tag{3.56}$$

$$= \frac{Z-Y}{Y} + \frac{Z(1-p)}{Y}\sum_{i=0}^{r} \frac{p^i \times i}{1-p^{r+1}}, \tag{3.57}$$

and as before, using (3.36) we find that:

$$\frac{Z-Y}{Y} + \frac{Z}{Y}\frac{(rp^{r+2} - (r+1)p^{r+1} + p)}{(1-p)(1-p^{r+1})} \tag{3.58}$$

$$= \frac{Z-Y}{Y} + \frac{Z}{Y}\left(\frac{p^{r+1}}{1-p^{r+1}}\right)\left(\frac{rp - (r+1) + p^{-r}}{1-p}\right) \tag{3.59}$$

$$= \frac{Z-Y}{Y} + \frac{Z}{Y}\left(\frac{p^{r+1}}{1-p^{r+1}}\right)\left(\frac{p^{-r}-1}{1-p} - r\right) \tag{3.60}$$

$$\tag{3.61}$$

3.2 Simple Gilbert Network Loss Model

As we have seen in Section 3.1, the Bernoulli network loss model has only one state and one parameter which is the loss probability p. As a matter of fact, the Bernoulli model is the simplest case of loss model and is used to define and model uncorrelated loss events and thus has its own limitations. In this section, we present the Simple Gilbert model which has two states (Good and Bad) and two independent parameters (p and q). While in the good state, there is very few packet loss; while in the bad state, most packets are lost. Such models provide the flexibility to model a network with consecutive loss events. Simple Gilbert network loss model offers a good approximation of packet losses and is widely used. This model is presented in Fig. 3.6. As per the figure, "0" is the good state and "1" is the bad state. Also, p is the transition probability from state "0" to state "1", q the transition probability from state "1" to state "0". Thus, $1-p$ and $1-q$ are the probabilities of staying within the same state. Also, $1-q$ is the probability of having successive loss which clearly impacts on retransmissions and thus CoAP losses.

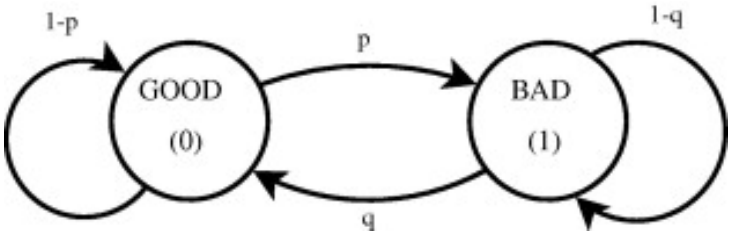


Figure 3.6: Simple Gilbert Network

3.2.1 Observed Losses

The observed CoAP loss ratio in a network modelled with Simple Gilbert equals to the average number of observed CoAP message losses at the application layer divided by the total number of total CoAP transmitted messages which includes the average number of observed losses and successful transmissions (Fig. 3.7). We define N_{OL} to be the random variable that refers to the number of observed losses, where N_{OL} takes values from the set $\mathbb{N}^+ - \{0\} = \{1, 2, 3, 4, \ldots\}$. Hence, we write the Observed Loss Ratio P_L as:

$$P_L = \frac{E[N_{OL}]}{E[N_{OL}]+1} \tag{3.62}$$

where $E[N_{OL}]$ is the average of the observed losses presented in Fig. 3.7. The figure shows a typical scenario in CoAP transmission, where the period of time between the reception of two successive ACKs

consists of many blocks of CoAP lost packets (OL), followed by an event of successful transmission. The number of said blocks is random, and its average value is denoted by $E[N_{OL}]$. During the"Loss Period", each block constitutes an observed loss and refers to $r+1$ total transmission attempts. On the other hand, the"Successful Period" is a single block. As the figure indicates, successful transmission of a packet usually entails a number of re-transmission attempts, before the eventual reception of an ACK. Specifically, multiple CON messages may be sent until the source receives the confirmation message from its intended destination.

To compute $E[N_{OL}]$, we must determine the probabilities of the random variable N_{OL} ($Pr(N_{OL} = 1) \dots Pr(N_{OL} = i), 1 \leq i \leq \infty$). For one observed loss, there are $r+1$ potential cases and they are presented in the table below with $r = 4$.

Case No	Transition Dynamics (State)	Probability
1	$0 \to 1 \to 1 \to 1 \to 1 \to 1 \quad (OL)$	$p(1-q)^r$
2	$0 \to 1 \to 1 \to 1 \to 1 \to 1 \to 0$	$p(1-q)^r q$
3	$0 \to 1 \to 1 \to 1 \to 1 \to 1 \to 1 \to 0$	$p(1-q)^r(1-q)q$
4	$0 \to 1 \to 1 \to 1 \to 1 \to 1 \to 1 \to 1 \to 0$	$p(1-q)^r(1-q)^2q$
5	$0 \to 1 \to 1 \to 1 \to 1 \to 1 \to 1 \to 1 \to 1 \to 0$	$p(1-q)^r(1-q)^3q$
6	$0 \to 1 \to 1 \to 1 \to 1 \to 1 \to 1 \to 1 \to 1 \to 1 \to 0$	$p(1-q)^r(1-q)^4q$

Table 3.1: Cases for observation of one CoAP loss with $r = 4$

To observe two losses, we start by a single loss and observe one more. This scenario also has $r+1$ cases, and they are presented in the table below. Accordingly, we write: $Pr(N_{OL} = i) = \sum_{j=0}^{r} qp(1-$

Case No	Transition Dynamics (State)	Probability
1	$0 \to 1 \to \dots \to 1 \to 1 \to 1 \to 1 \to 1 \quad (2OL_s)$	$p(1-q)^r(1-q)^{r+1}$
2	$0 \to 1 \to \dots \to 1 \to 1 \to 1 \to 1 \to 1 \to 0$	$p(1-q)^{2r+1}q$
3	$0 \to 1 \to \dots \to 1 \to 1 \to 1 \to 1 \to 1 \to 1 \to 0$	$p(1-q)^{2r+1}(1-q)q$
4	$0 \to 1 \to \dots \to 1 \to 1 \to 1 \to 1 \to 1 \to 1 \to 1 \to 0$	$p(1-q)^{2r+1}(1-q)^2q$
5	$0 \to 1 \to \dots \to 1 \to 1 \to 1 \to 1 \to 1 \to 1 \to 1 \to 1 \to 0$	$p(1-q)^{2r+1}(1-q)^3q$
6	$0 \to 1 \to \dots \to 1 \to 1 \to 1 \to 1 \to 1 \to 1 \to 1 \to 1 \to 1 \to 0$	$p(1-q)^{2r+1}(1-q)^4q$

Table 3.2: Cases for observation of two CoAP losses with $r = 4$.

$q)^{i(r+1)+j-1}, 1 \leq i \leq \infty$ Thus, the average number of losses is given by

$$E[N_{OL}] = \sum_{i=1}^{\infty} Pr(N_{OL} = i) \times i \tag{3.63}$$

$$= \sum_{i=1}^{\infty} \left(\sum_{j=0}^{r} qp(1-q)^{i(r+1)+j-1} \right) \times i \tag{3.64}$$

$$= qp \sum_{i=1}^{\infty} i \times (1-q)^{i(r+1)} \sum_{j=0}^{r} \frac{(1-q)^j}{(1-q)} \tag{3.65}$$

$$= \frac{qp}{1-q} \sum_{i=1}^{\infty} i \times (1-q)^{i(r+1)} \times \frac{1-(1-q)^{r+1}}{1-(1-q)} \tag{3.66}$$

$$= \frac{p(1-(1-q)^{r+1})}{1-q} \sum_{i=1}^{\infty} i \times (1-q)^{i(r+1)}. \tag{3.67}$$

Using the following identity:

$$\sum_{i=1}^{\infty} i \times x^i = \frac{x}{(x-1)^2} \text{ where } |x| < 1, \tag{3.68}$$

substituting $(1-q)^{(r+1)}$ for x simplifies (3.67) to

$$E[N_{OL}] = \frac{p(1-q)^r}{1-(1-q)^{r+1}}. \tag{3.69}$$

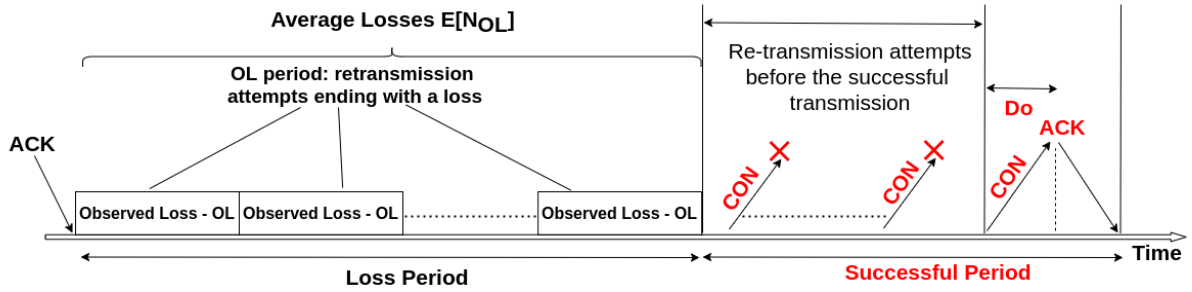


Figure 3.7: Loss Period in Simple Gilbert Network

Substituting equations (3.69) in (3.62), we deduce our mathematical formula for CoAP observed loss ratio P_L for the Simple Gilbert network loss model as follows:

$$P_L = \frac{\frac{p(1-q)^r}{1-(1-q)^{r+1}}}{1+\frac{p(1-q)^r}{1-(1-q)^{r+1}}} \tag{3.70}$$

$$= \frac{p(1-q)^r}{1-(1-q)^{r+1}+p(1-q)^r} \tag{3.71}$$

$$= \frac{p(1-q)^r}{1-(1-q)^r(1-q-p)} \tag{3.72}$$

$$= \frac{p(1-q)^r}{1+(p+q-1)(1-q)^r}. \tag{3.73}$$

Remark: In the special case where $q = 1-p$, substituting this value of q in (3.73) yields $P_L = (1-q)p^r = pp^r = p^{r+1}$, which equals the loss probability in the Bernoulli Loss Network Model.

3.2.2 Delay

In this section we consider the delay of a successful packet transmission. According to Figure 3.7, we notice that the total delay must include the time incurred due to observed losses in the "Loss Period" (LP), in addition to delays from re-transmission attempts before the successful transmission in the "Successful Period" (SP). We denote the former by D_{LP} and the latter by D_{SP}. Hence, the delay formula is the sum of these terms:

$$\bar{D} = D_{LP} + D_{SP}, \tag{3.74}$$

$$\implies E[D] = E[D_{LP}] + E[D_{SP}], \tag{3.75}$$

by linearity of the expectation operator.

The first term D_{LP} refers to the case of a "Loss Period", where the packet is dropped at least once. Therefore, this delay is a function of the number of blocks of observed losses (or N_{OL}), where the cost of a single block is given by the following:

$$D_{OL} = \sum_{k=0}^{r} 2^k T = (2^{r+1} - 1)T. \tag{3.76}$$

For instance, if $N_{OL} = i$, we have a delay denoted as

$$d_i = i \times D_{OL} = i \times (2^{r+1} - 1)T,$$

$$\text{with } Pr(D_{LP} = d_i) = P(N_{OL} = i) = \sum_{j=0}^{r} qp(1-q)^{i(r+1)+j-1}.$$

Thus, considering all different scenarios, that is, for all values of $N_{OL} \geq 1$, we obtain the first term as:

$$E[D_{LP}] = \sum_{i=1}^{\infty} P(D_{LP} = d_i) \times d_i \tag{3.77}$$

$$= \underbrace{\sum_{i=1}^{\infty} \sum_{j=0}^{r} qp(1-q)^{i(r+1)+j-1} \times i}_{\mathrm{E}[N_{OL}]} \left[\underbrace{(2^{r+1} - 1)T}_{\text{delay per block}} \right] \tag{3.78}$$

$$= E[N_{OL}] \times D_{OL}. \tag{3.79}$$

As for the second term, it includes D_0 and any delay incurred by retransmisson attempts before eventual success, where the number of re-transmission attempts i is such that $1 \leq i \leq r$. For instance, transmission after i failures corresponds to a switch from a good state to a bad state, remaining in the

bad state $i-1$ times, and finally resting on the good state. So, the incurred delay is:

$$(2^i - 1)T, \text{with probability } p(1-q)^{(i-1)}q. \tag{3.80}$$

Thus, the delay in the "Success Period" is written as

$$E[D_{SP}] = D_0 + pq\sum_{i=1}^{r}(1-q)^{i-1}(2^i-1)T \tag{3.81}$$

$$= D_0 + \frac{pq}{1-q}\sum_{i=1}^{r}(1-q)^{i}(2^i-1)T \tag{3.82}$$

$$= D_0 + \frac{pq}{1-q}\left(\frac{2(1-q)(1-2^r(1-q)^r)}{2q-1} - \frac{(1-q)(1-(1-q)^r)}{q}\right)T \tag{3.83}$$

$$= D_0 + pq\left(\frac{2(1-2^r(1-q)^r)}{2q-1} - \frac{1-(1-q)^r}{q}\right)T \tag{3.84}$$

Finally, the delay is given by the following:

$$\bar{D} = E[N_{OL}]D_{OL} + D_0 + pq\left(\frac{2(1-2^r(1-q)^r)}{2q-1} - \frac{1-(1-q)^r}{q}\right)T \tag{3.85}$$

Simplifying further, the mathematical notation for delay $\bar{D}$ is obtained:

$$\bar{D} = D_0 + \left(\frac{pq}{1-(1-q)^{r+1}}\right)\left(\frac{2q\,(1-2^r(1-q)^r)}{2q-1} + (1-q)^r - 1\right)T \tag{3.86}$$

$$= D_0 + \frac{pq}{1-(1-q)^{r+1}}\sum_{i=1}^{r}(1-q)^{i-1}(2^i-1)T \tag{3.87}$$

which will be useful in the latter section.

3.2.3 Goodput

As a reminder, the goodput is defined as the rate of packets received by the CoAP endpoint during a certain duration (in seconds). Equivalently, the goodput is the inverse of the duration between two successful packet deliveries, with a unit measured in packet per second. In a bursty network, the period of time between reception of two ACKs is usually interspersed with multiple observed losses and/or lost ACKs messages from the destination. Accordingly, the duration between two successful deliveries includes firstly, the delay incurred by the observed losses that might occur, and secondly, the delay of successful transmission. The former is equal to the average number of observed losses $E[N_{OL}]$ multiplied by the delay of each loss D_{OL}, while the latter is simply the delay presented in section 3.2.2, with a minor change. Namely, we must replace the one-way delay D_0 in (3.87) by the round-trip delay

between the sender and receiver R. Thus, the mathematical notation of the total duration is given as:

$$Duration = E[N_{OL}]D_{OL} + \bar{D} \tag{3.88}$$

The delay of each observed loss OL is

$$D_{OL} = (2^{r+1} - 1)T. \tag{3.89}$$

Substituting the corresponding values, the duration is presented as

$$\frac{p(1-q)^r}{1-(1-q)^{r+1}}(2^{r+1}-1)T + R + \frac{pq}{1-(1-q)^{r+1}}\sum_{i=1}^{r}(1-q)^{i-1}(2^i-1)T \tag{3.90}$$

Since the duration is calculated as per (3.88), then multiplying by $1 - p^{r+1}$ (as in Section 3.1.3) is not needed in this case to obtain the Goodput. In conclusion, we obtain the following formula for Goodput:

$$GP = \frac{1}{Duration} \tag{3.91}$$

$$= \frac{1-(1-q)^{r+1}}{p(1-q)^r(2^{r+1}-1)T + (1-(1-q)^{r+1})R + pq\sum_{i=1}^{r}(1-q)^{i-1}(2^i-1)T} \tag{3.92}$$

Here again, replacing $1 - q$ by p changes the context of study from the Simple Gilbert model to the Bernoulli model of losses seen before. Accordingly, the equation in (3.92) should match with (3.29) when $1 - q = p$. Indeed, equation (3.92) becomes:

$$GP = \frac{1-p^{r+1}}{p^{r+1}(2^{r+1}-1)T + (1-p^{r+1})RTT + p(1-p)\sum_{i=1}^{r}(p)^{i-1}(2^i-1)T} \tag{3.93}$$

$$= \frac{1-p^{r+1}}{p^{r+1}(2^{r+1}-1)T + (1-p^{r+1})RTT + \sum_{i=0}^{r}p^i(1-p)(2^i-1)T} \tag{3.94}$$

$$= \frac{1-p^{r+1}}{p^{r+1}(2^{r+1}-1)T + \sum_{i=0}^{r}p^i(1-p)RTT + \sum_{i=1}^{r}p^i(1-p)(2^i-1)T} \tag{3.95}$$

$$= \frac{1-p^{r+1}}{p^{r+1}(2^{r+1}-1)T + \sum_{i=0}^{r}p^i(1-p)\left[(2^i-1)T + RTT\right]} \tag{3.96}$$

$$= \frac{P(A)}{\bar{D}_{app}} = \text{GP obtained in 3.29} \tag{3.97}$$

3.2.4 Overhead for a successful transmission

The overhead is defined as:

$$\frac{N_A \times Z - Y}{Y}, \tag{3.98}$$

where N_A denotes the total number of transmission attempts. This number is the summation of the number of:

1. Attempts in a loss period, which is $(r+1) \times E[N_{OL}]$
2. Attempts before the successful transmission, denoted as B.

$$B = p \sum_{i=1}^{r} \frac{q}{1-(1-q)^{r+1}} (1-q)^{i-1} i \tag{3.99}$$

3. Attempt that is successful, which is 1.

The equation of B is similar to the simplified equation of delay in section 3.2.2 but instead of using the retransmission delay of the i^{th} attempt: $(2^i - 1)T_0 \frac{(f+1)}{2}$, we use the number of attempts i. As discussed in section 3.2.1, the average number of OLs $E[N_{OL}]$ is:

$$\frac{p(1-q)^r}{1-(1-q)^{r+1}} \tag{3.100}$$

Therefore, the overall overhead formula is given as:

$$\frac{\left[\frac{p(1-q)^r}{1-(1-q)^{r+1}}(r+1) + p\sum_{i=1}^{r} \frac{q}{1-(1-q)^{r+1}}(1-q)^{i-1} i + 1\right] * Z - Y}{Y} \tag{3.101}$$

By simplifying the above formula, we obtain the overall overhead in Simple Gilbert Network:

$$O = \frac{(p+q)Z - qL}{qL} \tag{3.102}$$

$$= \frac{Z - \pi(0) \times Y}{\pi(0) \times Y}, \text{where } \pi(0) = \frac{q}{p+q}. \tag{3.103}$$

Again, to check for the validity of our proposed model, we set $q = 1 - p$ in (3.102):

$$O = \frac{(p+q)Z - qL}{qL} \tag{3.104}$$

$$= \frac{Z - (1-p)Y}{(1-p)Y} = \text{Overhead obtained in (3.45)} \tag{3.105}$$

As we have seen before, the overhead for a successful transmission will include a CoAP parameter. Here, the overhead for successful transmission is calculated by removing the average number of attempts for observed losses in loss period $E[N_{OL}](r+1)$. Hence, the mathematical notation is presented by:

$$O_{ST} = \frac{(1 + p\sum_{i=1}^{r} \frac{q}{1-(1-q)^{r+1}}(1-q)^{i-1} i) * Z - Y}{Y} \tag{3.106}$$

Simplifying Eq. (3.106), we obtain the closed form:

$$O_{ST} = \frac{\left(1 + \frac{pq}{(1-q)(1-(1-q)^{r+1})} \cdot \frac{(1-q)(r(1-q)^{r+1}-(r+1)(1-q)^{r}+1)}{q^2}\right) * Z - Y}{Y} \tag{3.107}$$

$$O_{ST} = \frac{\left(1 + \frac{p(r(1-q)^{r+1}-(r+1)(1-q)^{r}+1)}{(1-(1-q)^{r+1})q}\right) * Z - Y}{Y} \tag{3.108}$$

$$O_{ST} = \frac{\left(1 + \frac{p(1-qr(1-q)^{r}-(1-q)^{r})}{(1-(1-q)^{r+1})q}\right) * Z - Y}{Y} \tag{3.109}$$

3.3 Experimental Environment

Our proposed models are validated by comparing their results with those obtained from experiments. To do so, in this chapter we used network emulation. The nodes were emulated using Linux virtualization network stack for which we used Ubuntu (v18.04 LTS). The emulated environment consists of virtual nodes *netns* [57] and procedures to connect the nodes *veth* [12] and *netem* [28]. Real protocols and applications run on the emulated nodes that form the emulated network. In our network model, the server is operating on one node whereas the client is running on another node within the same network.

The advantage of the netem tool is that it can generate losses according to several models including Bernoulli and Simple Gilbert. It emulates also delays through propagation delays and bandwidth emulation. In our experiments, the RTT value between the server and client node is 200 ms. The loss ratio p values are chosen to be: 10%, 30% and 50% respectively. The value of q is equal to 50%. The network parameters are summarized in table 3.3.

Description	**Value**
one-way delay D_0	100 ms
RTT	200 ms
Loss Ratio p	10-50%
q (Bursty network)	50%

Table 3.3: Network parameters

We implemented the CoAP server and client using the open source library libcoap [8]. Libcoap is a c-based project that provides the environment with CoAP functions. The client and the server were pushed to the emulated nodes to run the experiments. In both network scenarios, the traffic was created by CoAP client via sending "PUT" requests of type CON to the server. CoAP server responds by sending ACK message to each CON request. The client sends the PUT request for 10000 times consecutively. Different CoAP parameters are used and summarized in table 3.4.

Description	Value
RTO	0.5 – 1 – 1.5 – 2 – 2.5 s
r	1 – 9
Z	62 – 100 bytes
Y	10 – 48 bytes

Table 3.4: CoAP parameters

3.4 Models Validation and Performance Evaluation via experiments

3.4.1 Bernoulli Loss Model

Our observed loss ratio in the Bernoulli loss network model is p^{r+1}, hence increasing r should as a consequence decrease this metric. To check the accuracy of our model, we compared it to the experimental results shown in Fig. 3.8. The graph shows that, despite its simplicity, the observed losses model produces similar results to the experiments for different values of p and r. We can see that, for example for $p = 0.1$ the observed loss ratio decreases as r is increased.

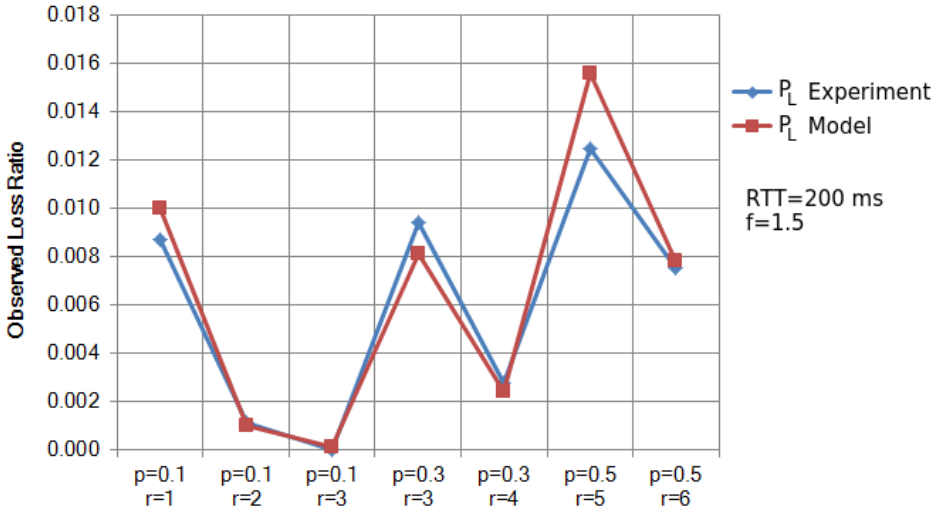


Figure 3.8: Experimental results vs. Model results for Observed Loss Ratio (P_L)

The comparison between the delay model and the experimental results is shown in Fig. 3.9. Here we note that we are examining a network with low losses, given that $p = 0.1$. According to our model, increasing r and RTO will result in an increase in the delay. What we can draw from this figure is that RTO has a more significant impact on the delay than r in networks with low losses. For instance, fixing $r = 2$ and changing T_0 from 0.5 to 2 increases the delay by a factor of roughly 2. However, fixing $T_0 = 2$ and changing r from 2 to 4 increases the delay by a negligible amount. This observation applies for a low-loss network; further aspects will be explored later in the analysis section.

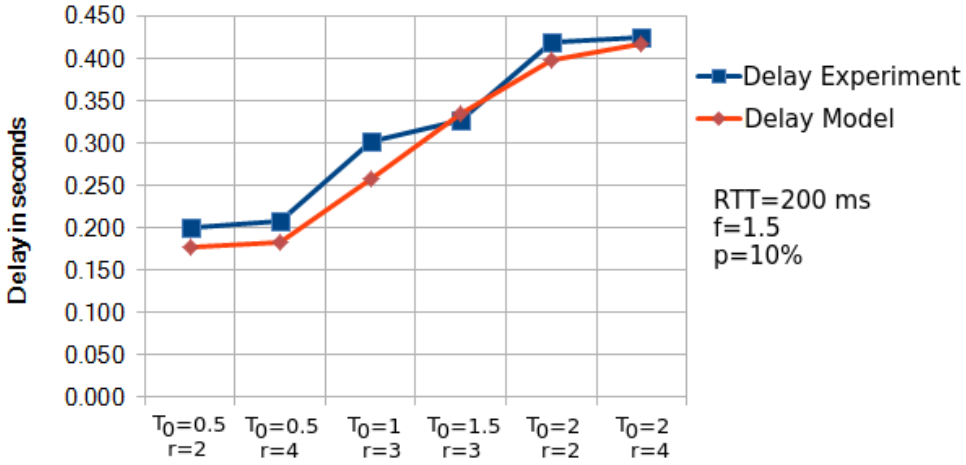


Figure 3.9: Experimental results vs. Model results for Delay

The experimental and model results of Goodput are presented in Fig. 3.10, where we used different combinations of r and RTO values. Again, in the case of a low-loss network ($p = 0.1$), reducing RTO is very beneficial for the Goodput and an increase in r does not yield significant changes for a fixed RTO.

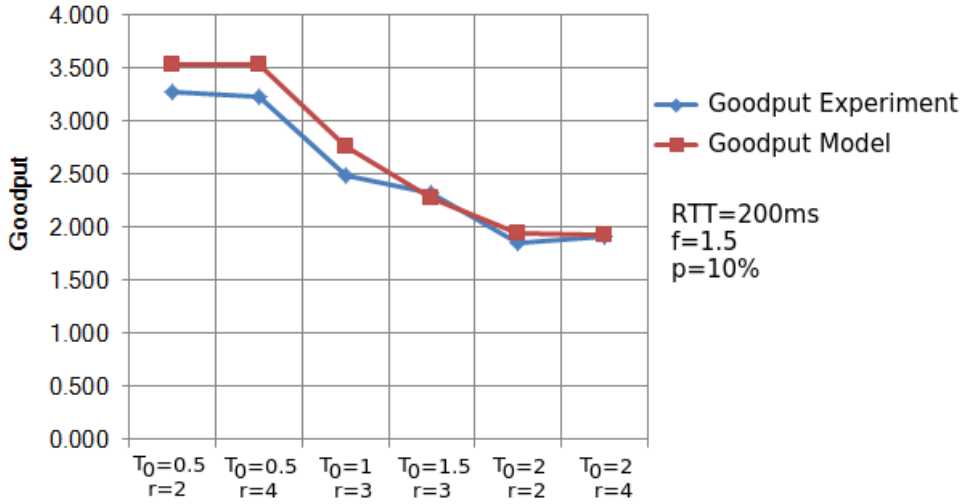


Figure 3.10: Experimental results vs. Model results for Goodput

To test the accuracy of the overhead for successful transmission model, we compared it to experimental results as per Fig. 3.11 again for different combinations of p and r. The graph shows that the ratios attained from the model and experiments are similar and almost exactly the same in many cases. To focus on extreme cases, we can examine low and high loss networks: for $p = 0.1$, changing r from 1 to 3 results in no additional overhead. For $p = 0.5$, changing r from 5 to 6 results in slightly more overhead in both the experimental results and the model, despite the little mismatch between the two. Therefore, for both low and high loss networks, we can see that p has the dominant effect on increasing

the overhead.

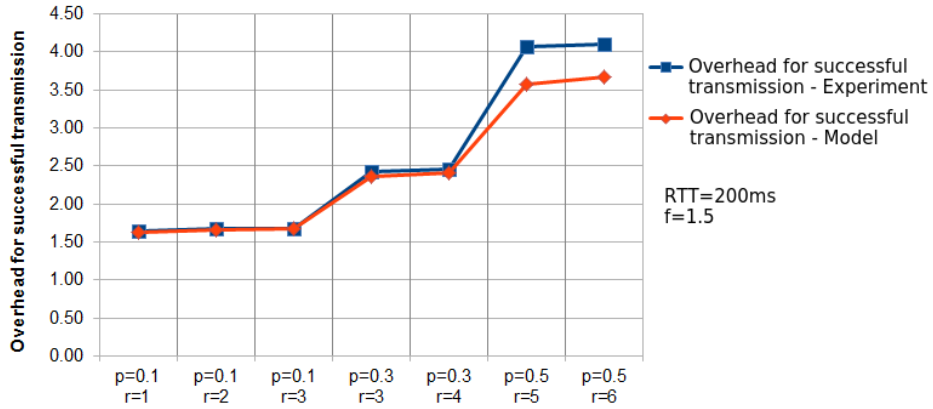


Figure 3.11: Experimental results vs. Model results for Overhead for successful transmission

3.4.2 Simple Gilbert Loss Model

The comparison of the observed losses ratio in Simple Gilbert loss model between the mathematical model and the experimental results is presented in Fig. 3.12. The observed losses obtained from the model and the experiment are similar for different values of r with a very small difference.

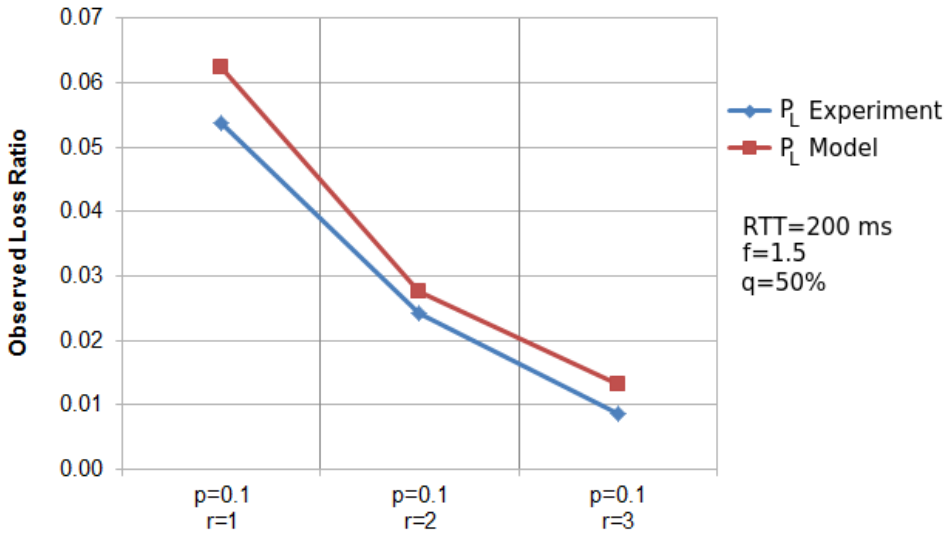


Figure 3.12: Experimental results vs. Model results for Observed Loss Ratio (P_L)

According to the presented results in Fig. 3.13, delay values of the model and the experiments are also similar for different values of r and RTO. In terms of the effects that the former values have on the delay, the Simple Gilbert model is similar to the Bernoulli model but not identical when the loss ratio p is low. In fact, looking at the experimental results, the retransmission counter r does not have as much effect on the delay as the RTO does, but it does nonetheless yield higher increments (especially

when RTO is high as well). For instance, when $r = 4$, changing T_0 from 0.5 to 2 increases the delay from 0.2s to roughly 0.8s, corresponding to a factor of 4. This phenomenon has been observed in the Bernoulli case. Now, if we keep RTO constant with $T_0 = 0.5$, changing r from 2 to 4 increases the delay from 0.2s to about 0.35s, where the difference is $\Delta_1 = 0.15$s. For a higher value of RTO where $T_0 = 2$, changing r from 2 to 4 increases the delay from 0.5s to about 0.78s, with $\Delta_2 = 0.28$s.

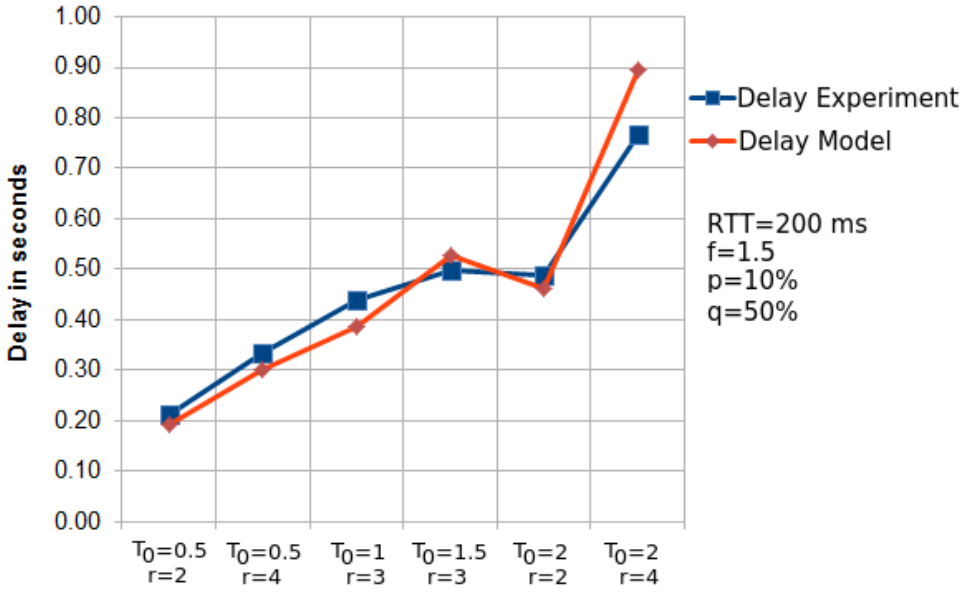


Figure 3.13: Experimental results vs. Model results for Delay

The experimental and model results of the goodput model are presented in Fig. 3.14. The graph shows that the goodput obtained from the model and experiments is similar for different values of r and RTO, where a reduction in RTO will result in a significant benefit for Goodput.

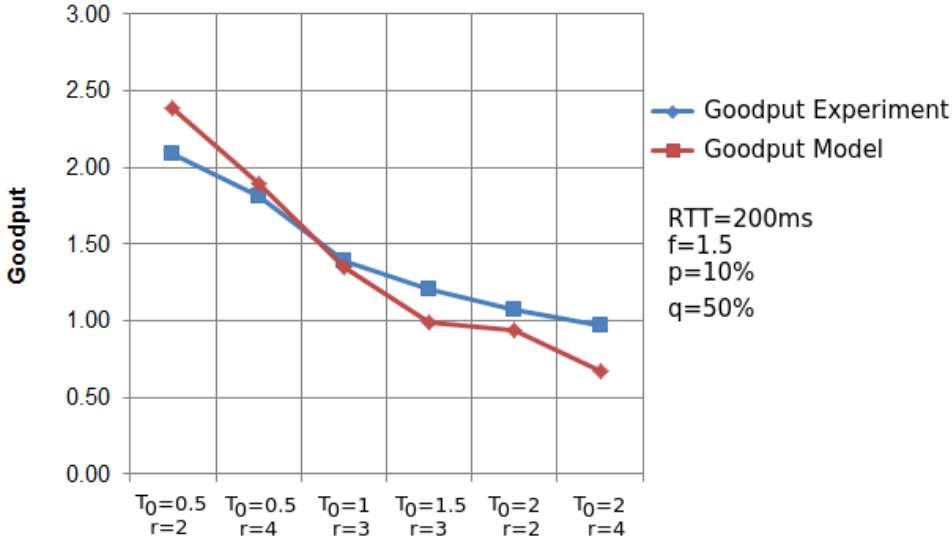


Figure 3.14: Experimental results vs. Model results for Goodput

The comparison between the model and experimental results of the Overhead for a successful transmission is shown in Fig. 3.15. At first glance, the results might not appear to be close. However, according to the numerical values, the relative error between our model and the experimental results is around 3%.

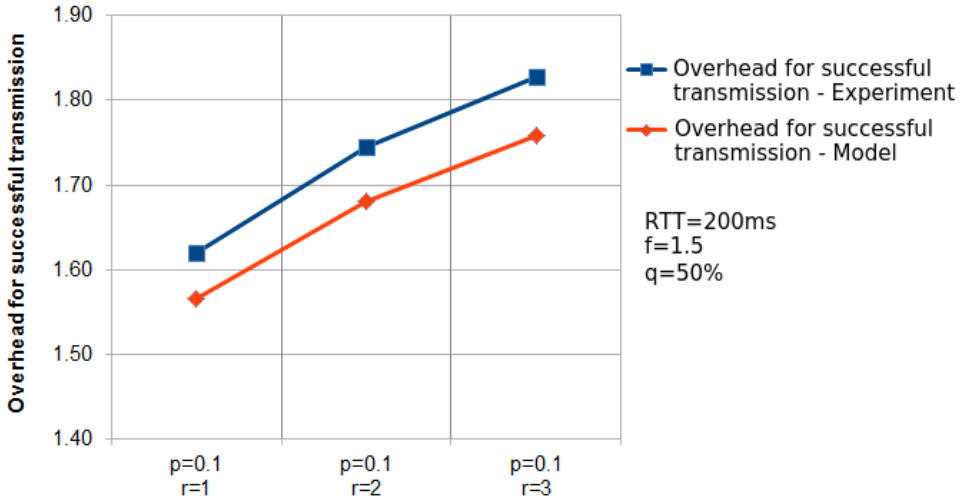


Figure 3.15: Experimental results vs. Model results for Overhead Ratio for successful transmission

3.5 Additional CoAP Performance Analysis using the analytical models

3.5.1 Bernoulli Loss Network Model

In Fig. 3.16, we show CoAP Observed Loss Ratio (P_L) as a function of the loss probability p and the retransmission counter r in a 3-D plot. The blue and green marks show the values of Observed losses while varying p from 10% to 50% and r from 1 to 9. We observe that observed losses are very high when the retransmission counter r is not high and the network is lossy. However, observed losses are reduced enormously when we increment retransmission counter gradually. This can be hired to set the retransmission counter adequately to avoid packet losses as required by the application.

Fig. 3.17 shows the delay of CoAP packets of successful transmission with different values of r and RTO while varying p with the following values: 10%, 20%, 30%, 40%. These figures can be used to determine the appropriate value of r to meet particular target delays. For example, when the loss ratio p is 10% and the target delay can be up to 0.7 sec, then r can be set to 3.

The delay of packet transmission in CoAP is affected by the parameters: RTO and r and the network loss ratio p. With low loss ratio p, the delay increases when RTO increases and r does not play a major role (Fig. 3.17 - a). However, when the losses increase in the network ($p > 10\%$), RTO and r both play a significant role in increasing the delay as seen in Figures 3.17 - b, 3.17 - c, and 3.17 - d. The main reason is that when a CON request is not acknowledged, retransmissions are attempted which exponentially

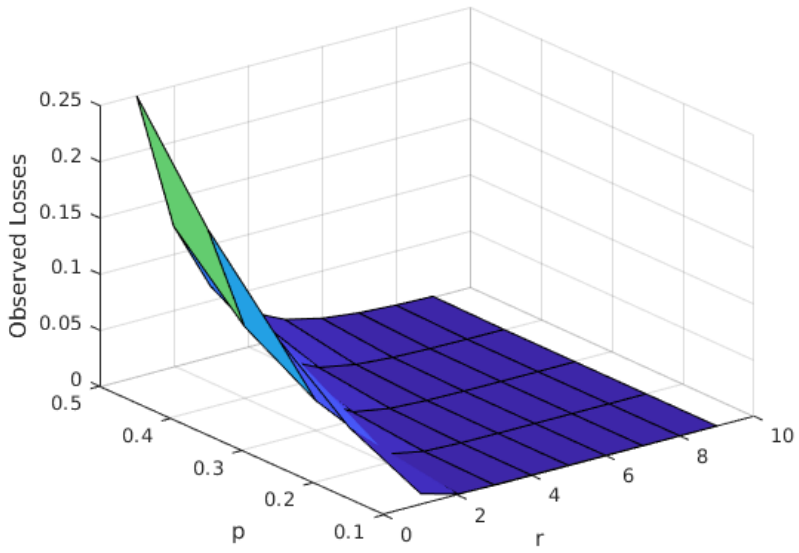


Figure 3.16: Observed Loss Ratio (P_L) as a function of p and r

double the timeout each time, and consequently increase the delay. Although retransmissions increase the delay of packet delivery in networks with high losses as previously discussed, increasing r reduces the observed losses as per Fig. 3.16. Thus, when reliability is more important than delay reduction, increasing r is justifiable.

Fig. 3.18 shows the goodput of CoAP transmission with different values of r and RTO while increasing the loss ratio p to: 10%, 20%, 30% and 40%. The obtained results can be used to determine the appropriate value of r to meet certain target goodput threshold. For example, if $p = 30\%$ and the target $goodput \geq 1$, then r can not be set greater than 2.

In general, selecting small values for the RTO parameter leads to a better goodput. In particular, when $p < 10\%$ (Fig. 3.18-a), r does not have much of an effect and only a reduction in RTO will yield better results since this corresponds to a low loss network. Predictably, goodput in low loss networks (Fig. 3.18-a) is better than the goodput in a higher loss network (Fig. 3.18-b) due to the value of p. As the network loss ratio increases $(p > 20\%)$, Goodput degrades when both r and RTO increase, as observed in Figures 3.18-c, 3.18-d. Hence, in networks with high loss ratio, r plays a significant role in goodput results along with RTO.

In Fig. 3.19, overhead for successful transmission is affected by high values of p. When there are few losses in the network, retransmission rate will not be high and the overhead for successful transmission will be reduced. On the other hand, overhead increases with the increased number of retransmissions in high lossy network.

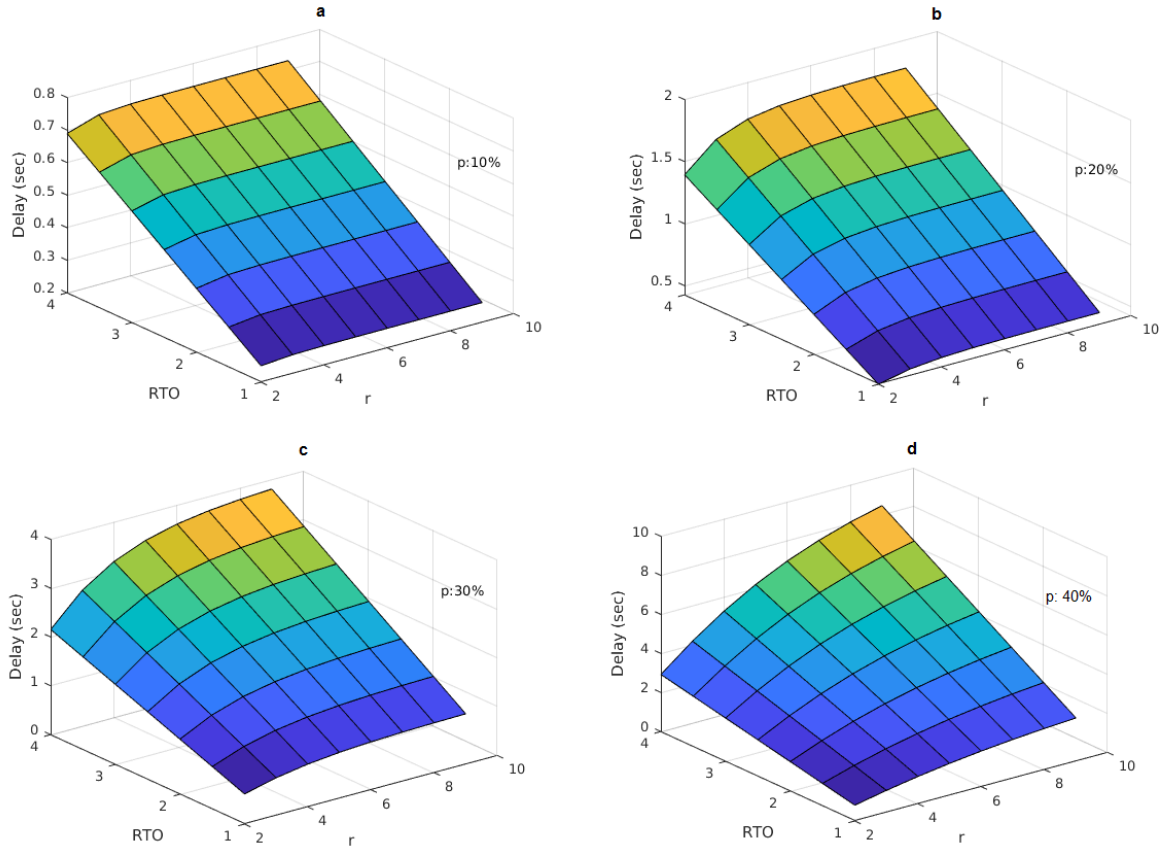


Figure 3.17: Delay as a function of RTO and r

3.5.2 Simple Gilbert Loss Model

Observed losses can be determined by our model according to the network loss ratio and number of retransmissions r. As before, we plot in Fig. 3.20a the CoAP Observed Losses (P_L) in 3-dimensions, while varying p from 10% to 50% and r from 1 to 9. As the figure indicates, the observed losses decrease to 0 when r is sufficiently high (e.g. r=6), despite the fact that the loss probability p is high (50%). On the other hand, in Fig. 3.20b, we plot the CoAP Observed Losses while varying q from 10% to 90% and r from 1 to 9, with p set to 20%. As can be observed from this figure, the CoAP observed losses increase as q decreases. This is expected since the complement $1 - q$ is increasing, and this term represents the network's burstiness. The importance of the results presented in 3.20 is that they help determine the value of the retransmission counter r in cases where we have high losses. As we will see later, the drawback of increasing r has a negative impact on the goodput.

The delay values are shown in Fig. 3.21, where we vary the parameters RTO from 1 to 4 sec, r from 2 to 10. We use different of value of p ($10\%, 20\%, 30\%, 40\%$). The value of q is set 50% which forces the occurrence of successive losses and hence packet delivery takes more time, which in turn increases the delay. As discussed before, although reducing the delay depends on RTO but r plays a significant role as well. This is especially true for high values of RTO where we can observe from Fig. 3.21 that,

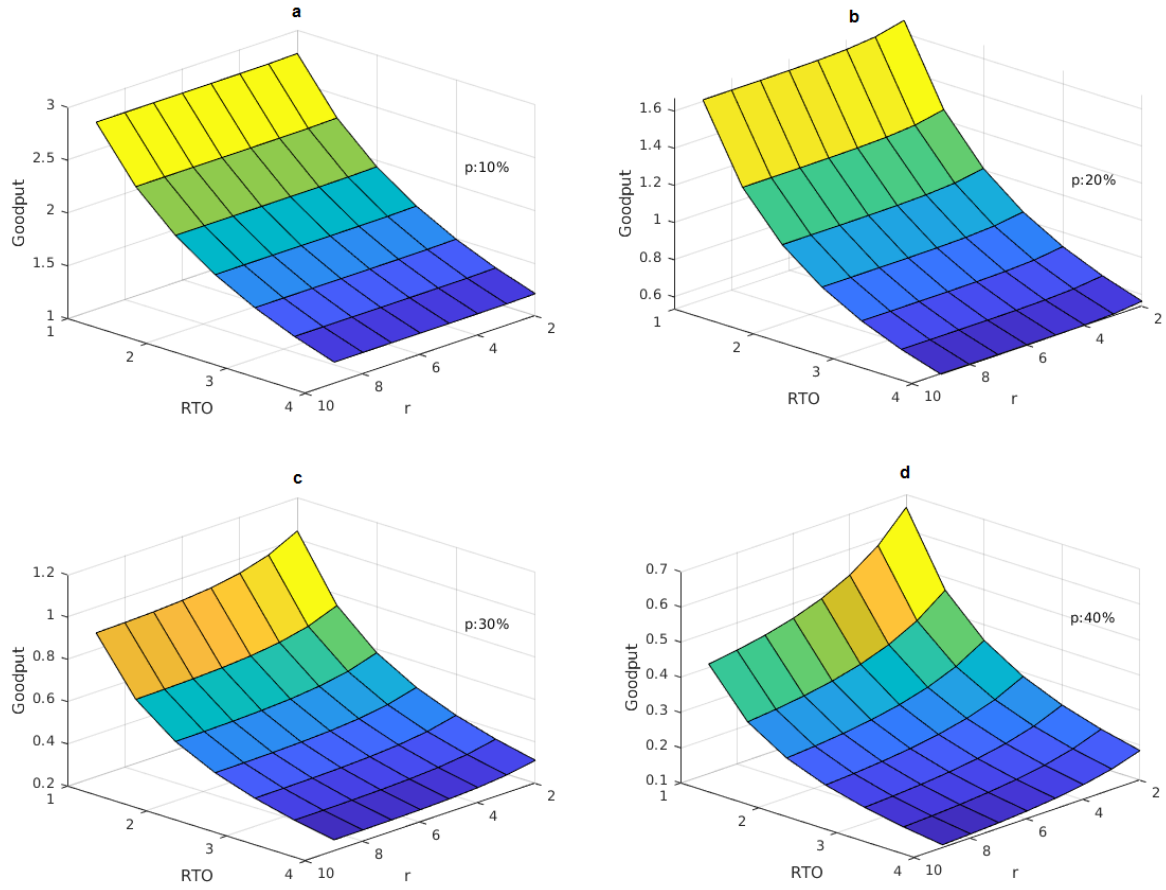


Figure 3.18: Goodput as a function of RTO and r

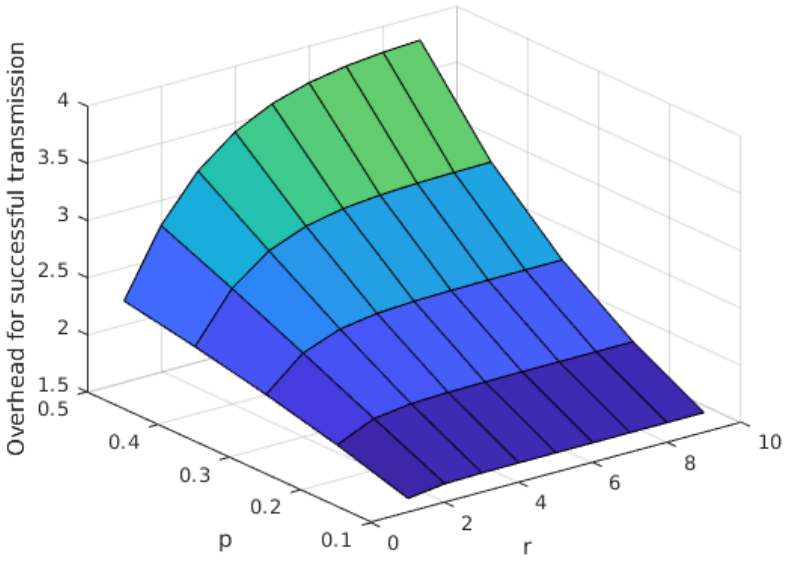
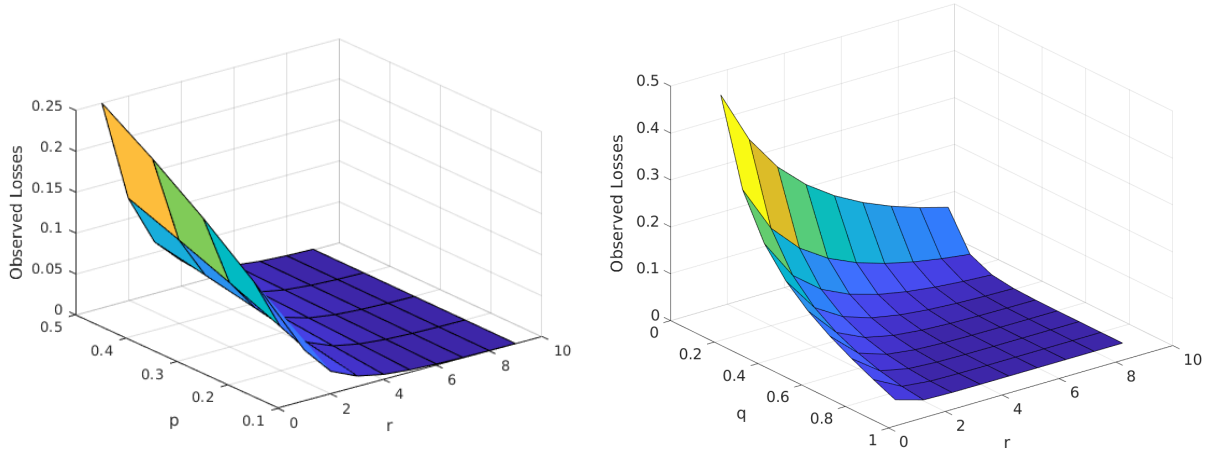


Figure 3.19: Overhead for successful transmission as a function of p and r



(a) Observed Loss Ratio (P_L) as a function of p and r with $q = 50\%$

(b) Observed Loss Ratio (P_L) as a function of q and r with $p = 20\%$

Figure 3.20: Observed Loss Ratio (P_L) as a function of p, q and r

for all network conditions, an increase in r negatively impacts the delay when RTO is also high.

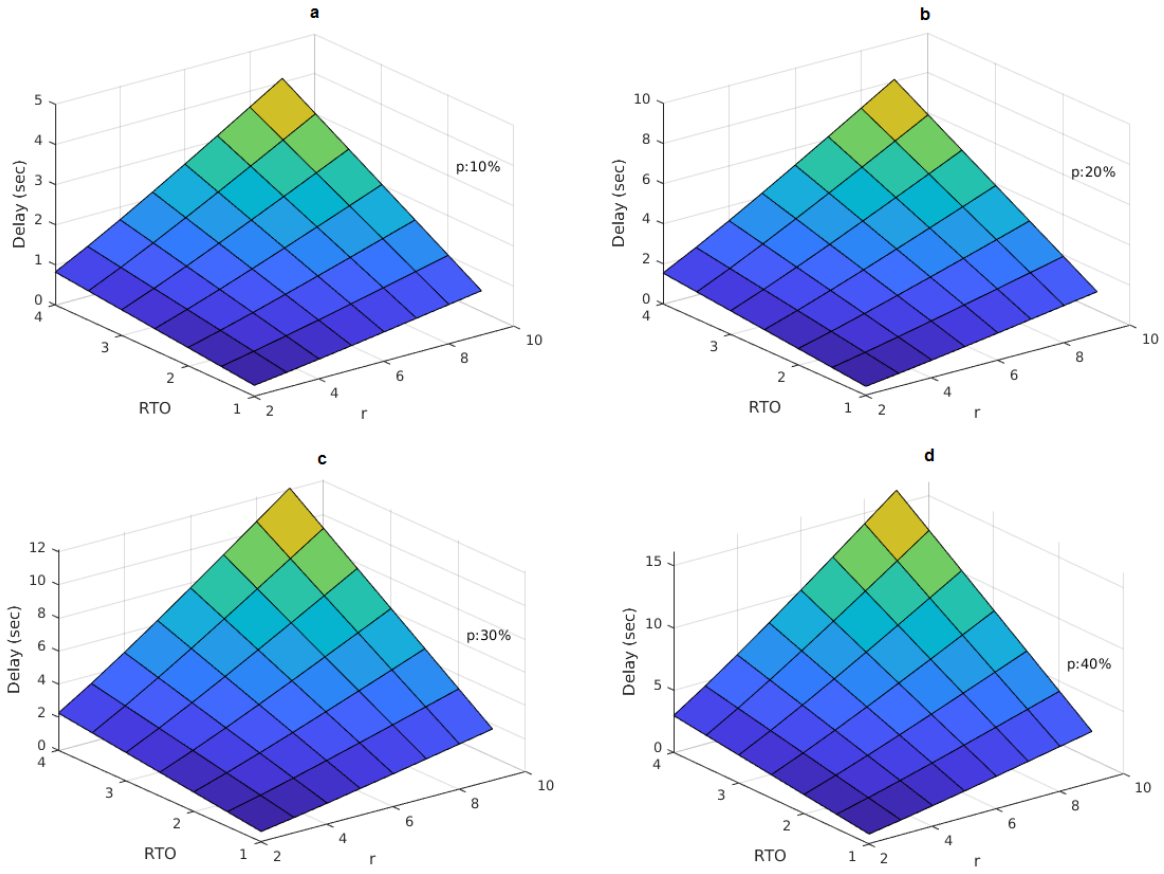


Figure 3.21: Delay as a function of RTO and r for $q = 50\%$

Goodput decreases when we increase r as seen in Fig. 3.22. At the first glance, the results look strange because when we increase r, losses are reduced and we are supposed to obtain a better goodput, which is not reflected in Fig. 3.22. However, the reason behind this is that increasing r will involve CoAP binary exponential back-off mechanism which will in turn increase the waiting time and decrease the goodput.

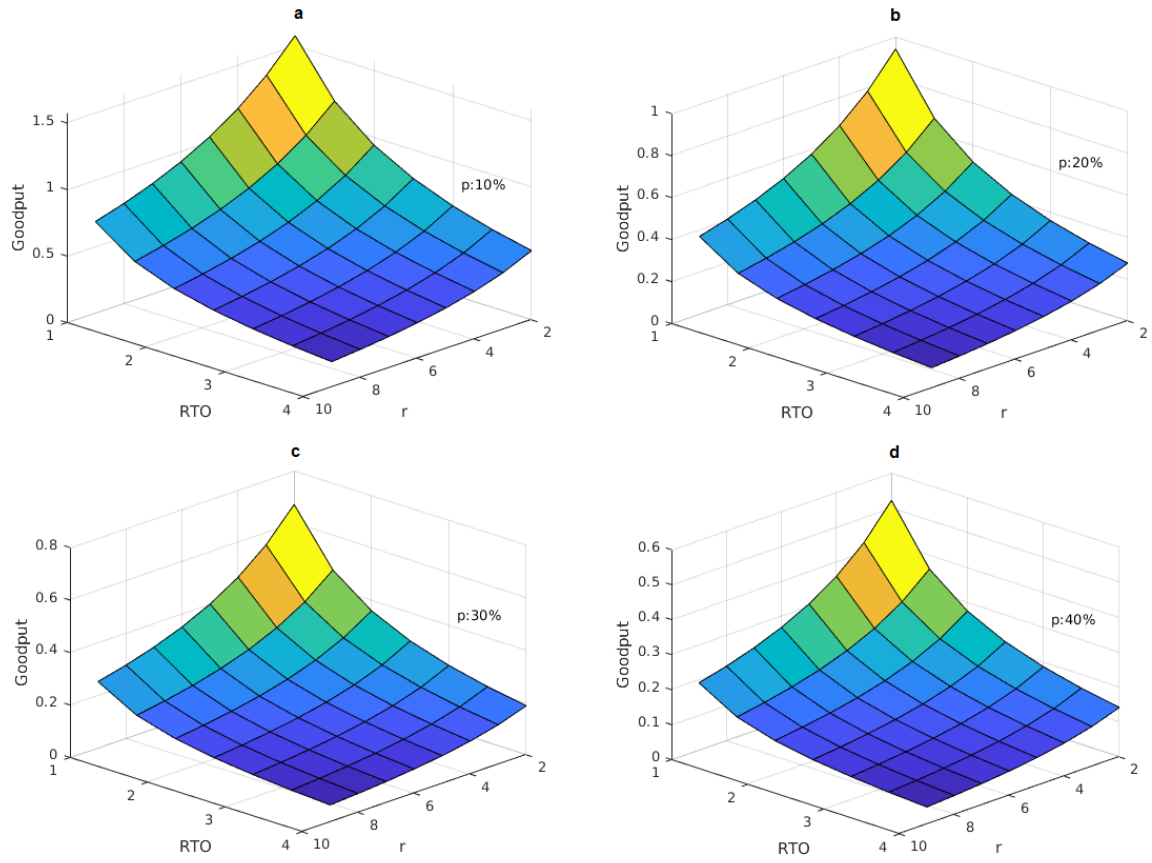
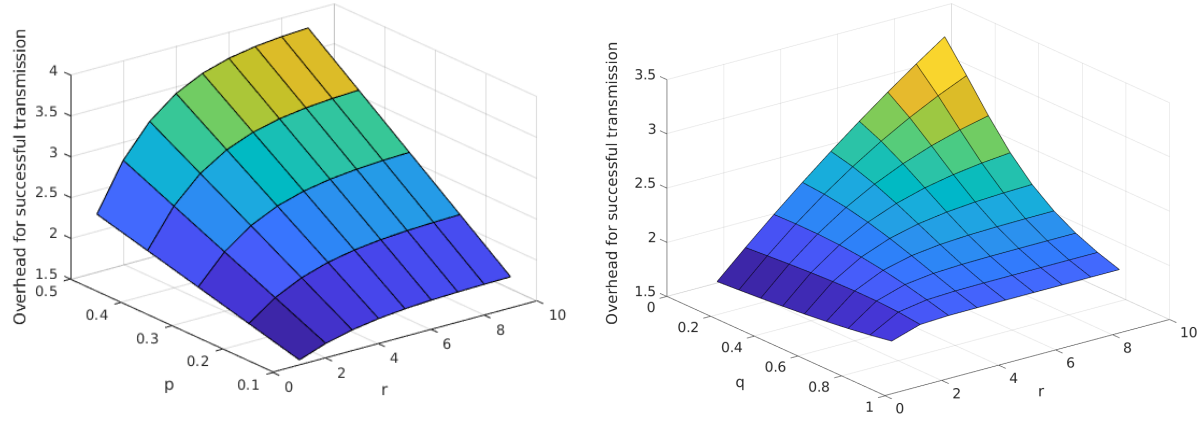


Figure 3.22: Goodput as a function of RTO and r for $q = 50\%$

As we have seen previously, in Fig. 3.23a we vary the values of p and r and we compute the Overhead for a successful transmission. As the figure indicates, the Overhead for a successful transmission increases slightly when only p is increased. However, when both variables p and r are increased, the overhead becomes much more. Similarly, for Fig. 3.23b but here we vary q and r. When the burstiness increase (i.e when q decreases), the overhead increases as r increases. Therefore, as we can see from Fig. 3.23, the overhead for a successful transmission in a lossy network increases when r is increased. The is because re-sending the same payload is imposed for each retransmission attempt.



(a) Overhead for successful transmission as a function of p and r with $q = 50\%$

(b) Overhead for successful transmission as a function of q and r with $p = 20\%$

Figure 3.23: Overhead for successful transmission as a function of p, q and r

3.6 Conclusion

In this chapter, we presented mathematical models of CoAP performance metrics. We showed how to find these models first using Bernoulli loss model then using the Simple Gilbert loss model. Direct computation and closed forms for all our mathematical models were presented. Secondly, we validated our models using experiments. Then, a profound analysis on CoAP performance was carried out by utilizing the presented models. The comparison with experimental results using a real implementation of CoAP shows the accuracy and the usefulness of our proposed models. On the one hand, the value in our models is that different network and protocol parameters can be computed, thus providing guidance on the impact of the CoAP parameters on its behavior and performance. For instance, we found that increasing retransmission attempts r does not increase the goodput in spite of reducing the loss ratio. The reason for this is that the RTO value must change as well. On the other hand, the importance of this study is that it can be used to determine good (or optimal) values of r and RTO to meet target requirements such as CoAP losses, latency and goodput which will help in reducing congestion.

Another concrete utility of our model is the following: Assume that $T = T_1$ and r are fixed, then CoAP will achieve a certain goodput denoted as $G_1(T_1, r)$. If we need to reduce losses, we can increase r by 1 but goodput will consequently decrease and be denoted as $G_2(T_1, r + 1)$. Can we determine a new value for the timeout T_2 in terms of T_1 such that despite the increase in r, the goodput level is maintained? This can be done easily and dynamically via our model and translates to solving the following equation:

$$G_1(T_1, r) = G_2(T_2, r + 1). \tag{3.110}$$

substituting (3.30) in (3.110), then

$$\frac{1-p^{r+1}}{(1-p^{r+1})R+T_0p\left(\frac{1-(2p)^{r+1}}{1-2p}\right)}=\frac{1-p^{r+2}}{(1-p^{r+2})R+T_1p\left(\frac{1-(2p)^{r+2}}{1-2p}\right)} \tag{3.111}$$

$$\implies T_2=T_1\times\underbrace{\frac{(1-p^{r+2})(1-(2p)^{r+1})}{(1-p^{r+1})(1-(2p)^{r+2})}}_{\text{Required factor}} \tag{3.112}$$

The behavior of the *Required factor* versus p and r is presented in Fig. 3.24. For instance, the higher the network losses, the lower the factor and that means, the new RTO must be reduced further to compensate the increase in r. As can be noticed from the figure that the relation between the $fraction$ and p and r is not linear which necessitate the need for the model to find the new value of T.

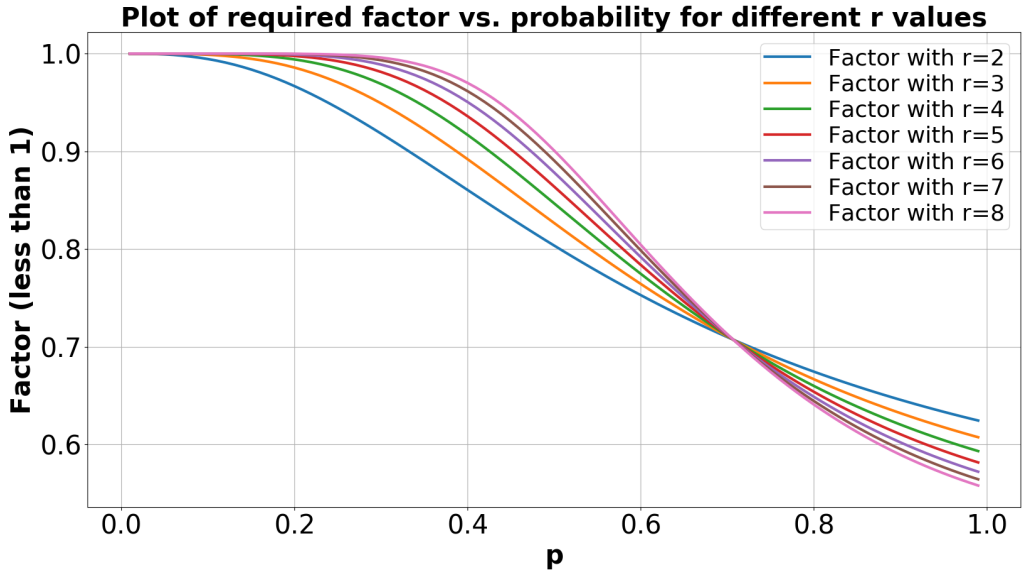


Figure 3.24: Required factor as a function of p for different r values

To conclude this chapter, CoAP was studied via modelling in two different network scenarios and this allowed us to identify the CoAP parameters that can lead to a better performance. For instance, one of CoAP's parameters that should be optimized is the initial retransmission timeout (RTO_{init}), where we realized the need for reducing RTO_{init} as much as possible to improve packet delivery and enhance reliability. This motivates Chapter 5 which is related to RTO optimization.

4

CoAP Modelling: Gilbert-Elliot Full Model

4.1 Introduction

In this chapter, we present another modelling framework, using Markov chains, allowing us to compute the performance metrics for the full model of Gilbert-Elliott [26, 22] (Figure 4.1). The gradual shift from a simple model to a more complex (and realistic) one is a good strategy for two reasons. Firstly, it gives space for validation and offers a correctness-check because the same formulas are obtained using different frameworks. Secondly, the modelling aspect itself allows for a better and deeper understanding of the protocol's functionalities in practice, such as the timeout behavior when an ACK is not received. Thus, the previous performed computations were based on the successive modelling of the protocol's behavior and the formulas were obtained from these studies, whereas in this chapter we will use the Markov chain .

The method based on *Markov Chains* can reduce the complexity of the modelling. Unfortunately, direct computation is not possible as before and the use of a 1-dimensional *Markov Chain* to model CoAP transitions is insufficient because it does not capture the hidden *Markov Chain* states of the full Gilbert-Elliott model. This fact sheds light on the importance of understanding the protocol first, via Simple Gilbert modelling, and next studying the Gilbert-Elliott model. The aforementioned model represents the general case and thus provides expressions for CoAP performance in all scenarios.

The Gilbert-Elliott model is commonly used to model lossy environments and especially wireless

networks. There are two states, Good (G) and Bad (B). $1-k$ is the loss probability in the G state. $1-h$ is the loss probability in the B state. p and q are the transition probabilities between the two states. $1-q$ is the probability for a successive loss in the B state while $1-p$ is the probability for a successive success in the G state.

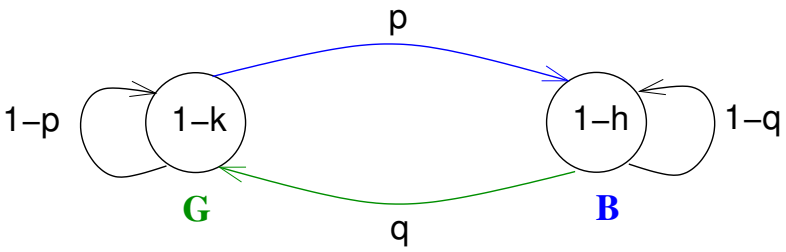


Figure 4.1: The Gilbert-Elliott Markov Chain model

The novelty in our work is the evaluation of CoAP protocol under Gilbert-Elliot model with a precise computation without approximations despite the hidden aspect of the Gilbert-Elliot Markov chain.

4.2 An Exact Model for CoAP Performance under Gilbert-Elliott

We consider as previously a CoAP sender that is transmitting packets whenever allowed by the protocol. We study the Confirmed mode of CoAP since it is the one which includes a loss recovery mechanism. The principle is based on an exponential backoff procedure in case of loss and it is reminded again in the following: When the CoAP sender transmits a packet with a confirmation request (CON packet), it waits for an acknowledgment (ACK packet). If the ACK is received after a round-trip-time, then it moves to the next packet to send. Otherwise, a first timeout expires. In this case, the sender retransmits the same packet and doubles the timeout. Successive retransmissions are limited to r times after which the packet is dropped and the sender moves to the next packet to send if any. The timeout value is randomized via a multiplication factor to avoid synchronization problems (Figure 4.2).

Denote by R the average round trip time, T the average of the first time value including the randomization factor.

4.2.1 Modeling CoAP Transmissions

It is important to notice first that modeling CoAP transmissions using one classic dimensional Markov chain will not provide exact analytical results. Such Markov chain is shown in Figure 4.3 representing the time between the sending instant and the reception of the acknowledgment or the expiration of the timeout. State R corresponds to a successful transmission. State 2^iT corresponds to a packet loss followed by a timeout equals to $2^iT, 0 \leq i \leq r$. The terms 2^iT are shown in Figure 4.2 which summarizes also the retransmission behavior of CoAP. State 2^rT is the last unsuccessful attempt (retransmission)

for the same packet, and thus the packet is dropped since no more retransmissions are allowed. The application moves to the next packet to send. If this new packet is successful then the chain moves back to state R, otherwise, it turns back to state 2^0T that corresponds to a loss and the expiration of the timeout.

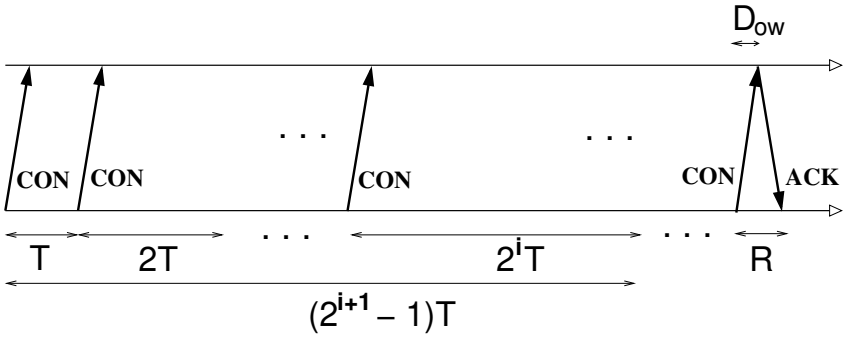


Figure 4.2: Summary of CoAP exponential backoff procedure for congestion control as described in [53]. T is the first timeout average value. R is the average Round Trip Time. D_{ow} is the average one-way delay.

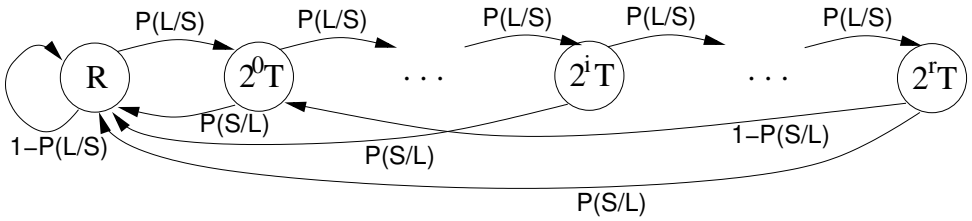


Figure 4.3: A possible Markov chain modeling CoAP successive transmissions

In order to determine all transition probabilities, we need only to find two probabilities: The probability of a lost transmission such that the previous one was a success denoted by $P(L/S)$, and the probability of a success given that the previous one is lost denoted by $P(S/L)$. These probabilities are difficult to obtain since the Markov chain of Gilbert-Elliott model is *hidden* [16]. This is because when a packet is lost, we can not decide if the Markov state is Bad (B) or Good (G) since the loss event is possible in both states. Consequently, we need to compute the probabilities of being in the B state or in the G state knowing that the packet is successful or lost. Of course, these probabilities can be computed using Bayes theorem as follows:

$$P(L/S) = P(L/G)P(G/S) + P(L/B)P(B/S) \tag{4.1}$$

$$P(S/L) = P(S/G)P(G/L) + P(S/B)P(B/L) \tag{4.2}$$

Using Bayes theorem yields to

$$P(L/S) = P(L/G)\frac{\mathbf{P(S/G)P(G)}}{\mathbf{P(S)}} + P(L/B)\frac{\mathbf{P(S/B)P(B)}}{\mathbf{P(S)}}$$

$$= \frac{((1-p)(1-k) + p(1-h))\,kq + ((1-q)(1-h) + q(1-k))\,hp}{kq + hp} \tag{4.3}$$

$$P(S/L) = P(S/G)\frac{\mathbf{P(L/G)P(G)}}{\mathbf{P(L)}} + P(S/B)\frac{\mathbf{P(L/B)P(B)}}{\mathbf{P(L)}}$$

$$= \frac{((1-p)k + ph)\,(1-k)q + ((1-q)h + qk)\,(1-h)p}{(1-k)q + (1-h)p} \tag{4.4}$$

Then, we can solve the Markov chain to obtain the following steady state probabilities of CoAP transmissions:

$$\begin{aligned}
&\pi(R) = \tfrac{s}{(l+s)} \\
&\pi(2^iT) = \tfrac{ls(1-s)^i}{(l+s)(1-(1-s)^{r+1})}, 0 \leq i \leq r \\
&\quad \text{with } l = P(L/S), \text{ and } s = P(S/L)
\end{aligned}$$

However, the drawback of this approach is that $P(G)$, $P(S)$ and $P(L)$ are *unknown* and can only be *approximated* by using steady state probabilities of the Gilbert-Elliott which is not applicable when analyzing instantaneous events that are correlated. Later, we will show that this approximation does hold in some particular cases.

To solve exactly the hidden state problem, we need to split the states of the previous Markov chain into two. One when transmission events occur in the G state and the other when transmission events occur in the B state as shown in Figure 4.4. Hence, transition probabilities can be computed exactly. Any transition probability in the Markov chain is the product of two probabilities. The first is the probability of moving from G to B and vice versa. The second is the probability of loss or success in the G state or in the B state. For example, the transition probability from state 2^iT_G to state $2^{i+1}T_B$ is $p(1-h)$.

It is more practical to write the steady state equations related to this Markov chain using matrices and vectors to better reflect the "two dimensions" of the Markov Chain and thus ensure its resolution. First, we define the following matrices.

$$A = \begin{bmatrix} (1-p)(1-k) & q(1-k) \\ p(1-h) & (1-q)(1-h) \end{bmatrix}, V_R = \begin{bmatrix} \pi(R_G) \\ \pi(R_B) \end{bmatrix}$$

$$B = \begin{bmatrix} (1-p)k & qk \\ ph & (1-q)h \end{bmatrix}, V_i = \begin{bmatrix} \pi(2^iT_G) \\ \pi(2^iT_B) \end{bmatrix}, 0 \leq i \leq r$$

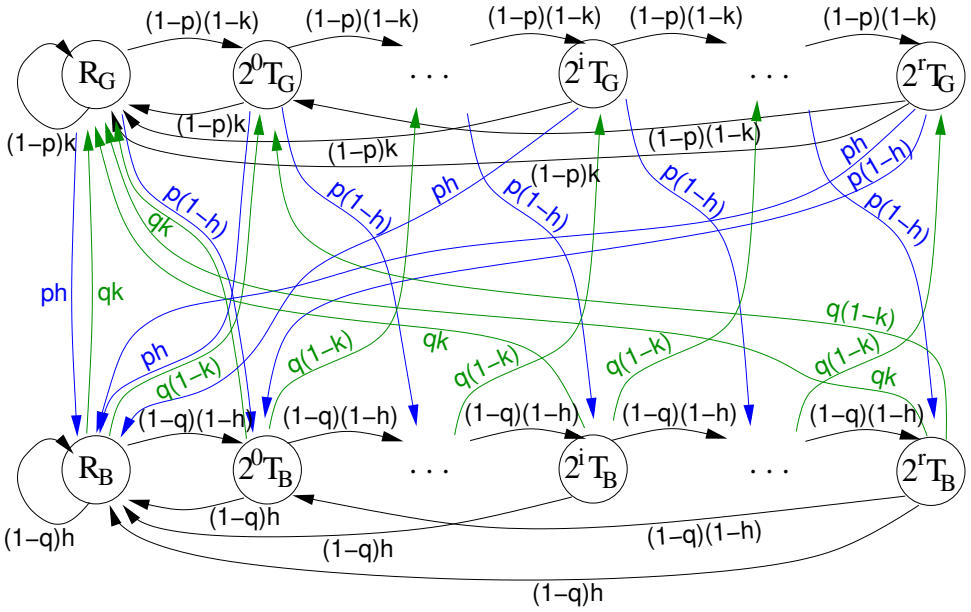


Figure 4.4: The exact Markov Chain modeling CoAP successive transmissions under the Gilbert-Elliott loss model

Matrices A and B are not to be confused with a transition probability matrix. Notice that A includes all transition probabilities to a loss state, and B includes all transition probabilities to a success state. Now, steady state equations can be written using linear algebra as follows

$$V_R = BV_R + B\sum_{i=0}^{r} V_i \tag{4.5}$$

$$V_0 = AV_R + AV_r\,, \quad V_i = AV_{i-1}, 1 \leq i \leq r \tag{4.6}$$

Now, we can solve the system and compute the transition probabilities thanks to matrix operations. First, we notice that $V_R + \sum_{i=0}^{r} V_i = [\pi(G), \pi(B)]^t$. Summing all the steady state probabilities which means all the terms of equations (4.5) and (4.6) we obtain

$$\begin{bmatrix} \pi(G) \\ \pi(B) \end{bmatrix} = (A+B) \begin{bmatrix} \pi(G) \\ \pi(B) \end{bmatrix} \tag{4.7}$$

which provides $\pi(G)$ and $\pi(B)$. We also have

$$V_R = B \begin{bmatrix} \pi(G) \\ \pi(B) \end{bmatrix} \tag{4.8}$$

Then, from equations (4.6) we deduce

$$V_i = A^i V_0, 1 \leq i \leq r \tag{4.9}$$

Combining equations (4.9) with (4.6) and (4.8) yields to

$$V_i = (I_2 - A^{r+1})^{-1} A^{i+1} V_R, 0 \leq i \leq r \tag{4.10}$$

I_2 is the 2×2 identity matrix. We also use the fact that $A^i(I_2 - A^{r+1})^{-1} = (I_2 - A^{r+1})^{-1}A^i$. The expressions of the elements of A^{i+1} can be determined in closed-form via the Cayley-Hamilton theorem [59] (Appendix A) or through diagonalization and similarity transformation (Appendix B). The Cayley-Hamilton theorem allows to say that a square matrix A satisfies its own characteristic equation $\det(xI_n - A) = 0$ where $det()$ is the determinant function and I_n is the $n \times n$ identity matrix. For a 2×2 square matrix, the equation is $x^2 - (\lambda_1 + \lambda_2)x + \lambda_1\lambda_2 = 0$ where λ_1 and λ_2 are the two eigenvalues of matrix A. Thus, we can write the following interesting matrix equation

$$A^2 - (\lambda_1 + \lambda_2)A + \lambda_1\lambda_2 I_2 = O \tag{4.11}$$

Now, define the following two matrices C_1 and C_2 as follows

$$C_1 = A - \lambda_1 I_2,\ C_2 = -A + \lambda_2 I_2 \tag{4.12}$$

then thanks to Cayley-Hamilton theorem which means equation (4.11), we can derive several useful properties:

$$C_1 C_2 = O,\ C_1^2 = (\lambda_2 - \lambda_1)C_1,\ C_2^2 = (\lambda_2 - \lambda_1)C_2 \tag{4.13}$$

$$C_1^i = (\lambda_2 - \lambda_1)^{i-1} C_1,\ C_2^i = (\lambda_2 - \lambda_1)^{i-1} C_2 \tag{4.14}$$

Next, we write the matrix A using the two matrices C_1 and C_2 as follows

$$A = \frac{\lambda_1}{\lambda_2 - \lambda_1} C_2 + \frac{\lambda_2}{\lambda_2 - \lambda_1} C_1 \tag{4.15}$$

Using properties (4.13) and (4.14), we deduce A^i

$$A^i = \frac{\lambda_1^i}{\lambda_2 - \lambda_1} C_2 + \frac{\lambda_2^i}{\lambda_2 - \lambda_1} C_1 \tag{4.16}$$

Finally, replacing C_1 and C_2 in the previous equation by their expressions as defined in equation (4.12),

we obtain the final result for A^i:

$$A^i = \frac{\lambda_2^i - \lambda_1^i}{\lambda_2 - \lambda_1} A + \frac{\lambda_2 \lambda_1^i - \lambda_1 \lambda_2^i}{\lambda_2 - \lambda_1} I_2 \tag{4.17}$$

For our matrix $A = \begin{bmatrix} (1-p)(1-k) & q(1-k) \\ p(1-h) & (1-q)(1-h) \end{bmatrix}$, the eigenvalues λ_1 and λ_2 are computed as

$$\lambda_1 = tr(A)/2 - \sqrt{tr(A)^2/4 - \det(A)} \tag{4.18}$$

$$\lambda_2 = tr(A)/2 + \sqrt{tr(A)^2/4 - \det(A)} \tag{4.19}$$

$$tr(A) = (1-p)(1-k) + (1-q)(1-h) \tag{4.20}$$

$$\det(A) = (1-k)(1-h)\,(\,(1-p)(1-q) - pq\,) \tag{4.21}$$

To compute all the steady state probabilities V_is, we first compute V_0 using equations (4.10) and (4.8). Then, we use equation (4.6) to compute all the others in an algorithm with a total time complexity equals to $O(r)$ only. It is also possible to compute directly any V_i using (4.9). Finally, we compute

$$\begin{array}{rcrcll} \pi(R_G) & + & \pi(R_B) & = & \pi(R), & \\ \pi(2^i T_G) & + & \pi(2^i T_B) & = & \pi(2^i T), & 0 \leq i \leq r \end{array} \tag{4.22}$$

Computing Performance Metrics of CoAP: Using the steady state probabilities of the CoAP Markov Chain, i.e. $\pi(R)$ and $\pi(2^i T)$, $0 \leq i \leq r$, now we can compute several performance metrics such as the experienced loss ratio P_L, the average goodput GP, the average delay $\bar{D}_s$, and the overhead for a successful transmission O_s. Other metrics can also be computed such as the average number of retransmissions required to send a CoAP packet successfully, the total overhead, the average number of losses in the Bad state.

4.2.2 Loss Ratio

The loss ratio is defined as the average number of losses divided by the total number of CoAP packets sent from the application. Thus,

$$P_L = \frac{\pi(2^r T)}{\pi(2^r T) + \pi(R)} \tag{4.23}$$

Using the results from equations (4.17) to (4.21), we can compute the elements of A^{r+1}. Denote them by $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$, we provide the expressions of the matrix elements in Appendices A and B. Then we can determine $\pi(2^r T)$ from equations (4.10) and (4.22):

$$\pi(2^r T) = \frac{(b+d)ph + (a+c)qk - (\lambda_1\lambda_2)^{r+1}(ph+qk)}{(p+q)\left(1 - \lambda_1^{r+1} + \lambda_2^{r+1} + (\lambda_1\lambda_2)^{r+1}\right)} \tag{4.24}$$

where λ_1 and λ_2 are the eigenvalues of matrix A as defined in (4.18) and (4.19). Thus, from equation (4.23), we obtain the closed form expression of the experienced loss ratio P_L:

$$P_L = \frac{(b+d)ph + (a+c)qk - (\lambda_1\lambda_2)^{r+1}(ph+qk)}{(b+1-a)ph + (c+1-d)qk} \tag{4.25}$$

It is important to notice that the loss ratio does not correspond to the probability of having $r+1$ successive losses in the Gilbert-Elliott Markov model which is usually computed through classic dynamic programming algorithms. Indeed, the loss ratio is computed only when the CoAP sender generates a new packet, and this event depends on previous sent packets and their status, i.e. loss, success, first retransmission, last retransmission. Henceforth, our Markov chain combines the Markov chain of the network model and the application model in order to capture exactly the interaction between the application and the network.

4.2.3 Goodput

The average goodput is computed as the average number of successful packets in a given period divided by the average delay of that period. Here, we select the period to be the inter-arrival delay between transmissions (new packets from the application and retransmissions). Thus,

$$GP = \frac{\pi(R)}{\pi(R) \times R + \sum_{i=0}^{r} \pi(2^i T) \times 2^i T} \tag{4.26}$$

$\pi(R)$ and $\pi(2^i T)$s are obtained as described above for the loss probability.

4.2.4 Delay

For the delay several calculations are possible. For instance:

$$\bar{D}_s = D_{ow} + T\frac{\sum_{i=0}^{r-1} 2^i \pi(2^i T) - \pi(2^r T)\,(2^r - 1)}{\pi(R)} \tag{4.27}$$

D_{ow} is the average one-way delay between the CoAP sender and receiver which is independent from the CoAP congestion control procedure. The next term in the equation is the total delay of attempts before the last one minus the waiting delay in case of loss, divided by the probability of success so that we obtain the average waiting delay before a success. It is worthy to notice that this delay is valid even if the CoAP sender does not send continuously its packets.

4.2.5 Overhead for a Successful Transmission

To compute the overhead, first we compute the average number of retransmissions required to send a CoAP packet successfully. It is obtained by computing the average of the probability of having retransmissions such that an ACK is received at the end of these retransmissions, without counting application losses, i.e. observed losses by the application ($-\pi(2^rT)$ in equation (4.28)). Then, we compute the total sent data using the size of a packet Z and of a payload Y, divided by the payload Y.

$$O_s = \left(1 + \sum_{i=0}^{r-1} \frac{\pi(2^iT) - \pi(2^rT)}{\pi(R)}\right) \frac{Z}{Y} - 1 \tag{4.28}$$

4.3 Using the CoAP Analytical Model to Tune CoAP

We provide here one example how to use the model to tune CoAP parameters. Figure 4.5 shows, for different values of the re-transmission limit r, the experienced loss ratio by the CoAP sender while increasing the duration of the Bad period which is controlled by $1 - q$. Here, we used the exact model. This result can be used for instance to determine the adequate value of r to deploy in order to meet some target loss ratio requirement. For instance, if $1 - q \leq 0.8$ in the network, and the target loss ratio is 0.2, then a re-transmission limit of 2 is quite sufficient. This will reduce the delay and the overhead due to re-transmissions.

4.4 Comparison Between the Exact and Approximated Model

Table 4.1 compares the exact model with the approximated one. In general, the approximated model provides the exact results if $p = 1 - q$, or $k = 1, h = 0$, or $k = h$ because in these cases the Gilbert-Elliott Markov Model is not hidden any more. In other words, it is possible to infer exactly the current state of the Markov chain (G or B), or not knowing the current state does not impact the computation due to symmetry of the Markov chain for instance. If one of these conditions is approximately satisfied, for instance $k \approx h$, then the approximated model provides close results. The farther from the cases mentioned above, the wronger the approximated model. We observe however that the simple classic

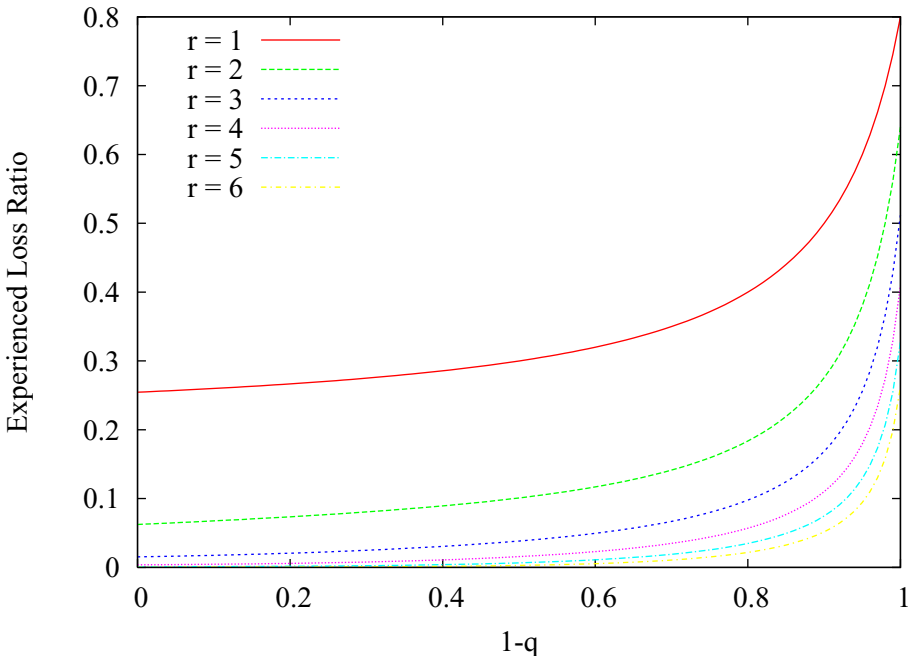


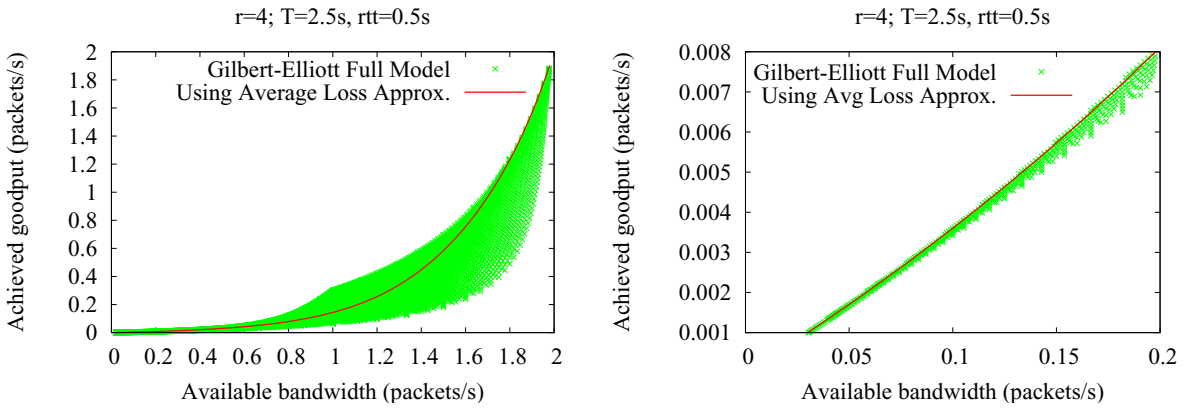
Figure 4.5: Experienced Loss ratio when $1-q$ increases. Parameters: p=0.1, k=0.8, h=0.2.

formula $(1-\frac{kq+hp}{p+q})^{r+1}$ is far from reality in most cases including the Simple Gilbert case, i.e. $k=1, h=0$ except if p is equal exactly to $1-q$.

Table 4.1: Comparison between the approximated and the exact models

r = 3				Exact	Approximated	Using
p	q	k	h	Model	Model	$1-\frac{kq+hp}{p+q}$
0.1	0.5	1	0	0.013158	0.013158	0.000772
0.1	0.5	0.95	0.05	0.013129	0.011279	0.001599
0.1	0.5	0.75	0.25	0.018239	0.016534	0.012347
0.1	0.5	0.4	0.4	0.129599	0.129599	0.129600
0.1	0.5	0.9	0.4	0.003700	0.002897	0.001129
0.1	0.1	0.75	0.25	0.096715	0.090301	0.062500
0.1	0.05	0.75	0.25	0.151292	0.146092	0.115788
0.1	0.8	0.75	0.25	0.009399	0.009321	0.008717
0.4	0.6	0.75	0.25	0.041006	0.041006	0.041006
0.75	0.25	0.75	0.25	0.152588	0.152588	0.152588

Fig. 4.6 shows the achieved goodput of the exact and the approximated model while varying the residual bandwidth. As can be observed from Fig. 4.6a where we vary the bandwidth from 0 to 2 packets per second, for small residual bandwidth values, the results of the approximated model looks the same as that of the exact model. We argue it is not the case. As per Fig. 4.6b where we zoom between 0 to 0.2 packets per second, the results are not the same.



(a) Goodput: Varying the bandwidth from 0 to 2 packets/s

(b) Goodput: Zooming between 0 and 0.2 packets/s

Figure 4.6: Approximated results vs. Exact Model results while varying the available bandwidth

4.5 Revisiting the Simple Gilbert Loss Model

We have considered in this chapter the 2-state Markov approach as introduced by Gilbert [26] and Elliot [22]. The Gilbert-Elliot model has two states (Good and Bad) and four independent parameters (p, q, h, k). Now, assuming that the good state G is packet loss-free $(k = 1)$ and the bad state B is packet gain-free $(h = 0)$, the model becomes equivalent to the Simple Gilbert model presented in the previous chapter. As a result, the transition matrix A is given by the two transitions:

$$p = P(B|G); \quad q = P(G|B); \quad A = \begin{bmatrix} (1-p) & p \\ q & (1-q) \end{bmatrix}$$

As mentioned before, the mathematical formula for CoAP observed loss ratio P_L is given by:

$$P_L = \frac{\pi(2^r T)}{\pi(2^r T) + \pi(R)} \tag{4.29}$$

which refers to the average number of CoAP observed losses divided by the total number of total CoAP transmitted messages. Substituting simplifies equation (4.29) to:

$$P_L = \frac{p(1-q)^r}{p(1-q)^r + (1-(1-q)^{r+1})} \tag{4.30}$$

$$= \frac{p(1-q)^r}{1 + (p+q-1)(1-q)^r} \tag{4.31}$$

$$= P_L \text{ obtained by the Simple Gilbert Loss Model (Chapter 3)} \tag{4.32}$$

Consequently, our presented formulas in chapter 3 can be handled by using the 1-dimensional *Markov Chain* for Simple Gilbert without any hidden aspects and the same results will be found. Here, we re-

Description	Value
RTO	2 s
r	4
Z (Total packet size)	100 bytes
Y (Payload of the packet received)	48 bytes

Table 4.2: CoAP parameters used in Cooja/Contiki

mind the reader about the gradual shift strategy that we followed during our research work.

4.6 Experimental environment

In this section, Cooja platform [48], being a common tool for performing evaluation in IoT environment with the ContikiOS [21], will be used to validate our model. The importance of Cooja is the emulation of the constrained nodes hardware taking into account the hardware specifications and processing capabilities available in real IoT nodes. More details regarding the Cooja environment will be presented in chapter 6. In the validation experiments, we consider a ring topology presented in Fig. 4.7. In the network, the RPL router is node 1, CoAP receiver is node 2 and other nodes acting as CoAP senders. Z1 motes [2] are used for CoAP nodes while a TMote Sky mote is used for the RPL router. All CoAP clients are sending messages that are routed through the RPL border router to the server using the IPv6 address. The RPL router serves only as relay for CoAP messages. It is worth noting that Z1 motes offer more ROM space, giving more room for the code imposed by CoAP protocol or any other enhanced version of the protocol. An initialization phase around 100 seconds for each simulation is allowed. No results are collected during this phase. Once the network is initialized, CoAP clients generate messages which are directed towards CoAP server. NSTART is set to 1 as per CoAP default specification which means that only 1 message is sent at a time per node. The value of Transmission (TX) ratio is: 100%. The simulations of the different scenarios have a 15 min duration and are repeated 5 times for each scenario. We use different CoAP parameters which are summarized in table 4.2. To challenge our experiments more, we used diverse values of Gilbert-Elliott parameters listed in table 4.3. More precisely, the values of p,q,k, and h are imposed in CoAP engine file. We carried out all our experiments after we corrected a bug with the existing CoAP implementation in Contiki. In particular, the sender did not wait for the timeout after the last retransmission.

Remark: Cooja/Conitki parameters and hardware specifications of the motes are presented in details in chapter 6.

p	q	k	h
0.1	0.6	0.8	0.2
0.2	0.6	0.8	0.2
0.3	0.6	0.8	0.2
0.5	0.6	0.8	0.2
0.7	0.6	0.8	0.2
0.1	0.5	0.4	0.4
0.1	0.5	0.5	0.4
0.1	0.5	0.6	0.4
0.1	0.5	0.7	0.4
0.1	0.5	0.8	0.4

Table 4.3: Gilbert-Elliott simulation parameters

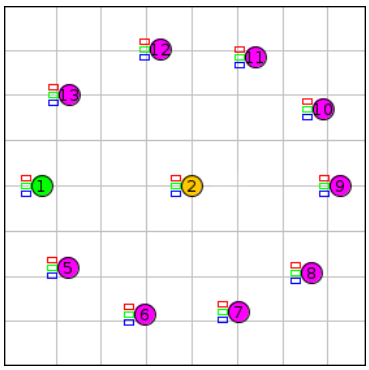
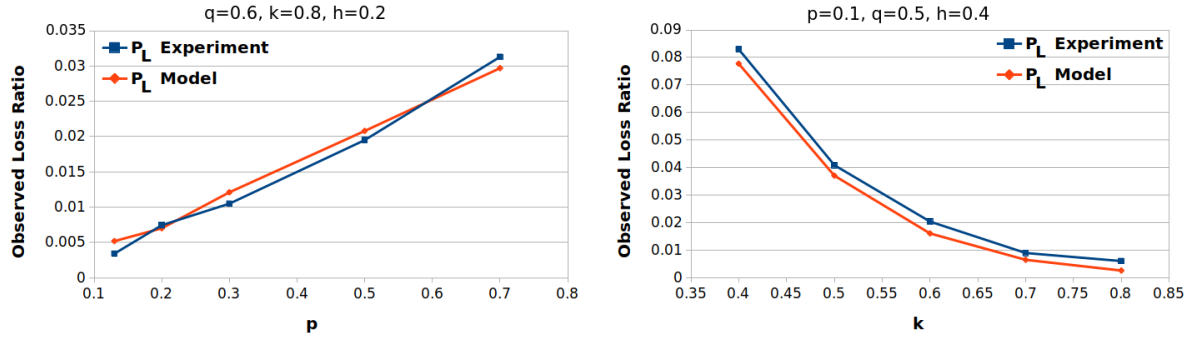


Figure 4.7: Network topology for Model validation with Cooja/Contiki OS environment

4.7 Model Validation via experiments

To check the correctness of our model, we compared it to the experimental results shown in Fig. 4.8. In the presented figure, we show CoAP Observed Loss Ratio (P_L) while varying the probabilities p and k respectively. The blue and red plots show the values of Observed losses while varying p from 10% to 70%, with q set to 60%, k set to 80% and h to 20%. Loss ratio refers to the case where all the re-transmission attempts are lost and ack is not received. When disruption happens, packets may be lost. The packet loss triggers the retransmission of the packet or transmission of a new packet. As figure 4.8a indicates, when p increases, observed loss ratio P_L increases accordingly. On the other hand, in Fig. 4.8b, we vary k from 40% to 80% while setting p to 10%, q to 50%, and h to 40%. We observe that as k increases, CoAP observed losses decreases and this is expected since the complement $1 - k$ which represents the loss probability in the good state is decreasing.

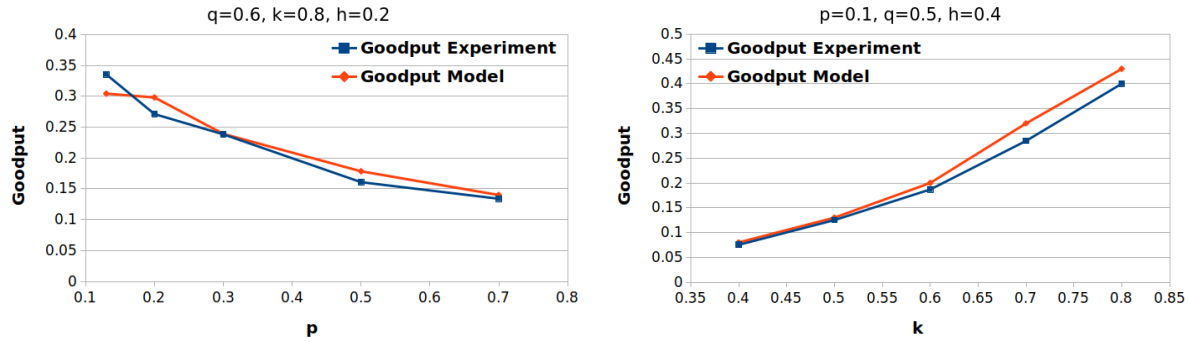
The experimental and model results of Goodput are presented in Fig. 4.9, where we vary p and k as mentioned in table 4.3. As can been seen, in the case of a low network losses ($p > 0$), the goodput increases in both the model and experiments for different values of Gilbert-Elliott parameters (Fig. 4.9a and Fig. 4.9b). The graph shows that the ratios attained from the model and experiments are similar



(a) Observed Loss Ratio P_L results while varying p

(b) Observed Loss Ratio P_L results while varying k

Figure 4.8: Experimental results vs. Model results for Observed Loss Ratio P_L



(a) Goodput results while varying p

(b) Goodput results while varying k

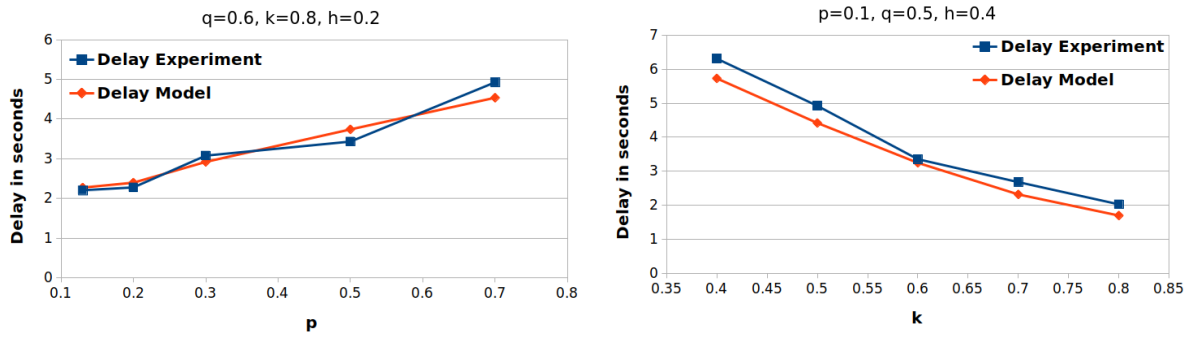
Figure 4.9: Experimental results vs. Model results for Goodput GP

and almost exactly the same in many cases.

To test the accuracy of the delay model, we compared it to experimental results as per Fig. 4.10, again with different values of p and k while fixing the remaining parameters as seen before. Fig. 4.10 shows the delay of CoAP packets of successful transmission with different values of loss probability p: 10%, 20%, 30%, 50% and 70%. As mentioned before, the delay of packet transmission in CoAP is affected by the network loss which is imposed by the loss probabilities. With high values of p which represents high network losses, the delay increases to around 3 sec (x-point 0.3 in Fig. 4.10a) and to 5 sec (x-point 0.7 in Fig. 4.10b). Here again, as we can see from the results, the loss probability plays a significant role in increasing the delay.

4.8 CoAP Performance Analysis via the Analytical Model

Figures 4.11a to 4.11d show the achieved goodput GP versus the available bandwidth in the network while varying the four Gilbert-Elliott parameters h, $k > h$, p and q in turn. The available bandwidth corresponds to $\frac{kq+hp}{(p+q)R}$ since CoAP is limited to one packet per RTT. The wastage of the bandwidth is varying between 32% and 80%. The more the losses in the network, the more the wastage. Here,



(a) Delay results while varying p

(b) Delay results while varying k

Figure 4.10: Experimental results vs. Model results for Delay $\bar{D}_s$

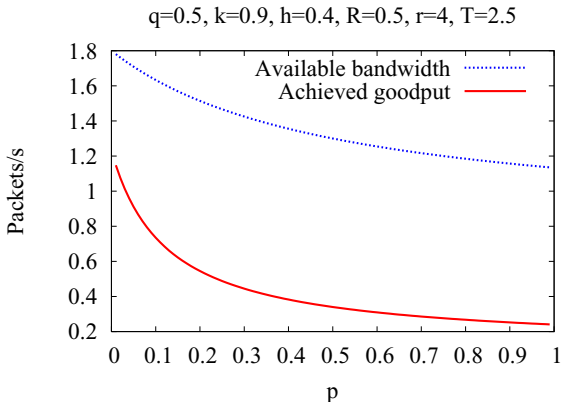
the round trip time R is 500ms. The wastage becomes much larger when R is smaller as shown also in Figure 4.13b where we vary R from 10ms to 2s which is the default CoAP initial timeout without randomization. Even with the excessive $T = R$, the wastage is 10% or more depending on the network state. The wastage is also due to the backoff procedure. Notice though that in reality the timeout should be sufficiently higher than the round trip time.

Figure 4.11f and 4.11e show the experienced loss probability and the overhead respectively for different values of the retransmission counter r while increasing the network loss probability $1 - h$ of the bad period. To improve the reliability of CoAP, it is better to increase r. However, the overhead is increased and vice versa. It is intricate to control the tradeoff with the parameter r and with retransmissions. Actually, retransmissions help only recovering from losses but they can not reduce losses without increasing the overhead.

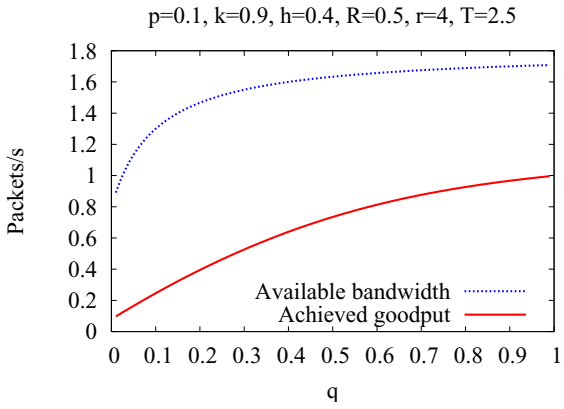
Fig. 4.12a presents the achieved goodput versus the optimal goodput while varying the available bandwidth. As can be seen, when the available bandwidth is less than $1/RTT$, the goodput wastage is high. In Fig. 4.12b, we vary the round trip time R from 0.05 s to 1.9 s. As we have seen previously, for small values of R, the achieved goodput is low.

Finally, in Figure 4.13a, we vary R from 500ms to 3.5s and we compute the goodput using backoff factors according to the Variable Backoff Factor (VBF) algorithm suggested in [10, 14]. VBF uses a 1.5, 2 or 2.5 multiplication factor depending on the value of the retransmission timeout, which in turn depends on R. Even though in this work we presented the above formulas with the default CoAP multiplication factor of 2, it is easy to extend our model to compute for any backoff factor b. Indeed, following the same reasoning in section 4.2, we obtain the mathematical models of CoAP performance metrics. For instance, goodput GP (4.26) and delay $\bar{D}_s$ (4.27) equations become:

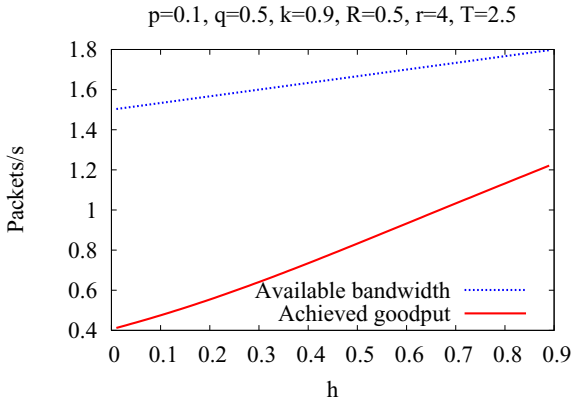
$$GP = \frac{\pi(R)}{\pi(R) \times R + \sum_{i=0}^{r} \pi(b^i T) \times b^i T} \tag{4.33}$$



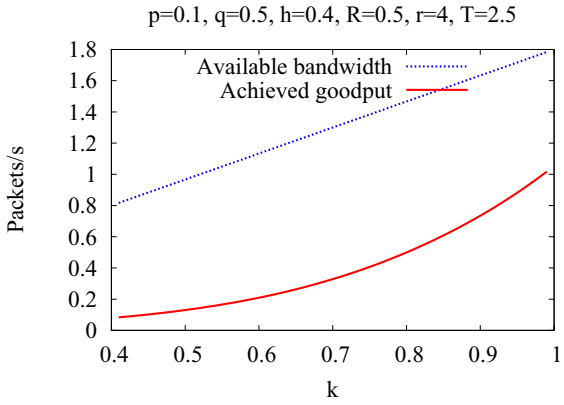
(a) Goodput vs. available bandwidth while varying p



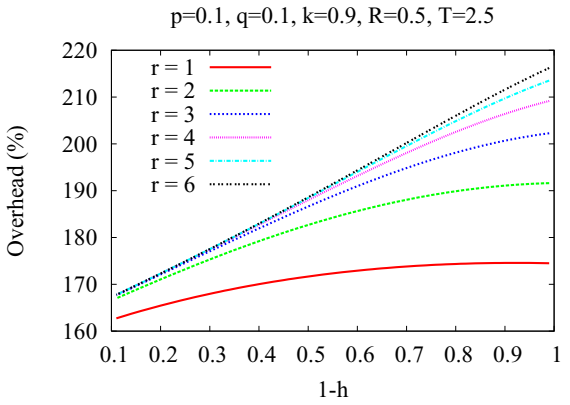
(b) Goodput vs. available bandwidth while varying q



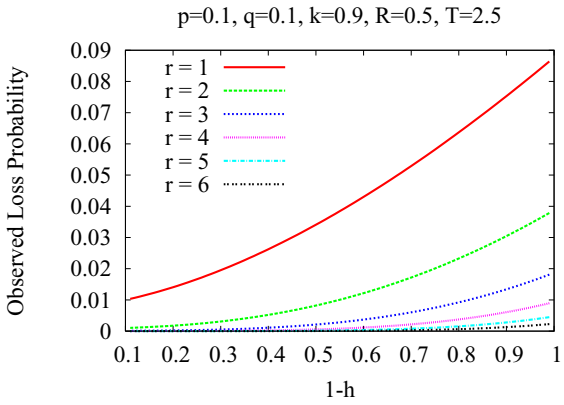
(c) Goodput vs. available bandwidth while varying h



(d) Goodput vs. available bandwidth while varying k

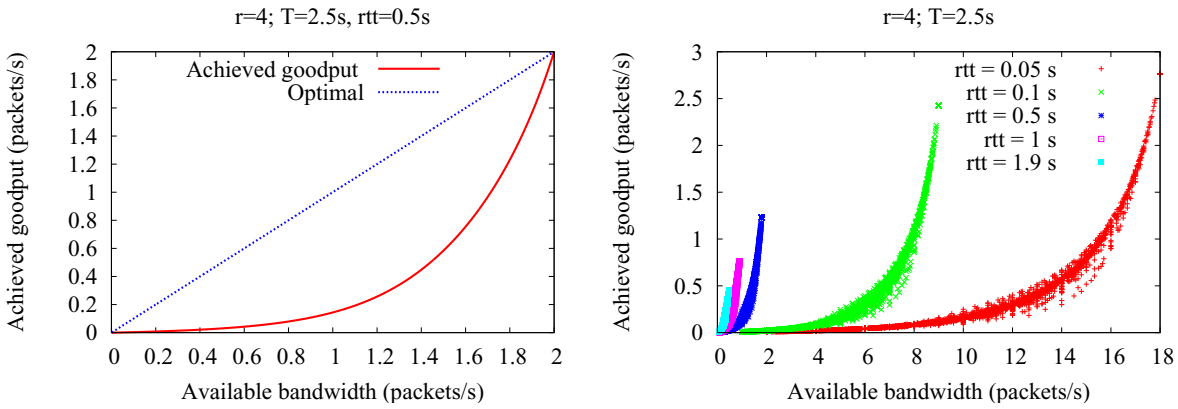


(e) Impact of the retransmission counter r on overhead. Z=89bytes, Y=37bytes



(f) Impact of the retransmission counter r on reliability

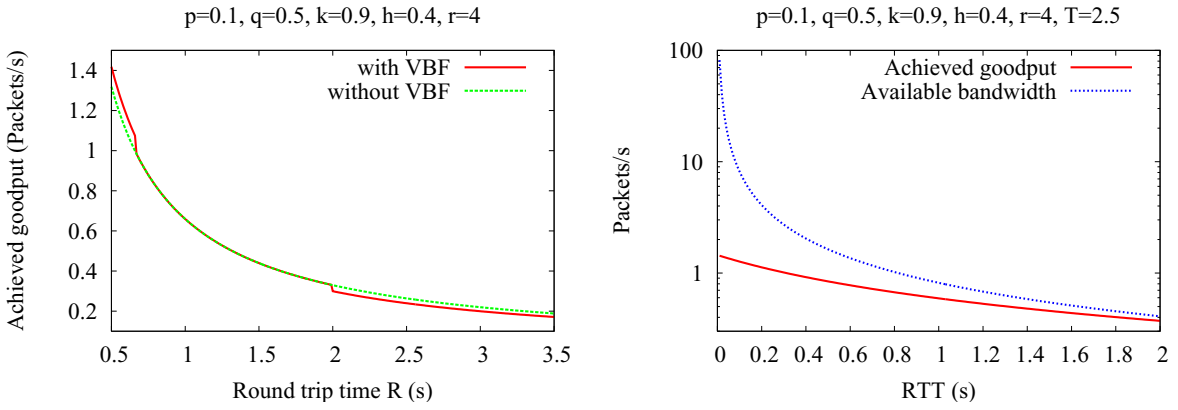
Figure 4.11: Performance evaluation results using our Makovian model under Gilbert-Elliott losses



(a) Achieved vs. Optimal goodput

(b) Achieved goodput for different RTT values

Figure 4.12: Performance evaluation results while varying the available bandwidth



(a) Impact of using variable backoff factors VBF [10]

(b) Impact of round trip time R on bandwidth utilization

Figure 4.13: Impact of round trip time RTT and variable backoff factors on goodput

$$\bar{D_s} = D_{ow} + T\frac{\sum_{i=0}^{r-1} b^i \pi(2^i T) - \pi(2^r T)\,(b^r - 1)}{\pi(R) \times (b-1)} \tag{4.34}$$

Referring back to Figure 4.13a, we see that the benefit from VBF is minor and depends on R. Actually, it is hard to achieve a fine-grained control using distinct backoff factors even if their number is large.

4.9 Conclusion

We developed a novel model based on Markov Chains to derive the performance of CoAP under the Gilbert-Elliott loss model and validated it via simulations. We showed how to compute several performance metrics using closed form expressions and with a time complexity no more than $O(r)$ with r is the maximum re-transmission limit. Thus, the model can also be used to tune CoAP parameters dy-

namically by IoT devices in order to adapt to network losses. The use of different models, as presented in this chapter and chapter 3, provides confidence in our analysis, since we find the same results using several approaches. In fact, we discovered a bug related to the implementation of CoAP in the COOJA simulator: the r-th re-transmission attempt did not wait for an ACK reception. Thus far, we carried out the modelling work and it allowed to study precisely the performance of CoAP and the effect that different parameters have on its behavior.

Synthesis. In order to improve the goodput and reliability while reducing overhead, first we have to reduce the retransmission timeout as much as possible to be close to the round trip time so that *congestion detection* is improved and the reaction to the congestion is accelerated. However, the timeout should not become lower than RTT to avoid creating *spurious* transmissions which are very harmful because they waste resources of the constrained device. Second, retransmissions and the backoff procedure are not the right action to counter congestion efficiently. For *congestion counteraction*, it is better for the CoAP sender to deploy a "real" congestion control mechanism in order to decide correctly how long to *wait* before sending the next packet and avoid losses. The inter-sending delay should be inversely proportional to the available bandwidth, and this is hard to achieve through the backoff mechanism even with variable backoff factors. The above results and analysis suggest to follow a rate-based approach for congestion control rather than a backoff-based approach. Nevertheless, the algorithms must be developed with minimal instructions so it can reduce the processing overhead and increase the lifetime of the constrained devices. That is why also spurious transmissions must be reduced as much as possible to reduce energy consumption. These algorithms will be covered in the next chapters.

5

Congestion Detection: Improving Retransmission Timeout Calculation

Up to this chapter, we focused on analyzing and evaluating the Constrained Application Protocol (CoAP) via modelling. In Chapter 2, we introduced the improvements carried out by previous works in order to enhance CoAP congestion control components, and in particular, the Retransmission Timeout (RTO) calculation which is used to enhance congestion detection. As a matter of fact, timeouts are the only indicator used in CoAP to detect losses and losses are used as a strong indicator to detect congestion. The second component, congestion counteraction, defines the procedures taken after RTO expiration. Then, we analyzed and presented their corresponding shortcomings. The results show that recent mechanisms suggested to calculate RTO are not sufficient and can be improved further. In this chapter, we present and evaluate our proposed algorithm to efficiently calculate RTO and overcome the addressed shortcomings. Using our Python simulator that we developed to analyze different algorithms, different network scenarios are considered to evaluate the performance of RTO estimation algorithms. The presented results prove that the proposed algorithm provides significant improvement in RTO estimation for the majority of the considered cases. The rest of the chapter is organized as follows: In Section 6.1, the design of our RTO calculation proposed algorithm is presented. Section 5.2 describes in detail our new simulator environment. Performance evaluation results are presented in Section 6.2 where we compare our algorithms with previous works. Section 5.4 concludes the chapter.

5.1 Proposed algorithm

In this section, we propose our RTO estimation algorithm in order to overcome the previous shortcomings that we have highlighted in Chapter 2 and improve RTO calculation. We design a new algorithm to calculate the initial retransmission timeout RTO_{init} more efficiently to improve congestion detection. The outcome will serve as the congestion detection portion of our new proposed congestion control protocols. The congestion counteraction part of these protocols will be explored in Chapter 6. In both algorithms, RTO_{init} is used only to detect losses and it is never doubled or modified in case of loss. Also, no dithering techniques are implemented, thus we call it also simply RTO.

5.1.1 RTO Calculation Version 1

We present in this section our first version of RTO estimation algorithm. At first, our main challenge is to keep the algorithm simple and adapted to IoT constrained devices. Our new algorithm is based on the concept of exponentially weighted moving average (EWMA) with minimal instructions. The pseudo code is presented in Algorithm 2. SRTT is the maintained average round trip time and RTTVAR is the computed variation of RTT. SRTT and RTTVAR are updated using the weight α (lines 1 and 3).

Algorithm 2 RTO calculation algorithm - Version 1

1: $SRTT = (1 - \alpha)SRTT + \alpha R$
2: **if** $R > SRTT$ **then**
3: RTTVAR $= (1 - \alpha)$RTTVAR $+ \alpha|SRTT - R|$
4: **end if**
5: $RTO = SRTT + 4 *$ RTTVAR

When the first RTT sample value is measured, the algorithm initializes the following variables:

$$R \longleftarrow RTT \tag{5.1}$$

$$SRTT \longleftarrow R \tag{5.2}$$

$$RTTVAR \longleftarrow \frac{R}{2} \tag{5.3}$$

$$\text{if no RTT sample is obtained: } RTO \longleftarrow 2\,sec \tag{5.4}$$

At the beginning of a new set of transmissions, the variables are initialized. RTO value is initialized to the default CoAP RTO value (2 seconds), if RTT sample could not be measured due to a missing ACK. Otherwise, RTO is estimated for each new R as shown in Algorithm 2. In this version, we update RTTVAR only if the measured RTT R is greater than the maintained average $SRTT$. This condition ($R > SRTT$) is used to adapt RTO estimation when RTT increases which is useful in reducing spurious transmission. The case when RTT decreases is not handled in this version to keep the code very

simple, however, this will be handled in the version 2. Finally, RTO for the next transmission is calculated using the smoothed value of RTT $SRTT$ to which we add the RTT variance RTTVAR multiplied by K (line 5). The concept of using $K \times RTTVAR$ is based on Jacobson/Karels algorithm (Timeout = Estimated_RTT_Average + 4 $\times$ Estimated_RTT_Deviation) [35].
Afterwards, the sender waits until RTO expires or ACK is received. If RTO expires before ACK reception, the sender retransmits the CON message. If ACK is received, new RTT value is measured. Then, SRTT and RTTVAR are updated again and then the sender waits for RTO expiration or ACK arrival and so on and so forth. Although this version is very simple, it outperforms other RTO calculation algorithms in some cases. However, the drawback is slow convergence when the weight α is small which may cause having RTO smaller than RTT. On the other hand, when $(R < SRTT)$ then RTO value will not change and hence there will be no convergence at all. As a conclusion, this simple version will not be able to converge adequately during high fluctuations in RTT samples. Therefore, we will propose a new version to allow fast convergence on the behalf of additional instructions.

5.1.2 RTO Calculation Version 2

Another challenge in RTO calculation is the fast convergence during high fluctuations in the network such as sudden increase and sudden decrease of RTT. Precisely, if the CoAP sender uses a smaller RTO than RTT, it will generate a spurious transmission and the sending rate will be greater than 1 packet per RTT which may worsen the situation in case of congestion. Also, RTO should not be much higher than RTT because the sender will wait long before detecting the loss and before sending the lost and the next packets. The sender might also skip some good time intervals where packets can be successfully delivered. Another challenge is the limited capacities of some IoT constrained devices where the algorithm must be developed with minimal instructions so it can reduce the processing overhead and increase the lifetime of the constrained devices. That is why also spurious transmissions must be reduced as much as possible to reduce energy consumption.

Our new algorithm is still based on the concept of exponentially weighted moving average (EWMA) with several modifications from previous works to minimize further RTO while reducing the number of spurious transmissions, and in the same time reduce the number of instructions. The pseudo code is presented in Algorithm 3. Similar to Algorithm 2, SRTT is the maintained average round trip time and RTTVAR is the computed variation of RTT. SRTT is updated using the weight α (line 1). However, in this version, RTTVAR is updated using the weights α and γ (lines 2 to 6).

In this version, we update RTTVAR using a different weight depending on the value of the measured RTT R compared to the maintained average $SRTT$. This condition $(R > SRTT)$ is necessary to adapt RTO estimation adequately to RTT fluctuations, on the behalf of some additional complexity. However, this condition was sufficient to estimate RTO correctly and the algorithm is still simpler than

Algorithm 3 RTO calculation algorithm - Version 2

1: $SRTT = (1 - \alpha)SRTT + \alpha R$
2: **if** $R > SRTT$ **then**
3: RTTVAR $= (1 - \alpha)$RTTVAR $+ \alpha|SRTT - R|$
4: **else**
5: RTTVAR $= (1 - \gamma)$RTTVAR $+ \gamma|SRTT - R|$
6: **end if**
7: $RTO_{init} = SRTT + K *$ RTTVAR

pCoCoA. The weights are fixed so that convergence is faster ($\alpha > \gamma$) when R increases ($R > SRTT$) because if convergence is not fast enough, there is a high risk to have $RTO < R$ causing spurious transmissions which is important to avoid utmost as we said before. In contrast, convergence is slower when R decreases ($R < SRTT$) in a preventive approach in order to observe first if this reduction is permanent or transient. Otherwise, if we converge fast, the estimated RTO can flip down below next R values causing again spurious transmissions. The weights used to compute RTTVAR should be tuned to find a good compromise between RTO reduction and spurious reduction as well.

Finally, in the algorithm, RTO for the next transmission is calculated also using the smoothed value of RTT $SRTT$ to which we add the RTT variance RTTVAR multiplied by margin factor K (lines 7 to 12). Another challenge though is choosing the right value for K. K plays an important role in estimating RTO value. When it is set to 4 for all transmissions, performance can be reduced. Indeed, K should be preferably chosen dynamically according to the spurious status. Hence, we improved the second version of our RTO calculation algorithm by adding another block for computing K dynamically. This is presented in the next section.

5.1.3 RTO Calculation Final Version

The pseudo code of the final version is presented in Algorithm 4.

Algorithm 4 RTO calculation algorithm - Final Version

1: $SRTT = (1 - \alpha)SRTT + \alpha R$
2: **if** $R > SRTT$ **then**
3: RTTVAR $= (1 - \alpha)$RTTVAR $+ \alpha|SRTT - R|$
4: **else**
5: RTTVAR $= (1 - \gamma)$RTTVAR $+ \gamma|SRTT - R|$
6: **end if**
7: **if** $spurious\,is\,true$ **then**
8: $K = 7$
9: **else**
10: $K = 4$
11: **end if**
12: $RTO = SRTT + K *$ RTTVAR

If the previous transmission is spurious, then RTO value must be increased by increasing K to force the sender to wait for a longer period (lines 7-8). A lower K value should be used when spurious is not detected (lines 9-11). The latter mechanism increases the value of RTTVAR when a spurious transmission is detected, thus limiting successive spurious transmissions.

The code presented in this section summarizes all the instructions to be performed. The initialization phase and the sequence of operations presented in Section 5.1.1 is common for this version and version 2 as well.

5.1.4 RTO Calculation: Choosing the weights

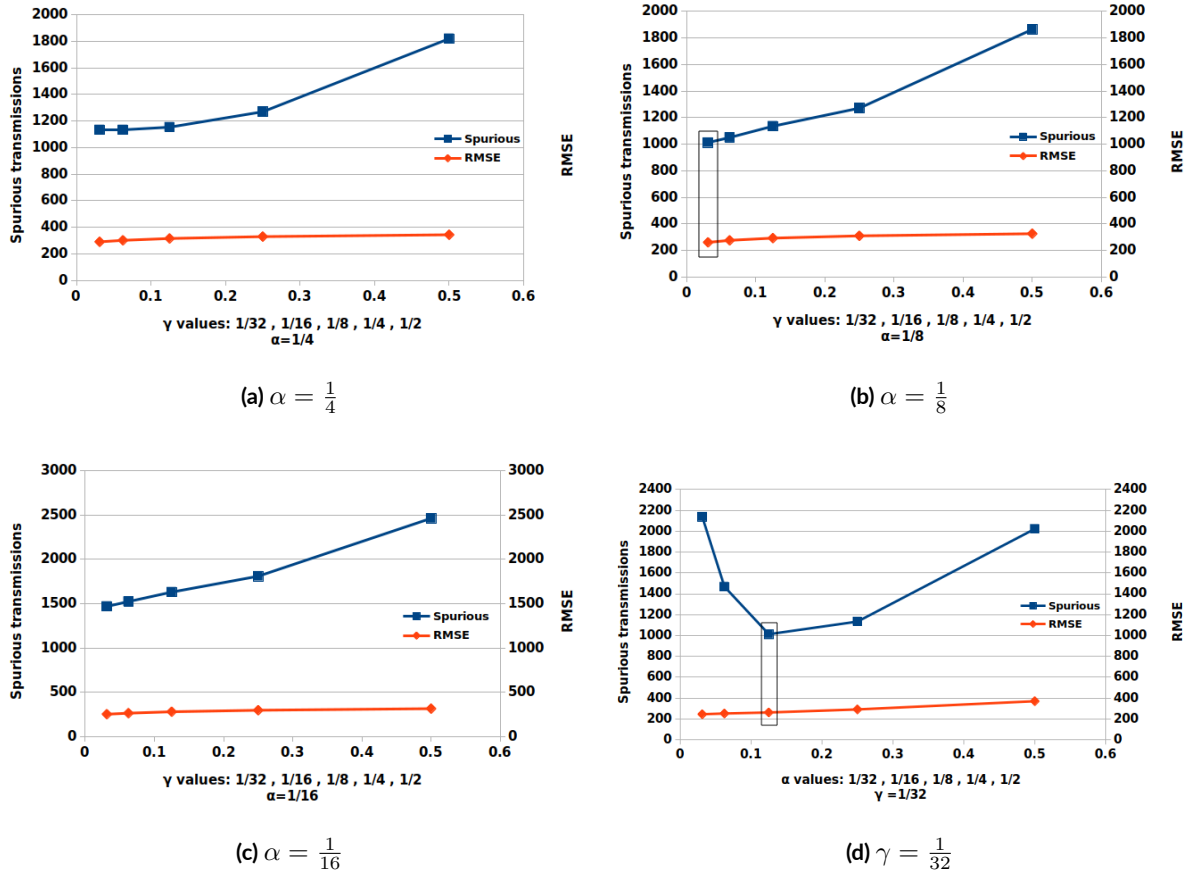
In order to choose a good combination of values for the parameters α, γ and K that support our design objectives, different values in different network scenarios were tested and analyzed. We show here only full analysis with one of the most challenging scenarios where RTT is varied continuously according to a uniform distribution between 500ms and 600ms during 100 times, then a sudden change to a normal distribution with mean 100ms and a standard deviation of 20ms during 100 times, then a sudden return to the uniform distribution and so on and so forth. Besides, a high increase of RTT to 1 second is generated 5 times every 1000 transmissions (Table 5.1 - Scenario 35). Then, while varying the weights, we compute both the total number of spurious transmissions and the Root Mean Square Error (RMSE) which measures the difference between estimated RTO and measured R values.

In Fig. 5.1a, when we fix the value of α to $1/4$, then any value of γ will lead to a number of spurious transmissions around or greater than 1200. It is impossible to find a good combination with this α value. If we tend to reduce α to any value less than $1/4$, spurious transmissions increase more and more. Therefore, we omit the weight $1/4$ and all other weights below it.

As per Fig. 5.1a, the results show that when the value of α is fixed to $1/16$, RMSE is reduced but spurious transmissions are increased to more than 1500. In this case, any chosen value of γ will not reduce the tradeoff between RMSE and spurious transmissions, therefore the α value $1/16$ can not be combined with any γ value. Consequently, the more we increase α, the more spurious transmissions, therefore, all α values greater than $1/16$ are also dropped.

According to the results from the figures 5.1c and 5.1b, when we fix the value of α, then a low value of γ reduces both spurious transmissions and RMSE. In Figure 5.1d, when γ is fixed to 1/32, a value of 1/8 for α is a good compromise between spurious and RMSE. Thus, we fix $\alpha = \frac{1}{8}$ and $\gamma = \frac{1}{32}$. This result confirms our design rule $\alpha > \gamma$.

Regarding the K parameter, in Fig. 5.2a, we fix K to 7 if there is a spurious transmission and we vary the value of K in the case when there is no spurious from 2 to 7. We observe that the higher these K values, the lower RMSE and number of spurious. However, the value of 4 is a good compromise since after this value, the improvement is minor compared to the improvement from the values of 2 to 4.



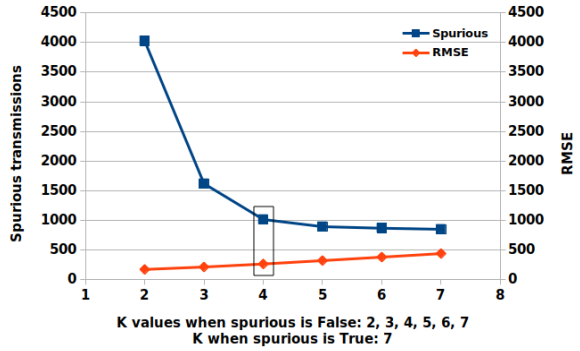
(a) $\alpha = \frac{1}{4}$

(b) $\alpha = \frac{1}{8}$

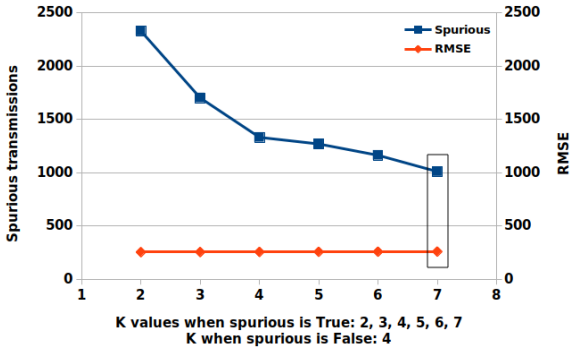
(c) $\alpha = \frac{1}{16}$

(d) $\gamma = \frac{1}{32}$

Figure 5.1: Spurious transmissions and RMSE values while varying α and γ weights

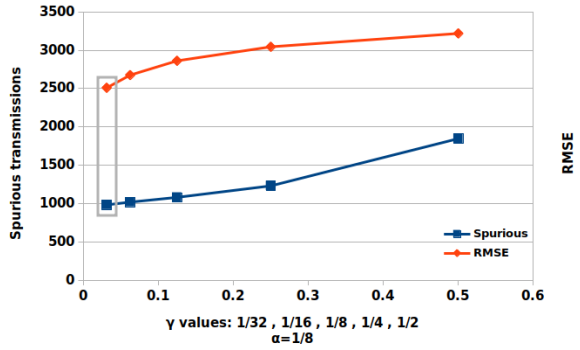


(a) K = 7 when there is a spurious

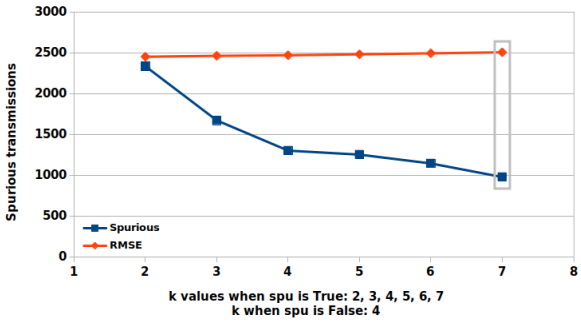


(b) K = 4 when there is no spurious

Figure 5.2: Spurious transmissions and RMSE values while varying K parameter



(a) $\alpha = \frac{1}{8}$



(b) K = 4 when there is no spurious

Figure 5.3: Spurious transmissions and RMSE values for the optimal values of α and K

In Fig. 5.2b, we fix K to 4 if there is no spurious transmission and we vary the value of K in the case when there is a spurious from 2 to 7. We observe that the value of 7 reduces both spurious and RMSE. Thus, the portion of the algorithm that computes K is useful to reduce further RMSE and the number of spurious transmissions.

Finally, to assure the adequacy of our selection, we show the performance of our algorithm with the chosen weights in an another challenging scenario where RTT is varied continuously according to a uniform distribution between 500oms and 600oms during 100 times, then a sudden decrease to a normal distribution with mean 100oms and a standard deviation of 200ms during 100 times, then a sudden increase to the uniform distribution and so on and so forth. Additionally, a high increase of RTT value to 10 seconds is generated 5 times every 1000 transmissions (Table 5.1 - Scenario 33). As can be seen from Fig. 5.3a and Fig. 5.3b, the chosen values of α, γ and K reduce spurious transmissions and RMSE values in this network scenario as well. In fact, all other simulation results, not shown here, has confirmed this setting.

5.2 Ad hoc Simulation Environment

We developed in Python language a dedicated simulator to analyze deeply CoAP algorithms in a controlled environment. Indeed, the simulator can generate different patterns of Round Trip Time (RTT) or reuse real RTT traces. It is able to emulate the available bandwidth observed by a CoAP sender and also emulate loss of packets. Hence, congestion and its strength and duration are emulated in a repeatable and supervised manner. The evolution of network conditions over time can be kept exactly the same from one simulation to another so that comparison between different algorithms is fair. Besides, compared to other simulation environments, our Python simulator produces results much faster. Hence, it is possible to test quickly many network scenarios and CoAP protocol variants. Moreover, the developed simulator helped us to generate and distinguish between network losses caused by interference or congestion

5.3 Performance evaluation

In this section, we use our python simulator presented in section 5.2, to evaluate the efficiency of our algorithm of RTO calculation, and to study and compare its performance against other previous algorithms. Our new algorithm of RTO estimation is compared with algorithms, in particular, pCoCoA [14], and CoCoA+ since couple of recent works used their method in RTO computation without any changes.

Table 5.1 shows RTT network scenarios that will be used in this evaluation.

ID	Network scenario				
	Period 1		Period 2		Additional events
	Number of Transmissions	RTT Distribution	Number of Transmissions	RTT Distribution	
1	100000	Pareto (3, 1000)	N/A	N/A	Yes
2	100000	Pareto (4, 1000)	N/A	N/A	Yes
3	100000	Pareto (5, 1000)	N/A	N/A	Yes
4	100000	Pareto (6, 1000)	N/A	N/A	Yes
5	100000	Real trace	N/A	N/A	No
6	100000	Real trace	N/A	N/A	No
7	100	Uniform(100,300)	100	Normal(3000,100)	No
8	100	Uniform(100,1000)	100	Normal(3000,100)	No
9	100	Uniform(100,300)	100	Normal(3000,1000)	No

ID	Network scenario				
	Period 1		Period 2		Additional events
	Number of Transmissions	RTT Distribution	Number of Transmissions	RTT Distribution	
10	100	Uniform(100,2000)	100	Normal(4000,1000)	No
11	100	Uniform(300,3000)	100	Normal(6000,200)	Yes
12	100	Uniform(300,3000)	100	Normal(4000,200)	Yes
13	100	Uniform(200,1000)	100	Normal(6000,200)	Yes
14	100	Uniform(100,2000)	100	Normal(6000,200)	Yes
15	100	Uniform(100,500)	100	Normal(4000,200)	Yes
16	100	Uniform(100,500)	100	Normal(4000,1000)	Yes
17	100	Uniform(100,500)	100	Fixed(1000)	Yes
18	100	Uniform(100,500)	100	Fixed(6000)	Yes
19	100	Normal(6000,200)	100	Uniform(200,1000)	Yes
20	100	Normal(6000,200)	100	Uniform(300,3000)	Yes
21	100	Uniform(300,3000)	100	Normal(4000,2000)	Yes
22	100	Uniform(100,2000)	100	Normal(6000,2000)	Yes
23	100	Uniform(300,3000)	100	Normal(6000,200)	No
24	100	Uniform(300,3000)	100	Normal(4000,200)	No
25	100	Uniform(200,1000)	100	Normal(6000,200)	No
26	100	Uniform(100,2000)	100	Normal(6000,200)	No
27	100	Uniform(100,500)	100	Normal(4000,200)	No
28	100	Uniform(100,500)	100	Fixed(6000)	No
29	100	Normal(6000,200)	100	Uniform(200,1000)	No
30	100	Normal(6000,200)	100	Uniform(300,3000)	No
31	100	Uniform(100,300)	100	Normal(3000,300)	No
32	100	Uniform(100,1000)	100	Normal(3000,300)	No
33	100	Uniform(5000,6000)	100	Normal(1000,200)	Yes
34	100	Uniform(5000,6000)	100	Normal(1000,200)	No
35	100	Uniform(500,600)	100	Normal(100,20)	Yes
36	100	Uniform(500,600)	100	Normal(100,20)	No
37	100	Uniform(4000,5000)	100	Normal(1000,200)	Yes
38	100	Uniform(4000,5000)	100	Normal(1000,200)	No

ID	Network scenario				
	Period 1		Period 2		Additional events
	Number of Transmissions	RTT Distribution	Number of Transmissions	RTT Distribution	
39	100	Uniform(1000,1500)	100	Normal(500,50)	Yes
40	100	Uniform(1000,1500)	100	Normal(500,50)	No

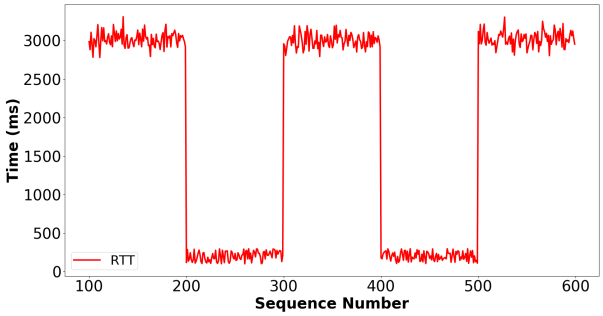
Table 5.1: Different simulation scenarios to challenge RTO calculation algorithms

In the first four scenarios, RTT is varied according to Pareto distribution with different shape parameters (3, 4, 5, 6). We present it in the table in the form of Pareto(alpha, m) where $alpha$ is the shape parameter and m is the scale parameter which represents the smallest value that the Pareto distributed random variable can take. In most of the other scenarios, RTT is varied continuously according to a Uniform distribution between two values during 100 times (Period 1), then a sudden change to a Normal distribution with a given mean and a given standard deviation during 100 times (Period 2), then a sudden return to the Uniform distribution and so on and so forth. The total number of transmissions is 100000 in all scenarios. Normal distribution is presented in the table in the form: Normal(n, std) where n is the mean value and std is the standard deviation. Fig. 5.4a shows scenario 7 where we switch from a high RTT average with a low deviation to a low average with a low deviation and Fig. 5.4b shows scenario 8 where we switch from a high RTT average with a low deviation to a low average with a high deviation.

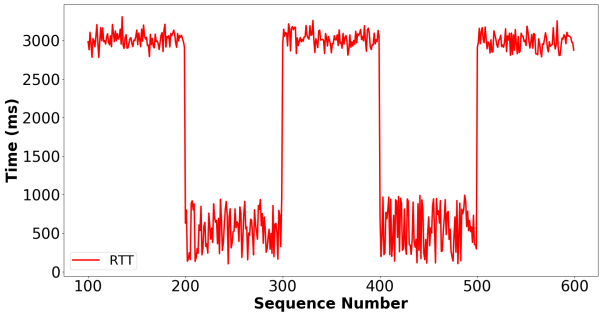
On the other hand, Fig. 5.4c shows scenario 9 where we switch from a high RTT average with a large deviation to a low average with a low deviation and Fig. 5.4d shows scenario 10 where we switch from a high RTT average with a large deviation to a low average with a high deviation.

Besides, to challenge more the algorithms, in some simulations, we can have additional events corresponding to a high increase of RTT to 10 seconds generated 5 times every 1000 transmissions. In scenario 35, the additional event corresponds to an increase in RTT to 1 second generated 5 times after 1000 transmissions. Scenarios 5 and 6 use RTTs from real measurements between two sites. Scenario 5 is from Paris in France to Auckland in New Zealand, and scenario 6 is from Paris to Rennes in France. Fig. 5.4f shows RTTs over time of the real trace of scenario 5, while Fig. 5.4e presents RTT samples of pareto distribution with $shape = 3$ (Scenario 1).

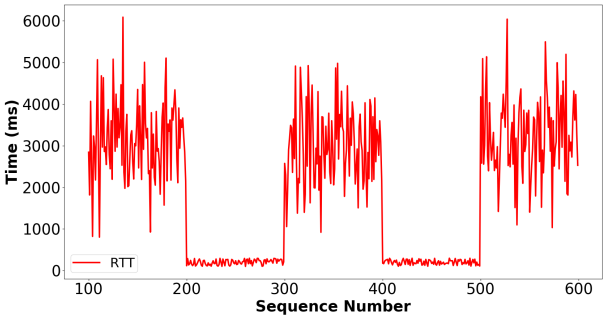
First, the global behavior of the three versions of RTO calculation algorithm is evaluated using two performance metrics: The Root Mean Square Error (RMSE) which measures the difference between RTO and RTT values, and the total number of spurious transmissions observed during the network simulation. Then, we analyze the instantaneous behavior of our proposed algorithm. For brevity, we present the instantaneous graphs of the final version only.



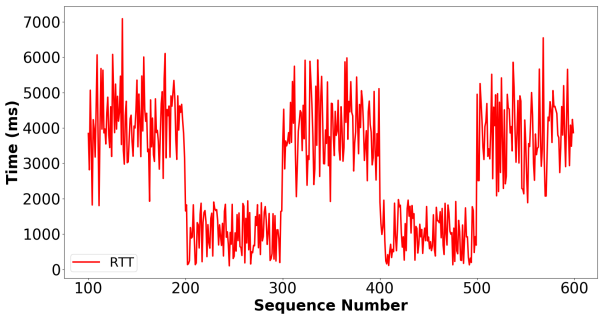
(a) Scenario 7: High average/low deviation to low average/low deviation and vice versa



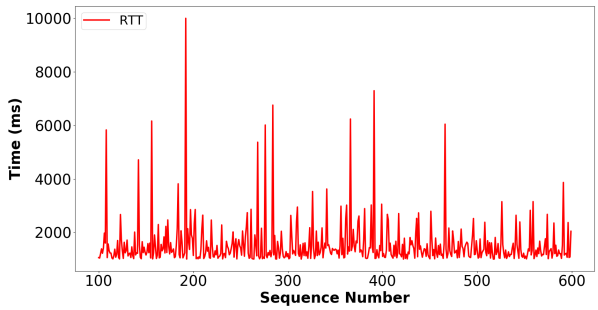
(b) Scenario 8: High average/low deviation to low average/high deviation and vice versa



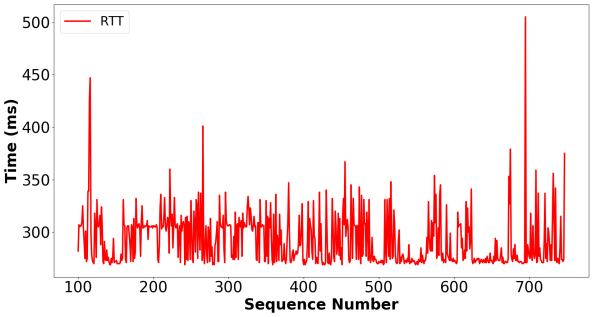
(c) Scenario 9: High average/high deviation to low average/low deviation and vice versa



(d) Scenario 10: High average/high deviation to low average/high deviation and vice versa

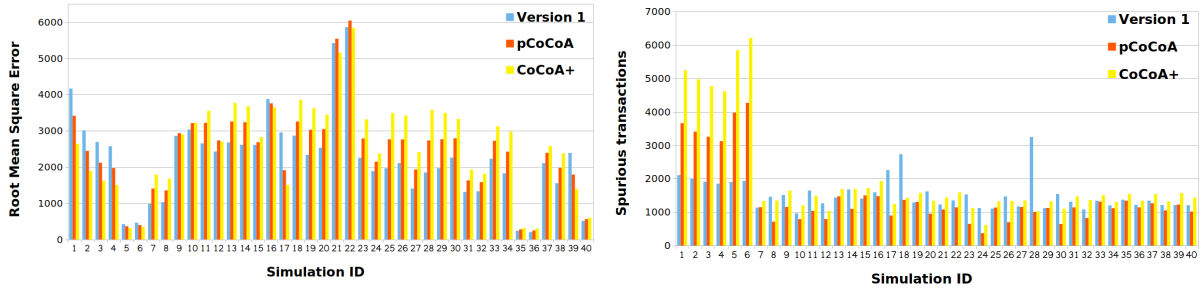


(e) Network scenario 1: Example of pareto scenario



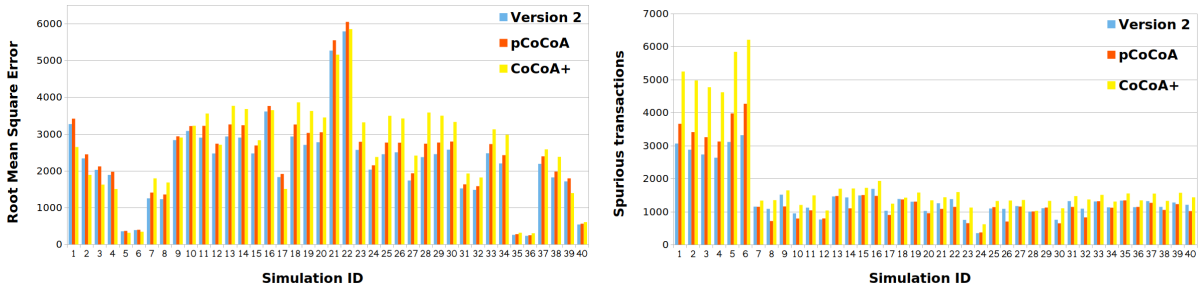
(f) Network scenario 5: Example of real RTT data set

Figure 5.4: Different RTT scenarios to evaluate the algorithms for RTO calculation



(a) RMSE **(b)** Spurious transmissions

Figure 5.5: Spurious transmissions and RMSE results of the proposed algorithm - Version 1



(a) RMSE **(b)** Spurious transmissions

Figure 5.6: Spurious transmissions and RMSE results of the proposed algorithm - Version 2

5.3.1 RTO Calculation Version 1

Fig. 5.5a and Fig. 5.5b show the average of spurious transmissions and RMSE values calculated with the first version of RTO estimation algorithm in different scenarios of Table 5.1. For instance, simulation IDs 1-4 refer to Pareto distributions, simulation IDs 5-6 refer to real RTT scenarios, simulation IDs 7-10 refer to RTT scenarios analyzed instantaneously, simulation IDs 11-40 refer to other challenging RTT sets varying from high average to a low average with different deviations.

As per the results presented in Fig. 5.5a, our RTO calculation algorithm (Version 1) provides lower RMSE than pCoCoA in many network scenarios and very close values to pCoCoA in some other scenarios(IDs 2, 3). Also, with this simple version and as per Fig. 5.5b, we achieved much better results in terms of spurious transmissions in some network scenario (IDs 1,2,3). CoCoA+ achieve better RMSE results in the first 4 scenarios according to the results in both figures (5.5a and 5.5b). For this reason, we had decided to improve the performance further by adding the second part of the algorithm to set the weights of RTTVAR using α and γ.

5.3.2 RTO Calculation Version 2

The second version of RTO calculation is evaluated using the same performance metrics of the previous section: RMSE and the total number of spurious transmissions.

Fig. 5.6a and Fig. 5.6b show the average of spurious transmissions and RMSE values calculated with our new simple RTO estimation algorithm (Version 2) in different scenarios of Table 5.1.

As per the results presented in Fig. 5.6a, the improved version of RTO calculation algorithm (Version 2) now provides lower RMSE than pCoCoA in all the network scenarios. Also, with this new version and as per Fig. 5.6b, we achieved better results in terms of spurious transmissions in more network scenario (IDs 1,2,3,22) and very close results (IDs 4,12,15,16) in many other scenarios. Although CoCoA+ achieves a little better RMSE results in some scenarios (IDs 1, 2, 3, 4), CoCoA+ generates almost double the number of spurious transmissions. We discovered that there is still a margin for improvement. Hence, we had improved the performance further by adding a second chunk to the algorithm in order to dynamically calculate K (Final Version).

5.3.3 RTO Calculation Final Version

In this section, we evaluate the global and instantaneous behavior of the final version of RTO calculation algorithm. For the global behavior, the algorithm is evaluated by means of simulations adopting the same methodology considered in the previous sections.

From Fig. 5.7 and Fig. 5.8, we report the results obtained by our Python simulator using different scenarios of Table 5.1. In particular, we present the average of spurious transmissions and RMSE values calculated with our final version of RTO estimation algorithm. As can be seen from Fig. 5.7, except few minor cases, in all other scenarios, our RTO calculation algorithm achieves better results than CoCoA+. In these few cases, as per Fig. 5.8, CoCoA+ generates up to 76% more spurious transmissions than our algorithm. In the same figure, the number of spurious transmissions generated by CoCoA+ in some cases is even double the amount generated by our RTO calculation algorithm (IDs 5, 6, 8, 25, 26).

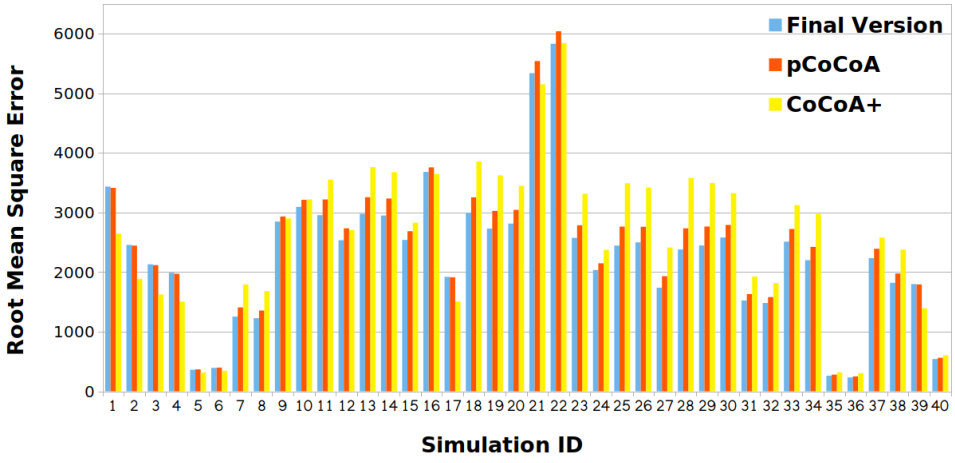


Figure 5.7: RMSE

According to Fig. 5.7 and Fig. 5.8, our proposed algorithm provides lower RMSE and lower number of spurious transmissions in almost all the network scenarios. In the case RMSE of pCoCoA is very

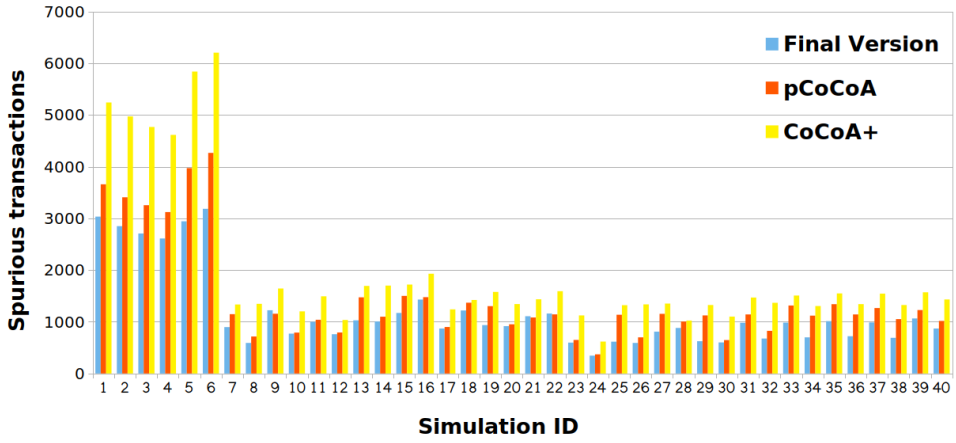


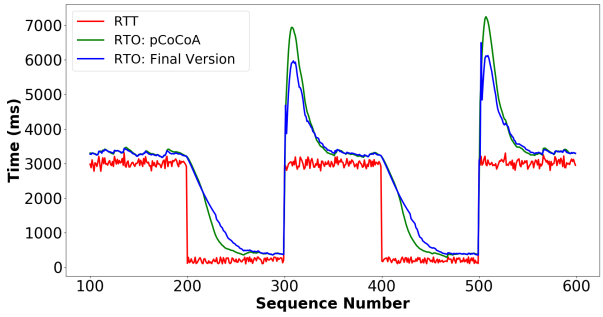
Figure 5.8: Spurious transmissions

close to our algorithm (IDs 1,5,6,17), pCoCoA generates more spurious transmissions. In the case pCoCoA generates few spurious transmissions less than the final version of our algorithm (IDs 9,21), RMSE of pCoCoA is worse. Thus, our RTO calculation achieves a better tradeoff between the two performance metrics.

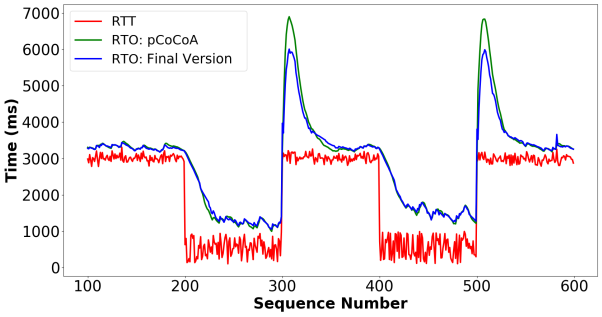
Next, we study the instantaneous behavior of RTO estimation (Final Version) using scenarios 7 to 10 mentioned in Table 5.1. These scenarios cover all possibilities regarding fluctuations of RTT average and variance. Figures 5.9a and 5.9b show that when observed RTT increases suddenly in the network, then RTO in the final version of our algorithm also increases quickly to avoid spurious transmissions due to underestimations of RTO. However, pCoCoA RTO increases more than required which leads to more delay in packet retransmissions. Besides, this increase is not quick enough to avoid spurious transmissions as shown more clearly in Fig. 5.10 that corresponds to the same network scenario as Fig. 5.9a.

Figures 5.9a and 5.9c show a better convergence behavior of our RTO calculation algorithm than pCoCoA when RTO decreases to smaller values with low variations. pCoCoA decreases faster which is risky because it can cause spurious transmissions as shown above. Our RTO calculation converges in a slower manner to be cautious and prevent spurious transmissions. Figures 5.9b and 5.9d show that when RTO decreases to smaller values but with high variations, then both RTO calculation algorithms are similar but still our algorithm reacts better to sudden increase by providing less spurious and low delay.

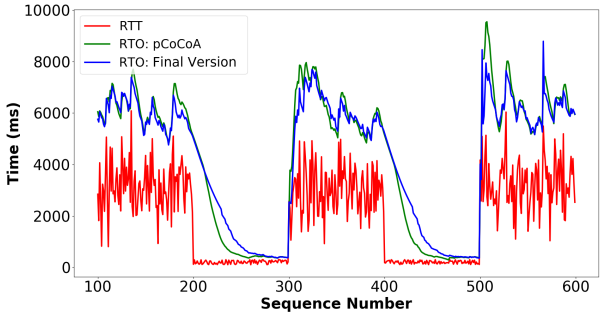
We observe the similar better behavior in Fig. 5.9f where RTT values in the python network simulator correspond to real RTT measurements (Scenario 5 - Table 5.1). Fig. 5.9e shows the same results with the $Pareto$ distribution for RTT values. By examining closely these figures, we can see that our design principle is still applied by converging fastly but not too high when RTT increases fastly, and by converging relatively slowly when RTT decreases fastly to avoid as much as possible spurious trans-



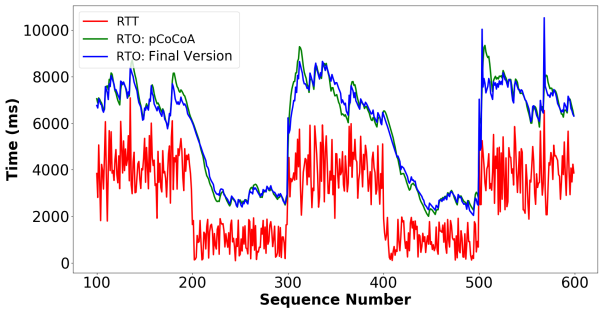
(a) RTO calculation - Scenario 7



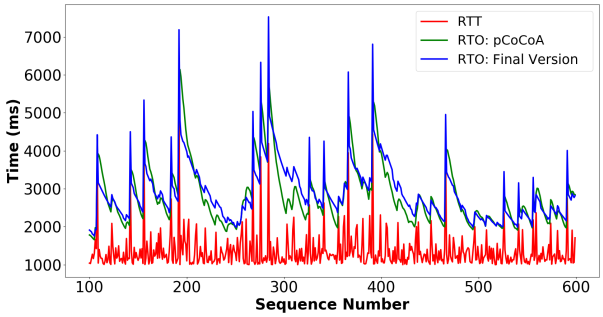
(b) RTO calculation - Scenario 8



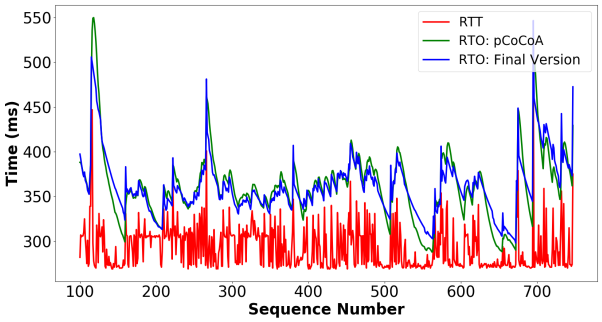
(c) RTO calculation - Scenario 9



(d) RTO calculation - Scenario 10



(e) RTO calculation - Pareto distribution



(f) RTO calculation - Real RTTs

Figure 5.9: Instantaneous behavior of RTO estimation (Final Version) for different network scenarios

missions while trying to minimize RTO values. Indeed, using the python simulator, we were able to calibrate the different parameters of our RTO calculation to reach this design principle as mentioned in Section 5.1.4.

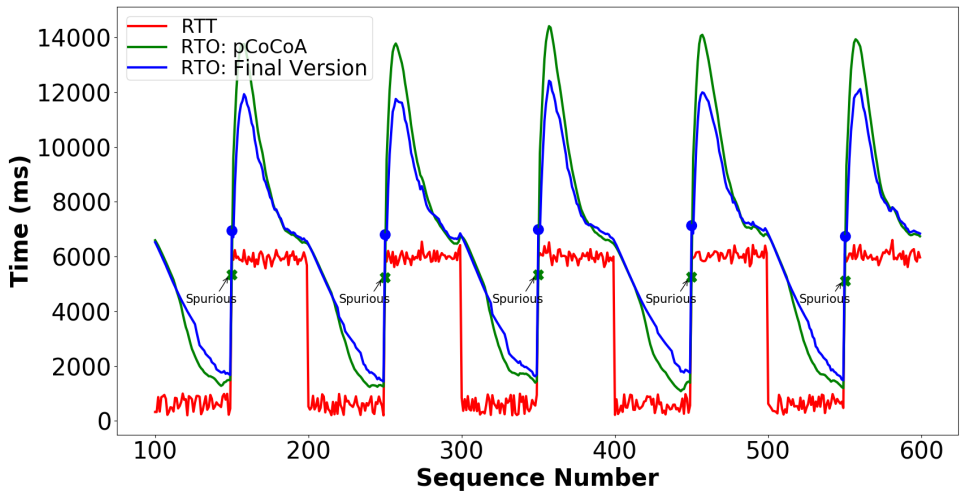


Figure 5.10: Spurious transmissions occurrence graph

5.4 Conclusion

In this chapter, we presented a thorough analysis of RTO calculation algorithms. In particular, we considered challenging RTT scenarios using our Python simulator to evaluate RTO estimation proposed by different mechanisms. The analysis allowed us to avoid some shortcomings of the previously and most recent suggested algorithms for RTO calculation highlighted in chapter 2. We have shown that CoCoA+ generates lot of spurious transmissions and that pCoCoA, with its complex instructions, could not estimate RTO adequately. In order to overcome the issues presented in chapter 2, a new algorithm to calculate RTO more efficiently was proposed. The new algorithm with its simple design, has shown to be effective in most of the considered cases. Moreover, the comparison has shown that our RTO calculation algorithm performs better even in very challenging scenarios. Furthermore, this RTO reduction is useful for all algorithms of congestion control especially those that use only RTO to detect congestion or that use RTO to control congestion through backoff mechanism. In the next chapter, we will explore the second part of the congestion control mechanism: The congestion counteraction.

6

Congestion counteraction: Towards a lightweight rate-based congestion control mechanism

Taking into consideration the hardware limitations of IoT constrained nodes and the limited available bandwidth in IoT networks, congestion control is essential to guarantee the delivery of data. Congestion is observed when the network is overwhelmed by traffic load generated by nodes or the buffer size of IoT devices is exceeded. This is likely to happen when packets between large number of devices are exchanged. CoAP is built over User Datagram Protocol (UDP) which does not implement any congestion control mechanism as the Transmission Control Protocol (TCP). Consequently, CoAP provides a basic congestion control mechanism based on doubling the timeouts to overcome the congestion issue. As we said in Chapter 2, advanced congestion control mechanisms have been proposed by the literature to improve CoAP basic operations. These are twofold: Backoff-based [10, 14, 15, 11] and rate-based [5]. Both approaches implement RTO estimation to detect losses and that was covered thoroughly in Chapter 5. In chapter 2, we presented a deep analysis of backoff and previous rate-based approaches and we showed the limitations and the corresponding shortcomings of the studied algorithms. In this chapter, we focus on the "second step" of the congestion control mechanism, named, congestion counteraction. Firstly, we propose our alternative approach to overcome these drawbacks. The resulting algorithms, named IDC-CoAP and MBC-CoAP, follow the rate-based tactic. Secondly, using our Python simulator and Contiki/Cooja [21, 48], the new presented algorithms are evaluated among previous

works such as CoCoA+, pCoCoA, 4-state and BDP-CoAP. We demonstrate from the results that the new algorithms overcome the issues presented, reduces packet losses and overhead while maintaining high goodput.

6.1 Proposed algorithms: IDC-CoAP and MBC-CoAP

In this section, we propose new mechanisms in order to overcome the previous shortcomings that we have discussed in chapter 2 and improve the overall performance of CoAP. First, we include RTO calculation algorithm (presented in Chapter 5) to calculate the initial retransmission timeout RTO_{init} more efficiently to improve congestion detection. Then, we design the algorithms for congestion counteraction that follows the rate-based approach. The outcome is two new proposed protocols called IDC-CoAP and MBC-CoAP. In both cases, RTO_{init} is used only to detect losses and it is never doubled or modified in case of loss.

As we have observed, most of congestion control algorithms for CoAP follow the backoff based approach where they try to use a different static backoff factor or use a variable backoff factor according to some conditions. We claim that the backoff mechanism is not sufficient to leverage the bandwidth available to the CoAP sender. It is better to deploy a "real" congestion control mechanism in order to decide correctly how long to wait before sending the next packet and avoid losses. The inter-sending delay should be inversely proportional to the available bandwidth, and this is hard to achieve through the backoff mechanism even with variable backoff factors.

Evidently, if one wants to maximize the throughput achieved by the CoAP connection, the sender can send aggressively at its maximum rate to ensure a maximum throughput. However, this will engender a lot of packet losses, retransmissions and possibly losses of CoAP messages at the application layer if many successive packets are lost. More substantially, it will also waste a lot of energy due to wasted transmissions. This is a different constraint from classic congestion control where losses are not harmful if they are recovered quickly. Here the damage of a packet loss is irreversible and should be avoided. The challenge is that in order to check if the bandwidth is available or not, the only way for the CoAP sender to do is to send a packet, but if this packet is sent while the bandwidth is not available then it will be lost. Retransmission of the lost packet does not reduce the incurred cost related to energy consumption. As a consequence, when a timeout expires indicating that the bandwidth is not available, determining the right time to wait before sending the retransmission is crucial. A short time may cause additional losses and a long time may reduce dramatically the goodput. It is clear that this time should relate to the available bandwidth.

Hence, the main idea is to remove entirely the backoff mechanism from CoAP and integrate a new mechanism that determines the adequate *spacing* between successive transmissions including retrans-

missions regardless of the RTO value or the retransmission counter. It is essential though to keep this algorithm as simple as possible. That is why our first proposal is based on the simple Additive Increase Multiplicative Decrease of the transmission rate even though we apply it to the time spacing between packets. We call the resulted protocol IDC-CoAP. The second proposal follows the measurement-based approach and it is inspired from the recent BBR congestion control [17] where the transmission rate is determined based on available bandwidth measurements and periodic probing of the bandwidth. We call this protocol version MBC-CoAP. Different from BDP-CoAP, MBC-CoAP adapts more adequately BBR to the existing properties of CoAP and avoids all BDP-CoAP design inaccuracies.

6.1.1 IDC-COAP: Increase/Decrease sending rate

In this version of the protocol, we aim at keeping the control algorithm as simple as possible by simply following the Additive Increase Multiplicative Decrease principle to control the rate with two differences. The first consists on working on the spacing between successive packets instead of the rate. The second consists on adding a phase of fast rate increase to benefit more from the available bandwidth in case it opens up. When the CoAP sender transmits a packet, there are two main network events from its point of view:

- An ACK is received. It means that the sending rate $\leq$ residual bandwidth. The current time *spacing* between packets can be decreased to increase the rate.
- The RTO expires. It means in case of congestion that the sending rate $>$ residual bandwidth. The current time *spacing* should be increased to decrease the rate.

Algorithm 5 IDC-CoAP pseudo-code

1: Wait for CoAP ACK or RTO expiration
2: **if** ack is true **then**
3: **if** spacing $\geq$ loss_spacing **then**
4: $spacing = spacing - dw * spacing$
5: **else**
6: $spacing = spacing - fw * spacing$
7: **end if**
8: **else**
9: $loss\_spacing = spacing$ /* Save congestion level */
10: $spacing = iw * spacing$
11: **end if**
12: $spacing = max(spacing, current\_time - last\_send\_time)$
13: Send next packet (transmission or retransmission) at: $last\_send\_time + spacing$

The pseudo code of the control algorithm is presented in Algorithm 5.

When a loss is detected, the sending rate should be decreased. Therefore, the spacing is increased by multiplying it by the incremental weight iw (line 10). When ACK is received, the spacing is reduced gradually using the decremental weight dw (lines 2 and 4). When spacing becomes lower than the spacing saved at the loss event (line 5) which corresponds to the last known available bandwidth, then spacing is reduced with a higher decremental factor fw in order to find quickly the new possible expanded available bandwidth (line 6). The maximum function is invoked to make sure that the sender can not send before the reception of the next ACK or the expiration of RTO (line 12). In the performance evaluation section, we show that a good combination of the algorithm parameters is: incremental weight $iw = 1.5$, decremental weight $dw = 0.01$ and fast decremental weight $fw = 0.5$. These parameters were chosen to achieve a good tradeoff between goodput and loss ratio. However, they can be easily tuned when the application requires better goodput on behalf of losses and energy consumption, or vice versa.

6.1.2 MBC-COAP: Measurement-Based sending rate

In this version of the protocol, we follow the measurement-based approach implemented in BBR to compute the $spacing$ between packet sending instants. We adopt the same concept of the max-filtered estimation of the available bandwidth and the same values for the probing and preventive pacing gains, i.e. 1.25 and 0.75 respectively. We also use the same length of the pacing cycle and the same update procedure. However, in order to overcome the shortcomings mentioned earlier (Section 2.7), we do not estimate neither the bandwidth delay product nor the minimum round trip propagation delay. We also do not maintain the packets in flight. These components are unnecessary for CoAP so removing them simplifies the protocol. Besides, the window used in the bandwidth estimation filter slides each time the CoAP sender receives an ACK. Thus, we compute the sending rate based on the maximum of the last m measurements, with m being the size of the sliding window. Which means, instead of using a time window, we use a space window. This modification is especially useful in high lossy environments. Importantly, in contrast to BDP-CoAP, we include each received ACK in the estimation of the bandwidth including those corresponding to retransmissions so that the number of measurement samples is sufficient to estimate more precisely the available bandwidth and converge to it rapidly. We simplified further the algorithm by removing all function calls as presented in the pseudo-code of Algorithm 6.

In the algorithm, we measure the spacing between successive received ACKs as an estimation of the "available spacing" which is inversely proportional to the available bandwidth. The last m spacing measurement samples are maintained to be used for computing the sending spacing (lines 2-7). After an ACK reception or an expiration of a timeout, MBC-CoAP sends the next packet according to the previously computed spacing (line 8). The next spacing for the next sending is computed by calculating first the next pacing gain (lines 9-20), then calculating the minimum of the last m spacing measurements

Algorithm 6 MBC-CoAP pseudo-code

1: Wait for CoAP ACK or RTO expiration
2: **if** ack is true **then**
3: $measurement\_sample = current\_time - last\_ACK\_time$
4: $last\_ACK\_time = current\_time$
5: Add $measurement\_sample$ to $Spacings[m]$
6: Remove oldest measurement sample from $Spacings[m]$
7: **end if**
8: Send next packet (transmission or retransmission) at:
$max(last\_send\_time + spacing, current\_time)$
9: $cycle\_index = (cycle\_index + 1)$ % 8
10: **if** $cycle\_index = 0$ **then**
11: **if** retransmission **then**
12: $pg = 0.75$
13: **else**
14: $pg = 1.25$ /* Probing phase */
15: **end if**
16: **else if** $cycle\_index = 1$ **then**
17: $pg = 0.75$ /* Preventive phase */
18: **else**
19: $pg = 1$
20: **end if**
21: $spacing = min(Spacings[m])/pg$

corresponding to the maximum of bandwidth samples (line 21). The pacing gain controls how fast our sending rate is. A pacing gain $= 1$ allows the cycle to cruise. On the other hand, a pacing gain > 1 increases the sending rate, while a pacing gain < 1 has the opposite effect. The probing state checks for higher bandwidth. Consequently, when the residual bandwidth increases, the design of MBC-CoAP allows to probe for more bandwidth. Similarly, when the residual bandwidth decreases, MBC-CoAP allows the preventive phase to reduce the sending rate accordingly. Therefore, our design principle using the cycle allows to send faster and slower in order to correctly estimate the bandwidth. Even though this algorithm is much simpler than the one in TCP BBR and BDP-CoAP, it is still more complex than IDC-CoAP due to the need of maintaining measurements.

6.2 Performance evaluation

In this section, we use our python simulator presented in Chapter 5 Section 5.2 to study and compare the performance of the full congestion control algorithms of IDC-CoAP and MBC-CoAP against other previous congestion control algorithms. In this aspect, we have chosen two challenging simulation sets where each set consists of different network scenarios. Then, we use Cooja/Contiki [48] to complement the comparison between rate-based and backoff-based approaches in a more realistic IoT environment. In this part, we define different network topologies with variable number of nodes. It is important to note we are not evaluating the congestion counteraction part only but the full algorithm. Hence, although we evaluated RTO estimation separately to show the originality of our work in chapter 5, we include it in the full version because, first RTO is considered as a detector for packet losses. Which means that when the first packet is lost, the sender should wait at least for the timeout to expire before taking any action. Second, when other packets are lost, the sender can not transmit before the expiry of RTO value even if the sending rate allows to transmit before, therefore, in this case, the RTO estimated value is considered as a floor for our sending rate. Indeed, each of the two full versions of our algorithms is now presented by its two parts: RTO estimation for congestion detection and congestion counteraction.

Firstly, we study the instantaneous behavior of the algorithms using network scenarios mentioned in Table 6.1. Secondly, the performance of the full congestion control of IDC-CoAP and MBC-CoAP are analyzed and compared with several previous algorithms using mainly the following performance metrics:

- Goodput or success rate: Total amount of successfully received data in a given time interval
- Loss ratio: Observed losses at the application level
- Overhead: Total amount of lost packets in the network over total amount of packets sent successfully

In particular, the Overhead performance metric measures the ability of the congestion control to send packets only when there is no congestion in the network to avoid losses, and also avoid wasting energy. In other words, Overhead computes how much effort or energy is spent to send one packet successfully. This performance metric is very important especially for IoT devices supplied by batteries.

6.2.1 Python Simulation Results - Congested (Bad) period

Here, the performances of our IDC-CoAP and MBC-CoAP are analyzed and compared against pCoCoA, CoCoA+, 4-state, BDP-CoAP and BEB of the standard CoAP. Particularly, the objective is to explore the ability of the congestion control algorithms to adjust their sending rate to utilize the available network bandwidth.

Fig. 6.1 shows the behavior of the instantaneous sending rate achieved by the different congestion control algorithms in presence of a variable residual bandwidth. The red and green plots show respectively the sending and success rates of the algorithms, while the blue plot corresponds to the residual bandwidth available in the network during the simulation period. Simulation parameters for these figures and the next ones are all summarized in Table 6.1. Regarding the algorithms of previous work, we have used their default or advised parameters [53, 10, 11, 14, 5].

According to Figures 6.1a, 6.1c and 6.1e, pCoCoA, 4-state and CoCoA+ sending rates are very high compared to the residual bandwidth because they do not adjust the sending rate according to the available bandwidth but try to send at the maximum allowable rate as soon as it seems to be possible. In fact, when a packet is lost, the initial value of the retransmission timeout RTO_{init} is multiplied by the backoff factor and hence the sending rate is decreased but without a direct relationship with the residual bandwidth. Still, after one or several losses and timeout multiplications, one retransmission goes through the network successfully. When the ACK of this packet is received, a new CoAP packet is immediately sent resulting in a sending rate that moves back again to the maximum allowable rate of $1/RTT$, which will cause again another loss. The transmission rate after this loss is $1/RTO_{init}$ which will very likely cause also another loss since RTO_{init} is optimized and its value is close to RTT. Even when the retransmission counter is reached and the CoAP packet is dropped definitively, the algorithms do not change their congestion counteraction and start sending the next packet using RTO_{init}.

In Figures 6.1a and 6.1c, the success rates of pCoCoA and 4-state are able to approach sometimes the residual bandwidth when it increases but the sending rate continues to be much higher causing unnecessary retransmissions. In contrast, in Fig. 6.1e, the success rate of CoCoA+ is always far from the residual bandwidth despite the fact that its backoff mechanism is the same as pCoCoA. This is because CoCoA+ does not minimize the computation of RTO_{init} as pCoCoA, and hence the time required to retransmit lost packets is larger and the convergence to the residual bandwidth is slower. Once again, this shows the importance of minimizing RTO_{init}.

Parameter	Value	Description
r	4	Retransmission counter
α	$1/8$	First weight for RTO calculation
γ	$1/32$	Second weight for RTO calculation
K	4 or 7	Spurious weight for RTO calculation
Residual Bandwidth (packets/sec)	U(0.6, 1)	Instantaneous behavior simulations
	U(0, 0.2) . . . U(0.9, 1.1) . . . U(2.1, 2.3)	Low variability simulations
	U(0.9, 1.1) . . . U(0.5, 1.5) . . . U(0, 2)	High variability simulations
Bad Period (ms)	Exponential(5000)	Distribution and average duration
RTT (ms)	N(500, 10)	Round Trip Time
MBC-CoAP specific parameters		
m	3 or 10	Number of measurements
pg	1.25, 0.75, 1, 1, 1, 1, 1, 1	Cycle pacing gains
IDC-CoAP specific parameters		
iw	1.1, 1.5	Incremental weight
dw	0.01	Decremental weight
fw	0.5	Fast decremental weight

Table 6.1: Simulation parameters used in evaluating congestion control algorithms for CoAP

We notice also that in Fig. 6.1c, the success rate of 4-state is better than pCoCoA because 4-state uses more tuned backoff factors that allow to retransmit more quickly which can be seen from the oscillations of the sending rate plot. Unfortunately, this goodput gain comes with the cost of increasing retransmissions.

As a first conclusion, these algorithms can be efficient if the bandwidth is available most of the time and/or losses occurs sparsely due to other reasons such as interference. However, if losses occur because many connections are using the same bottleneck link in the network, i.e. congestion, then the three backoff-based algorithms fail to adjust the sending rate adequately, justifying the need for a "real" congestion control for CoAP.

In IDC-CoAP, the available bandwidth is respected in the calculation of the sending rate to minimize losses during congestion periods (Fig. 6.1b). When the packets are lost, IDC-CoAP tends to increase the spacing between successive transmissions which will reduce the sending rate to converge back to

the available bandwidth and that is why the ratio of packet losses over successful packets is reduced as per Fig. 6.5e. When the available bandwidth expands, IDC-CoAP ends up increasing its sending rate after a reasonable amount of time which can be reduced further by tuning the spacing decremental factor. As a result, the sending rate is the same as the success rate most of the time leading to a small loss ratio. Besides, the success rate tends to be very close to the available bandwidth when the latter is somewhat stable.

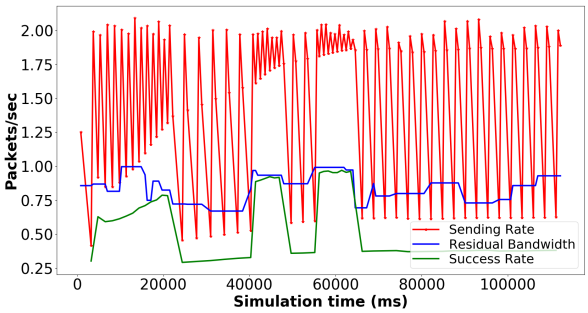
According to Fig. 6.1d, MBC-CoAP is also avoiding losses by trying to equate the sending rate with the success rate in order to stay below the residual bandwidth limit. We can see clearly the eight-phase cycle including the probing phase using the pacing gain of 1.25 that allows the sending rate to increase and thus converge slowly but surely to the available bandwidth offered to the CoAP sender. When the rate decreases suddenly, then MBC-CoAP takes some time to reduce its sending rate due to the cycle. However, this is compensated by a closer sending rate to the available bandwidth when the latter is somewhat stable.

The previous work rate-based BDP shown in Fig. 6.1f seems to perform similarly as backoff-based algorithms. This is because when the available bandwidth is lower than the initial estimated bandwidth which is the starting point of all simulated algorithms, successive losses prevent BDP from converging as we have explained in Chapter 2 - Section 2.7. Fig. 6.2 shows the same simulation for BDP when including retransmissions in bandwidth measurements. The modified BDP behaves now similarly to MBC-CoAP but in reality, the convergence to the available bandwidth is still much slower resulting to less losses but to a much lower goodput. This is because the bandwidth estimation of BDP includes in addition to the maximum of previous measurements, the minimum of these measurements. Fig. 6.3 shows the sending rate of BDP when we replace the min-max filter by a max-filter. This second modification approaches now the behavior and the performance of MBC-CoAP.

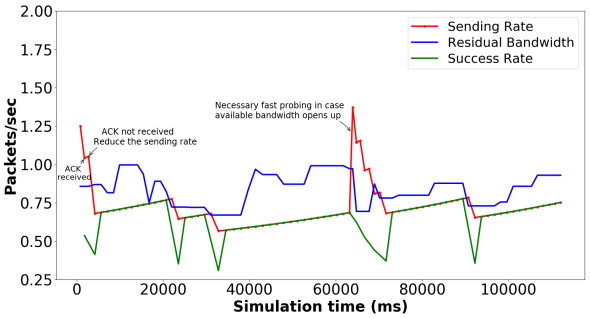
From these instantaneous figures, one can conclude that the rate-based approach if well designed is more appropriate than the backoff-based approach. This will be confirmed further with next results where averages of Goodput, Loss ratio and Overhead are computed and compared in several network scenarios.

Fig. 6.5a - Fig. 6.5e represent the average of the 3 performance metrics: Goodput, Application Loss ratio and Overhead per simulation. The simulation was run up to 5 hours and each simulation was repeated 3 times. The residual bandwidth has been varied between 0.1 and 2 packets per second.

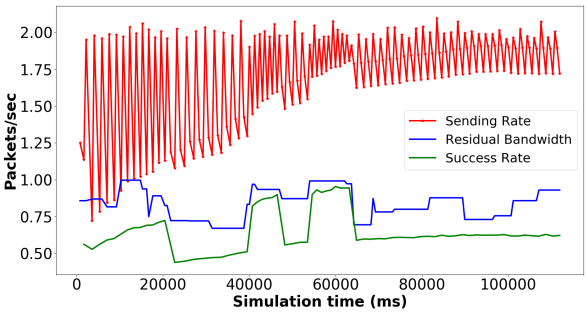
Fig. 6.5a, shows the average goodput of all algorithms while varying the average residual bandwidth in each set of simulations. The variance around the average is fixed to the same value since the residual bandwidth is varied uniformly between the average - 0.1 and the average + 0.1. For small residual bandwidth values, IDC-COAP and MBC-CoAP are slightly better than backoff-based algorithms since there is no enough bandwidth to send packets. When the available bandwidth increases, our rate-based



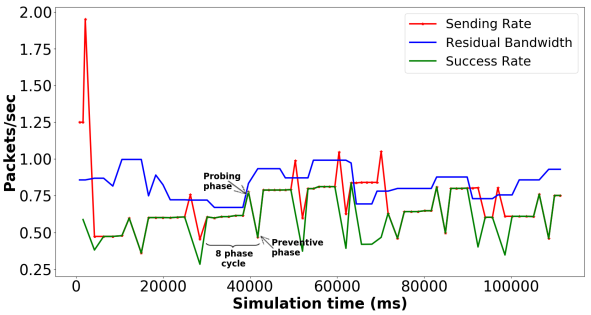
(a) pCoCoA



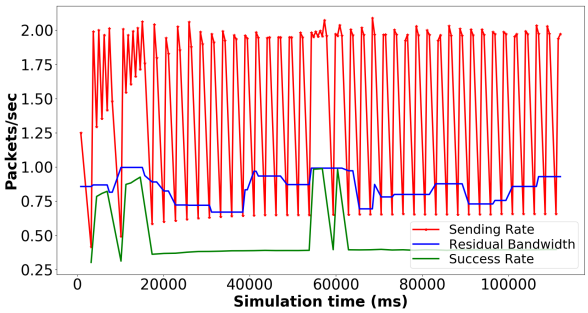
(b) IDC-CoAP



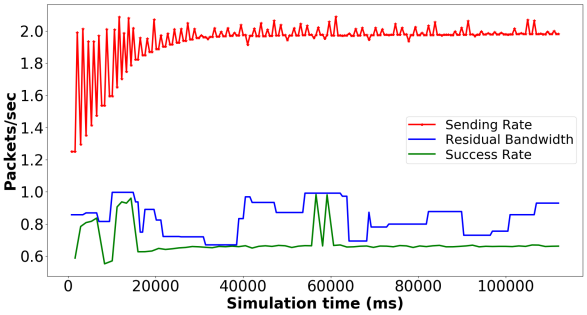
(c) 4-state



(d) MBC-CoAP



(e) CoCoA+



(f) BDP-CoAP

Figure 6.1: Instantaneous behavior of backoff-based (left) and rate-based (right) congestion control algorithms

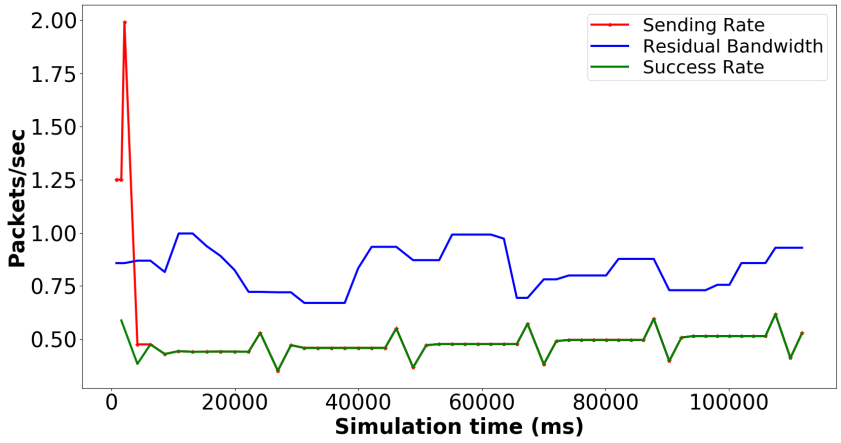


Figure 6.2: Correcting BDP-CoAP: Including bandwidth measurement samples from retransmissions

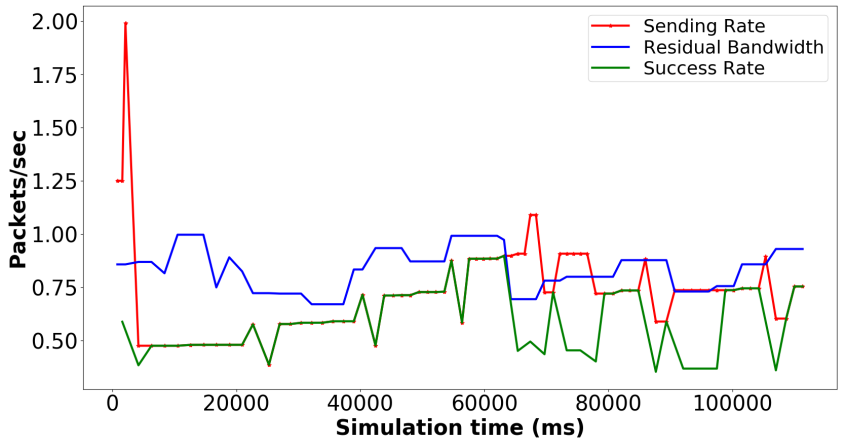


Figure 6.3: Correcting BDP-CoAP: Removing the samples minimum from the bandwidth estimation while including bandwidth measurement samples from retransmissions

algorithms IDC-CoAP and MBC-CoAP show a linear behavior, however a step-wise behavior with the presence of a large plateau is shown by all backoff-based algorithms. This is caused by the non fine-grained control performed by the fixed backoff factors that impose few possible values of the retransmission timeouts.

In the plateau, the goodput does not increase with the increase of the residual bandwidth. The value of the plateau corresponds approximately to

$$\frac{1}{RTO_{init}average + RTTaverage}$$

corresponding to a lost transmission followed by a successful retransmission. Thus, for the default CoAP the plateau is at $\frac{1}{2\frac{1+1.5}{2}+0.5} = \frac{1}{2\times 1.25+0.5} = 0.33$. Recall that 1.5 is the randomization factor

of the retransmission timeout. For other backoff-based algorithms that attempt to minimize RTO_{init}, the plateau is approximately at $\frac{1}{0.5\times1.25+0.5} = 0.88$. Another smaller plateau appears for backoff-based algorithms around $\frac{1}{0.5+(0.5+2*0.5)\times1.25} = 0.42$. In general, the plateau values correspond to

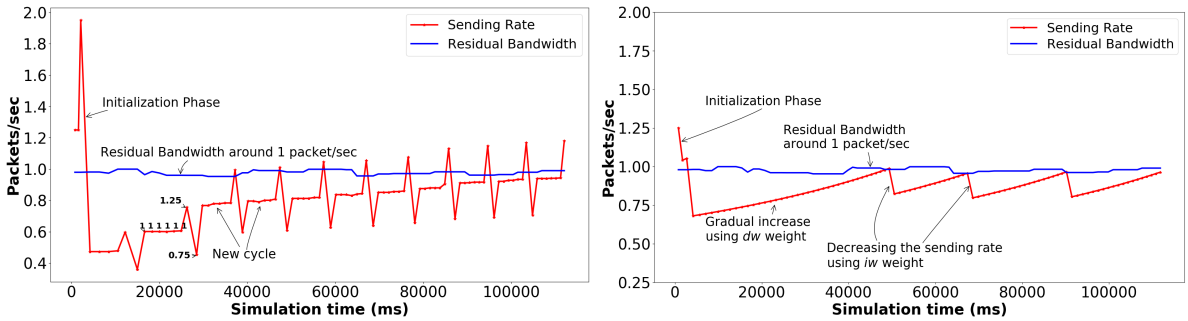
$$\frac{1}{RTTaverage + \sum_{i=0}^{j} b^i \times (RTO_{init}average)},$$

$j = 0, 1, 2, \cdots$. The variable backoff factors used by the algorithms are not sufficient to perform a fine-grained control. Indeed, according to pCoCoA and CoCoA+, the chosen backoff factor for the given RTT is 2.5, thus the plateau is more precisely at value $\frac{1}{0.5+(0.5+\mathbf{2.5}*0.5)\times1.25} = 0.37$. 4-state uses 1.7 half of the time and 2.5 most of the other half during the simulation which allow avoiding a clear first plateau but not the second large one.

On the contrary, the linear behavior of rate-based IDC and MBC algorithms engenders an additional gain of the goodput. Ideally, the goodput will be equal to the average available bandwidth which is not achievable because the rate control algorithm is operating blindly without prior knowledge of the network status. If the residual bandwidth offered to the CoAP sender is not very variable, then the expression of the linear relationship between the goodput and the average residual bandwidth can be obtained through a steady state analysis. For MBC-CoAP, the goodput is computed by assuming that the gain cycling is operating close to the residual bandwidth and that bandwidth probing with the pacing gain 1.25 will bypass the available bandwidth as illustrated in Fig. 6.4a where the blue and red lines represent the residual bandwidth and the sending rate respectively. The residual bandwidth, varying around 1 packet per second, and the sending rate of our MBC-CoAP algorithm (Algorithm 6) are sketched over time (x-axis). As per the figure, the relation between the sending rate and the residual bandwidth can be observed as $Residual\ Bandwidth = 1.25 \times SendingRate$ when the residual bandwidth is almost stable. The goodput can be approximated by

$$\frac{7}{6 \times 1.25 + 1 + \frac{1.25}{0.75}} ResBW$$

For IDC-CoAP, Fig. 6.4b shows the sending rate behavior in red line and the residual bandwidth in green line. We vary the residual bandwidth to around 1 packet per second. The sending rate of our IDC-CoAP algorithm (Algorithm 5) is sketched over time (x-axis). Now, we can compute the spacing values and the number of transmissions in a period comprised between two successive losses, which means when the spacing is equal to $1/ResBW$. Denote by S_i the current value of the spacing, then we will have successively $S_0 = iw/ResBW, S_1 = (1 - dw)iw/ResBW, \cdots, S_n = (1 - dw)^n iw/ResBW = 1/ResBW$. Hence, the total number of transmitted CoAP packets during this period is equal to



(a) Behavior of MBC-CoAP with Residual Bandwidth around 1 packet / sec

(b) Behavior of IDC-CoAP with Residual Bandwidth around 1 packet / sec

Figure 6.4: MBC-CoAP and IDC-CoAP: Stable residual bandwidth

$$n = -\frac{\log(iw)}{\log(1-dw)}$$

The goodput is then approximated by

$$\frac{n}{\sum_{i=0}^{n}(1-dw)^i \frac{iw}{ResBW}} = \frac{dw}{iw}\frac{n}{1-(1-dw)^{n+1}}ResBW \tag{6.1}$$

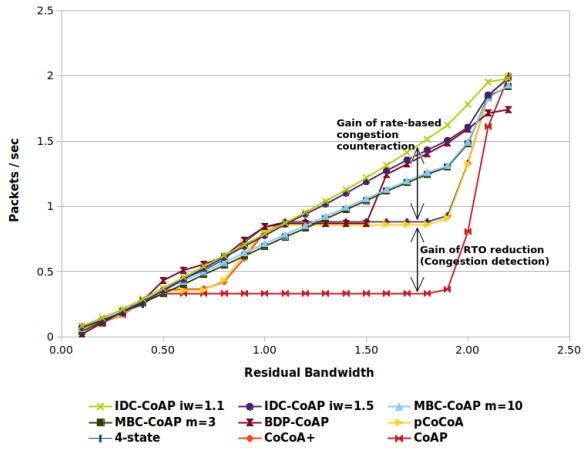
This formula can be used to tune the incremental and decremental weights of IDC-CoAP for a given performance objective. Indeed, an incremental weight $iw = 1.1$ provides a better goodput than $iw = 1.5$ and than MBC-CoAP when the bandwidth variability is limited.

The last observation is for BDP-CoAP which behaves almost like backoff-based algorithms when the available bandwidth is small. Then, when the bandwidth increases approaching the maximum which is 1 packet/RTT (corresponding to 2 packets/sec in our simulation) it provides similar behavior as rate-based algorithms IDC-COAP and MBC-CoAP in terms of goodput.

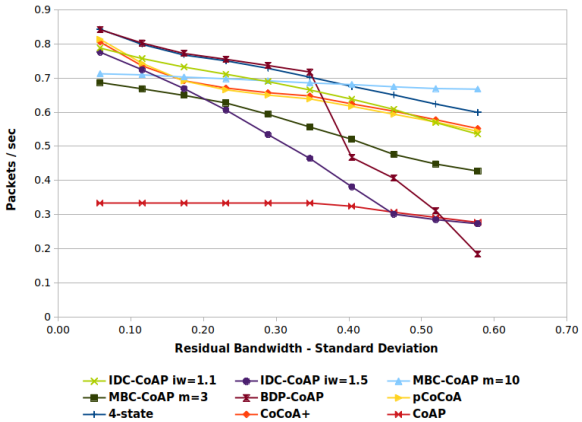
Observed Loss ratio at the application level is presented in Fig. 6.5c. When the residual bandwidth is increased, successive losses are reduced and hence application losses are reduced and even totally canceled because the re-transmission counter r is 4. However, when the residual bandwidth is small, we see clearly that CoAP losses are higher with the backoff-based algorithms which means that these algorithms experience more successive network losses than rate-based algorithms. These results are confirmed in Fig. 6.5e where the Overhead metric is shown. Usually, the algorithm that achieves a much higher goodput, experiences also a much higher Overhead. Nevertheless, all backoff-based algorithms show higher Overhead despite of having lower goodput as seen before. Thus, they consume more energy and reduce battery life. It is difficult though to achieve a good tradeoff between Overhead and goodput. Tuning the parameters of IDC/MBC-CoAP helps improving this tradeoff. For IDC-CoAP, increasing the incremental weight iw and/or decreasing the decremental weight dw, decreases

the goodput and reduces the Overhead. As a matter of fact, IDC-CoAP with $iw = 1.5$ achieves almost no overhead when the average residual bandwidth is greater than 0.3 packets/s. As for MBC-CoAP, decreasing the measurement window m, reduces the Overhead but does not affect the goodput because the residual bandwidth is not high variable, however, a decrease in goodput is observed when the variability of the residual bandwidth is high as we will see in next simulations.

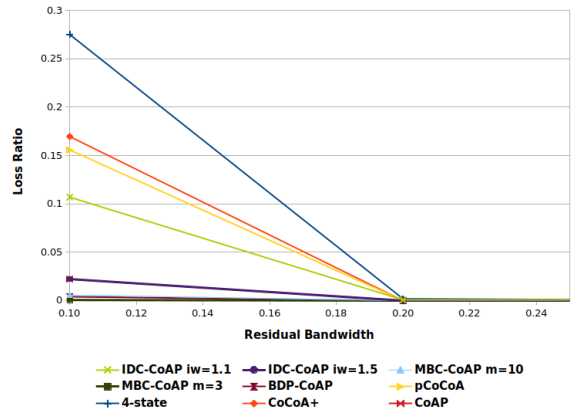
Finally, we test the robustness of the algorithms in front of a more dynamic network environment. In Fig. 6.5b, 6.5d and 6.5f, we fix the average residual bandwidth to 1 packets/s and we increase the standard deviation of the uniform distribution of the residual bandwidth by increasing the maximum and the minimum values from $[0.9, 1.1]$ to $[0, 2]$. Recall that 2 packets/s is the maximum possible rate which corresponds to $1/RTTaverage$. The residual bandwidth changes every 5 seconds. Here again, all backoff-based congestion control algorithms fail to reach an acceptable tradeoff between goodput and losses. Fig. 6.5f shows that the overhead of these algorithms is extremely high more than 100% and up to 140% indicating that any packet must be transmitted at least twice in order to be received successfully. The Overhead of IDC/MBC-CoAP is much lower while they still achieve a reasonable goodput in Fig. 6.5b even when the residual bandwidth is very variable. The overhead of IDC with $iw = 1.5$ is even around 10%. IDC with $iw = 1.5$ can be considered as a good tradeoff between goodput and Overhead especially when the residual bandwidth variability is medium. If the performance objective is a high goodput in a highly dynamic environment regardless of the Overhead and processing complexity, then MBC-CoAP with $m = 10$ is the choice because its goodput shows more stability thanks to the bandwidth estimation procedure and the gain cycling. The choice for selecting $m = 10$ is also confirmed in Fig. 6.7 where we have performed several simulations and we varied the measurement window size m with high variability in the residual bandwidth. Fig. 6.7a shows the impact of varying m on the goodput while we increase the variance (x-axis) and Fig. 6.7b presents the overhead. The more the network is variable, the more the residual bandwidth is unstable. In this case, we need to increase the window size to stabilize more the measurements in order to increase the goodput. However, the impact is always the opposite on the overhead. For instance, with $m = 14$, we achieve better goodput than other values of m variables but with more overhead. The main reason is that when the residual bandwidth variability increases which means the network is unstable, we need to compensate by stabilizing more our algorithm. Therefore, the design principle that should be applied is that when the network is unstable, the algorithm must provide stability and when the network is stable, the algorithm does not need to stabilize but in this case, it should converge fast. For such cases, and to provide fast convergence, m should be small. As a matter of fact, this is the compromise between stability and fast convergence. We must converge fast and at the same time, we should have a stable algorithm. Usually, it is very difficult to have a good compromise between these two factors because we do not know what is going on in the network. In this case, we prioritize stability. However, if we can ensure the stability



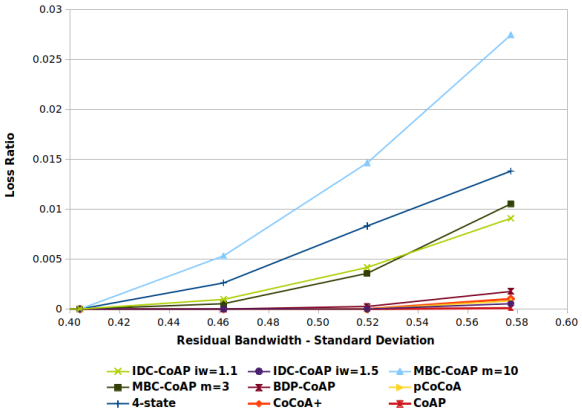
(a) Goodput - Low variability residual bandwidth



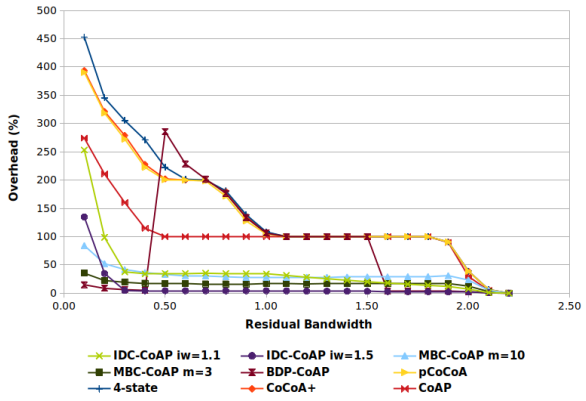
(b) Goodput - High variability residual bandwidth



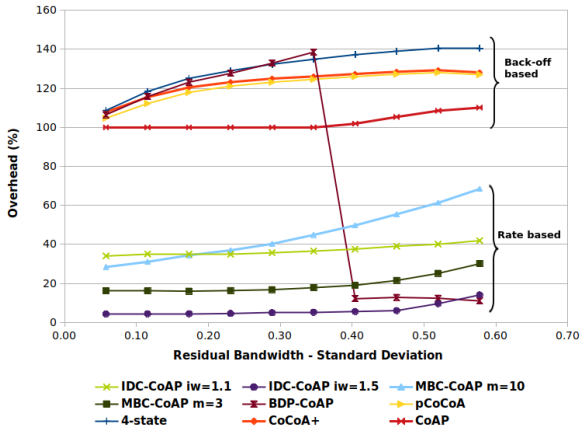
(c) Application Loss ratio - Low variability residual bandwidth



(d) Application Loss ratio - High variability residual bandwidth

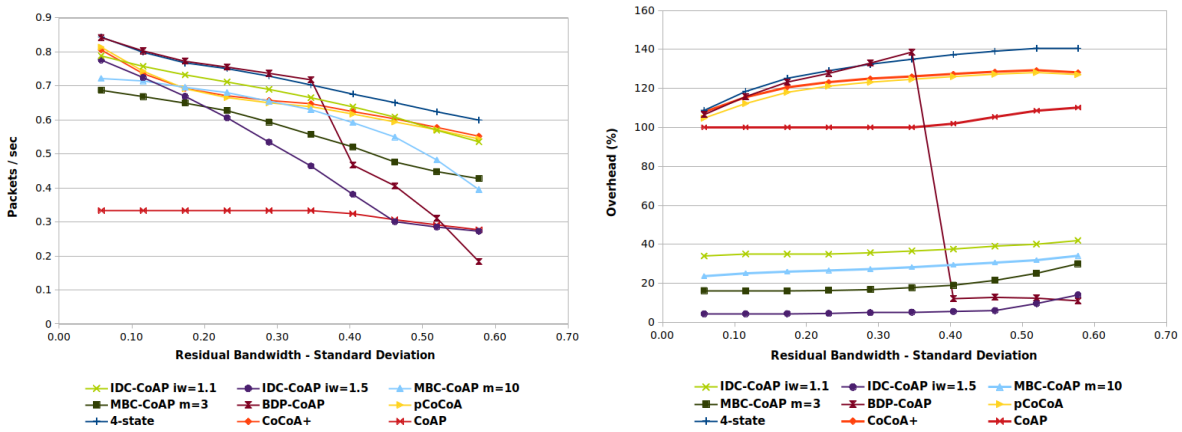


(e) Overhead - Low variability residual bandwidth



(f) Overhead - High variability residual bandwidth

Figure 6.5: Simulation results: Low and high variability residual bandwidth

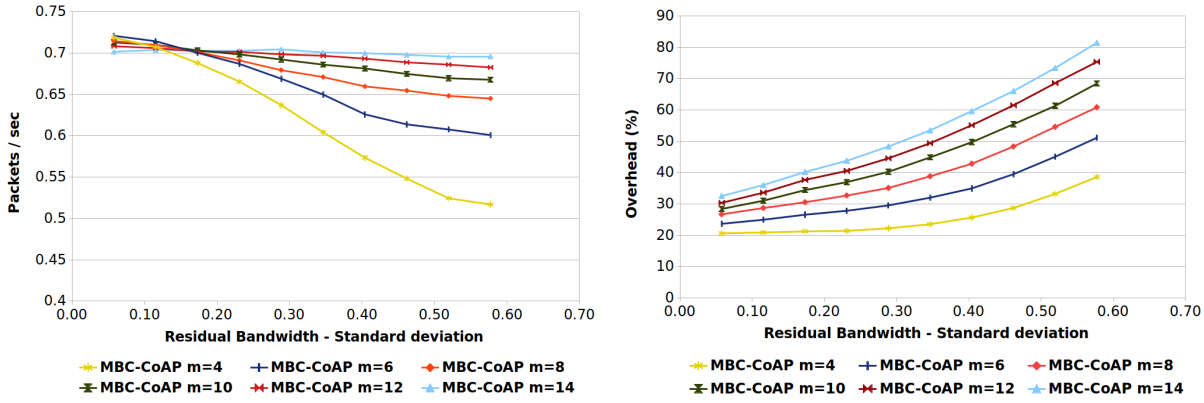


(a) Goodput **(b)** Overhead

Figure 6.6: Performance evaluation of MBC-CoAP using window of attempts

of a network, then we can tune the algorithm to converge fast without caring about stability which is already provided by the network. As a summary, the first observation is that when the bandwidth is not variable, m can be small (≤ 4) to allow fast convergence of the sending rate. The second observation is that when the bandwidth is variable and we need to maximize the goodput regardless of the overhead, then m must be set high (≥ 14). For our choice, and as can be observed from Fig. 6.7a and Fig. 6.7b, $m = 10$ maintains a good tradeoff between goodput and overhead when the bandwidth is high variable. On the other hand, we evaluated a different version of MBC-CoAP by replacing the sliding window of measurements with a sliding window of attempts, which means the window slides at every transmission attempt. Therefore, instead of considering the last m measurements that are computed when an ACK is received, the attempts where an ACK is not received are also included in the window. This version is tested in different network scenarios. We show here the results of a challenging case with high variability residual bandwidth whose network parameters are listed in Table 6.1. Although the overhead is reduced as shown in Fig. 6.6b, however, as per the results presented in Fig. 6.6a, when the standard deviation of the residual bandwidth increases, the goodput degrades. As a consequence, it is hard to attain an accepted tradeoff between goodput and overhead using this kind of sliding window.

Referring again to the results presented in Fig. 6.5b, the goodput of BDP-CoAP decreases dramatically with the increase of bandwidth variability. This is because the bandwidth is varying quickly between low and high values, thus the min operator used in the estimation of the residual bandwidth is not adequate at all. Besides, BDP uses the last ten attempts as measurement points instead of the last ten measurements, i.e. the measurement window slides at every transmission attempt. Unfortunately, this generates a similar behavior to a time window which is not adequate for CoAP because the number of sent packets per RTT is small especially when there are losses.



(a) Goodput as a function of bandwidth variation

(b) Overhead as a function of bandwidth variation

Figure 6.7: Simulation results: Varying measurement window size m

6.2.2 Python Simulation Results - Congested and non-congested periods (Good and bad)

In this section, we consider two network scenarios with different good and bad periods to challenge more the algorithms. In the first scenario set, the average duration of the good period is set to 1 minute and the average duration of the bad period is set to 5 minutes. While, in the second scenario set, the average duration of the good period is set to 5 minutes and the average duration of the bad period is set to 5 minutes. The average residual bandwidth in the good period is 1.8 packets/s and we vary the average residual bandwidth of the bad period from 0.1 packets per second to 1.8 packets/s for both scenarios. The performance of our IDC-CoAP and MBC-CoAP is analyzed and compared against different congestion control protocols. In particular, we show the results where averages of goodput, observed loss ratio and overhead are computed and compared in presence of these good and bad periods with different available bandwidth constraints. The simulation was run up to 5 hours and each simulation was repeated 3 times. Confidence intervals are not shown to not encumber the figures. Simulation network parameters for the next figures are summarized in Table 6.2. Regarding the other parameters of the evaluated protocols, we have used the same ones presented in Section 6.2.1. Fig. 6.8a - Fig. 6.8b show these 3 performance metrics.

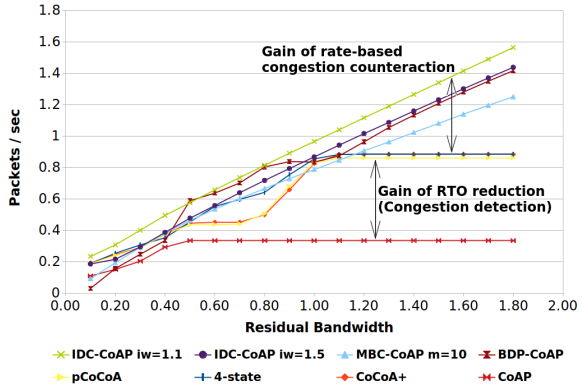
Parameter	**Value**	**Description**
Residual Bandwidth (packets/sec)	U(1.7, 1.9)	Bandwidth in good period
	U(0, 0.2) . . . U(0.9, 1.1) . . . U(2.1, 2.3)	Bandwidth in bad period
Good Period (mn) Bad Period (mn)	Exp(1), Exp(5) Exp(5)	Distribution and average duration
RTT (ms)	N(500, 10)	Round Trip Time

Table 6.2: Python simulation network parameters

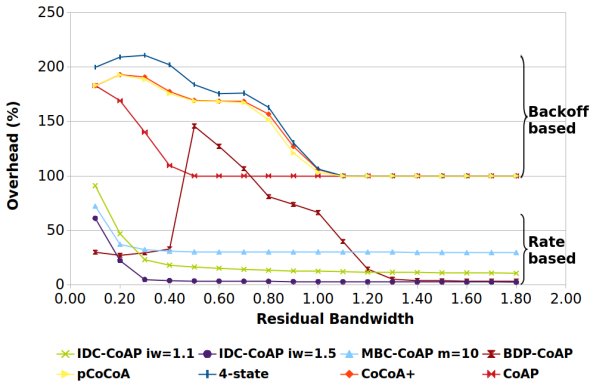
From Figure 6.8a we can first observe that for a residual bandwidth of the bad period less than 0.5 packets per second, IDC-COAP (iw=1.5 and iw=1.1) and MBC-CoAP (m=10) achieves somewhat better goodput than backoff-based algorithms. This is expected since there is no sufficient bandwidth to send packets. In order to get a better insight of the overall performance, we consider an additional metrics. In particular, Figure 6.8b shows the overhead generated by the algorithms for the same residual bandwidth intervals. For a residual bandwidth less than 0.5 packets per second, IDC-CoAP and MBC-CoAP outperforms CoCoA+, pCoCoA and 4-state by 175% reduction in the overhead. This is a massive difference and has a huge impact on energy consumption. Similar to what we have seen in the previous section, when the available bandwidth increases, the goodput of IDC-CoAP and MBC-CoAP behaves linearly while a step-wise behavior with the presence of a small plateau shown by the backoff-based algorithms (0.4 - 0.8 Residual Bandwidth on the x-axis), then a second larger plateau at the x-points (1 - 1.8 on the x-axis). The value of the plateau, detailed in Section 6.2.1, corresponds approximately to $\frac{1}{T+R}$ (corresponding to a lost transmission followed by a successful retransmission). Also, as shown in Figure 6.8a, BDP-CoAP achieves better goodput than other algorithms and is close to IDC-CoAP in some cases (0.5 till 0.9 packets/sec residual bandwidth), however, this improvement is on the behalf of the overhead and this can be seen in Fig. 6.8b where the overhead of BDP-CoAP suddenly increases till 150% (0.5 and 0.6 packets/sec residual bandwidth) and around 100% in other cases (at residual bandwidth 0.6 - 0.9 packets/sec). Therefore, this increase in goodput is on the cost of energy consumption due to the increased number of losses. In general, and according to the results presented in Fig. 6.8, we can still see the inability of backoff-based algorithm to avoid overhead and bandwidth wastage. As we have seen in this Section and Section 6.2.1, all backoff-based algorithms behave in a similar manner. It is important to note that we have also observed this phenomenon while assessing the algorithms in other network scenarios not mentioned here.

Figure 6.8d shows the observed Loss ratio at the application layer. When the residual bandwidth is increased, successive losses are reduced and hence application losses are reduced and even totally canceled because the re-transmission counter r is 4. On the other hand, when the residual bandwidth is small, we see that losses are higher for all. However, as per Fig., these losses vanish even for small residual bandwidth if we increase the re-transmission counter r. In Figure 6.8b and when the residual bandwidth is greater than 0.5 packets per second, the overhead of backoff-based algorithms is greater by around 100% than rate-based algorithms which means that the former algorithms experience more successive network losses.

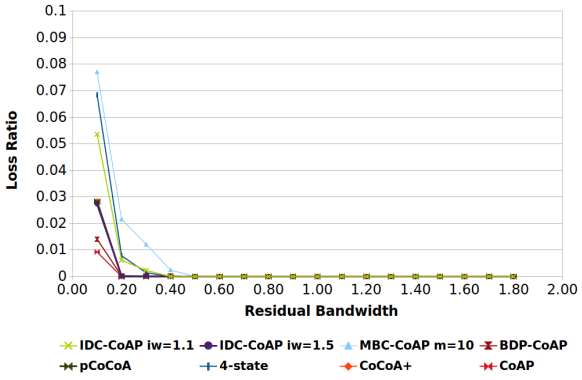
In the second scenario set, we report the results of the goodput and overhead metrics (Fig.6.9) while having longer good period duration set to 5 minutes and keeping the same average duration of the bad period (5 minutes). As per Fig. 6.9a, IDC-CoAP and MBC-CoAP perform always better than pCoCoA,



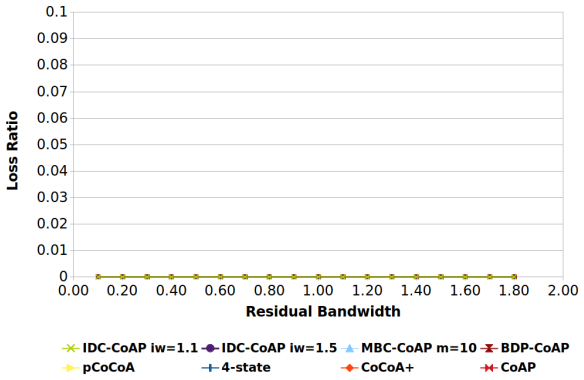
(a) Goodput



(b) Overhead

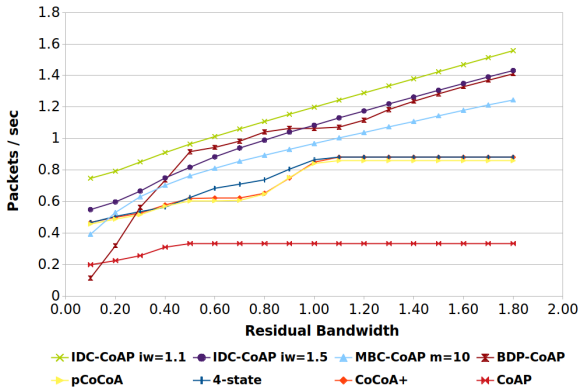


(c) Application Loss ratio: $r = 4$

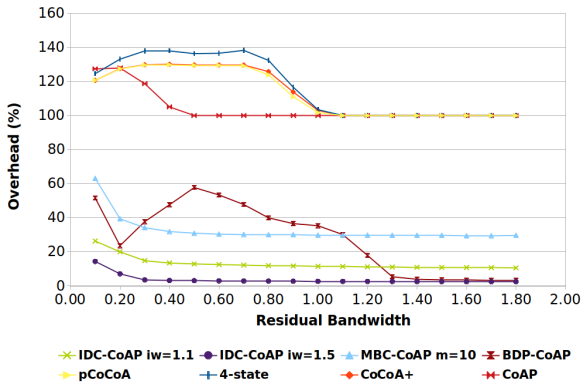


(d) Application Loss ratio: Increasing r

Figure 6.8: Simulation results while varying the residual bandwidth in the bad period



(a) Goodput



(b) Overhead

Figure 6.9: Goodput and overhead with long good and bad periods

4-state and CoCoA+ while varying the residual bandwidth of the bad period on the x-axis. The main difference from the first scenario is that all algorithms are now able to achieve better goodput due to the extended good period duration. For instance, when the residual bandwidth is 1 packet/sec in the bad period, IDC-CoAP ($iw = 1.1$) can successfully send 1.2 packets/sec. At the first glance, that looks strange because it is greater than the residual bandwidth, however, this is justified because the algorithm can now compensate the bad period by sending more during the good period where the residual bandwidth is up to 1.8 packets per second. This is observed as well for other values of the residual bandwidth. As shown Fig. 6.9a, BDP-CoAP guarantees a performance close to IDC-CoAP in some cases and better than MBC-CoAP in most of the cases. Here, the variability is not high and hence the min-max filter and the window of attempts imposed by IDC-CoAP in this special case have a better impact on the goodput than using a max filter only. However, as mentioned earlier, this is achieved on the behalf of overhead which is reflected in Fig. 6.9b. In the same figure, all backoff-based algorithms show high overhead equal or even greater than 100%. Here again, the backoff algorithms fail to achieve a good compromise between the performance metrics. Finally, it is worth noticing that we have also performed same set of simulations with lower RTT values down to 50ms and higher values of the residual bandwidth up to 20 packets/s and we observed similar results including the step-wise behavior of backoff-based algorithms and the satisfactory performance of IDC-CoAP and MBC-CoAP in both low and high network bandwidth variability.

6.2.3 Implementation in Contiki OS and Cooja Simulations

In order to validate our study in a realistic environment, we have implemented IDC-CoAP in the Contiki Operating System [21]. Contiki OS is used for IoT devices and especially tiny ones such as the TelosB/SkyMote family and Zolertia Z1 mote [2]. Then, we use the real hardware emulator MSPSim [1, 4] to load Contiki OS on it, and we use Cooja simulator [48] to create several network scenarios composed of motes playing the role of a CoAP receiver and CoAP senders.

One of MSPSIM and Cooja features is the ability of emulating constrained real devices while reflecting their hardware specifications and processing capacities. The motes implement IEEE 802.15.4 at the physical and MAC layers. Distinguished from previous works, the radio duty cycle (RDC) feature of the MAC layer is kept enabled in all our experiments. Two types of motes are used for CoAP senders and receivers which are Z1 and wismote. Table 6.3 shows the hardware specifications of these motes and the settings in the Contiki OS loaded in the motes. The RPL router uses the more constrained sky mote since it implements neither transport nor application layers.

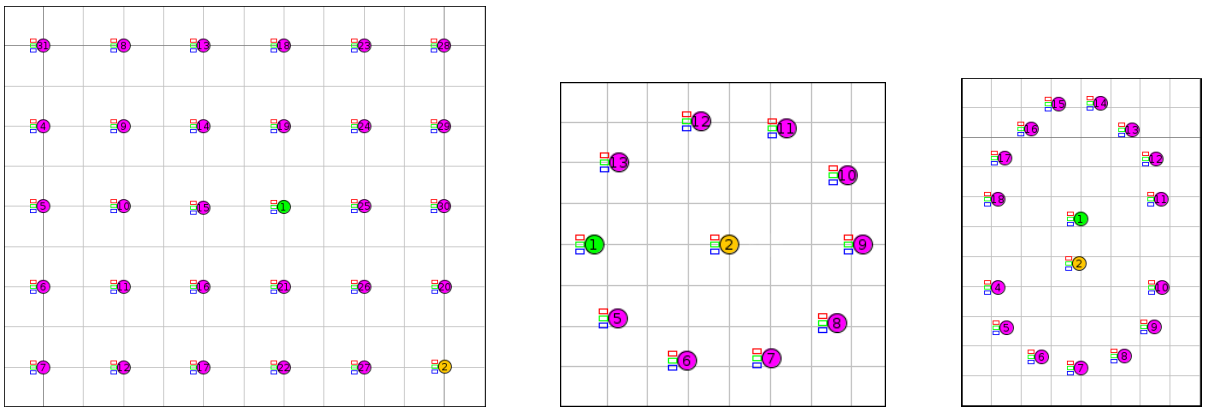
Six network topologies with different number of nodes are defined for the performance analysis. These topologies are: a grid of 30 nodes, a U-shape with 15 nodes, a Square-shape with 18 nodes, a ring and a chain with 12 nodes, and a dumbbell with 7 or 10 or 15 nodes. The Square-shape and the

Zolertia Z1 mote specifications	
RAM	8KB
ROM	92KB
Micro-Controller	MSP430F2617
CPU Clock speed	16MHz
RF standard	CC2420 2.4GHz / 250Kbps data rate
Wismote mote specifications	
RAM	16KB
ROM	256KB
Micro-Controller	MSP430F5
CPU Clock speed	25MHz
RF standard	CC2520 2.4GHz / 250Kbps data rate
Simulation Parameters	
Physical protocol	IEEE 802.15.4
RDC	On (Contikimac driver)
MAC	CSMA driver
Transmission (TX) ratio	90% or 95% or 100%
Routing protocol	RPL
Network protocol	6LoWPAN/IPv6
UIP buffer size	256
CoAP frame size	80 bytes

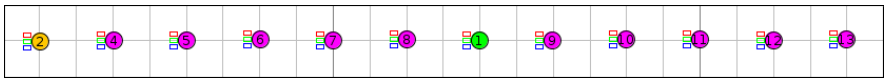
Table 6.3: Cooja/Conitki parameters and hardware specifications of Z1 and Wismote motes

U-shape are obtained by shutting down some nodes in the grid topology. In the first one, only border nodes communicate with the CoAP receiver and in the second one only the border nodes forming a U-shape communicate with the receiver. Fig. 6.10 illustrates these 6 topologies with 1 RPL border router (green color), CoAP receiver (yellow color) and CoAP senders (pink color). The distance between the unit squares is 10m. Choosing various network topologies determines how many direct neighbors each node has. Also, it determines how many nodes compete for the radio channel and the available bandwidth. These topologies create a diversity of available links, bandwidth and number of CoAP connections in the network.

Destination Oriented Directed Acyclic Graph (DODAG) is initiated by the RPL border router which stores the routing information for all the nodes. The RPL border router serves as a relay for CoAP messages, it does not send or receive any CoAP message. An initialization phase around 100 seconds for each simulation is allowed since the RPL border router needs an amount of time to build the DODAG across the network. No results are collected during this phase. Once the network is initialized, CoAP



(a) Grid Topology **(b)** Ring Topology **(c)** Dumbbell Topology



(d) Chain Topology

Figure 6.10: Network topologies for CoAP congestion control performance evaluation with Cooja/Contiki OS environment

senders generate messages which are directed towards the CoAP receiver. NSTART is set to 1 as per CoAP default specification. The simulations of the different scenarios have a 15 min duration. These simulations are repeated 5 times for each scenario. On one hand, as shown in the previous sections, IDC-CoAP with $iw = 1.5$ is a good tradeoff between goodput, Overhead and code simplicity then we can choose it as a representative of rate-based congestion control algorithms in this study with Contiki/Cooja. On the other hand, we have obtained the last version of the implementation of CoCoA+ from its authors, which will serve as a representative of backoff-based approach for congestion control. In the previous section we showed that the performance of CoCoA+ is indeed very close to pCoCoA and 4-state. Besides, we will compare with the existing CoAP implementation in Contiki that follows the current standard using the simple binary exponential backoff [53].

Fig. 6.11 shows the three measured performance metrics: Goodput, Loss ratio and Overhead for the three protocols: IDC-CoAP, CoCoA+ and CoAP for each of the four network topologies: Ring, Chain, Dumbbell and Grid. As a first observation, it is clear that IDC-CoAP is always better than CoCoA+, and that CoAP is evidently not efficient enough in all cases.

More closely, Fig. 6.11a shows that the Goodput of IDC-CoAP is higher than CoCoA+ even when the transmission ratio TX is 100%, i.e. no loss simulation by Cooja. In the ring topology, most of the motes send directly to the CoAP receiver and thus the network is somewhat stable with less congestion events. To challenge more the algorithms, we decrease the transmission ratio to 95% and 90%. Still CoCoA+ and CoAP achieve lower Goodput than IDC-CoAP. They also experience more application

losses (Fig. 6.11b) and overhead (Fig. 6.11c). Even when transmission losses are high (TX=90%), IDC-CoAP is able to reduce the Overhead compared to other algorithms as shown in Fig. 6.11c. This is because transmission losses are perceived as a residual bandwidth reduction from the sender and thus IDC-CoAP which applies the rate-based congestion control reacts better to these losses.

In the chain topology, we varied the number of nodes in the chain to study the impact on the Goodput, Loss ratio and Overhead. Naturally, the overall performance degrades when the number of nodes in the chain increases because the nodes near to the receiver becomes more congested. In Fig. 6.11d, IDC-CoAP algorithm results in a better goodput than CoCoA+ and CoAP in all number of nodes. It also achieves zero loss ratio and lower overhead when the number of nodes varies as per Fig. 6.11e and Fig. 6.11f. Hence, IDC-CoAP preformance is still robust when we have more congested nodes in the network.

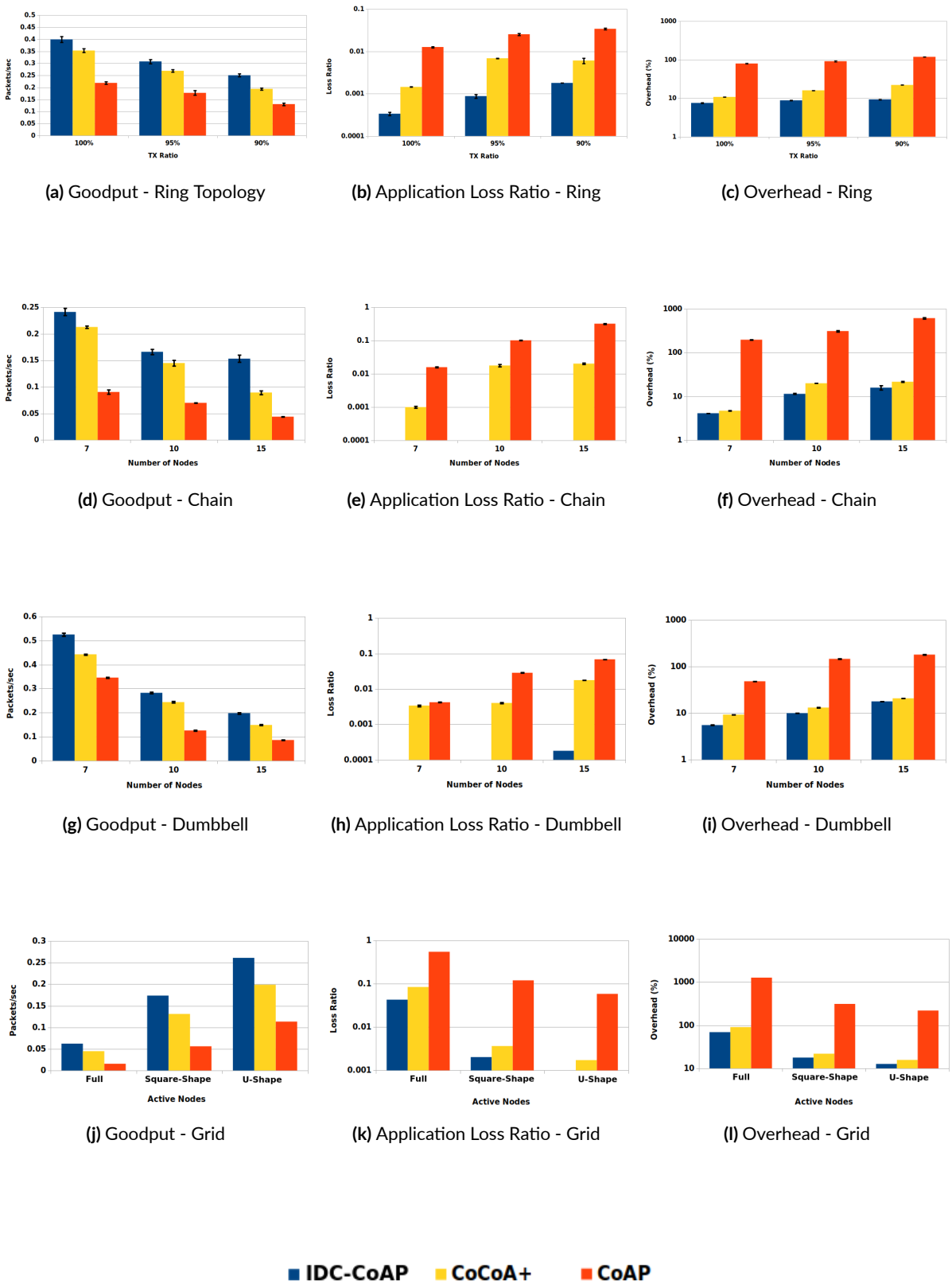
Thanks to our rate-based mechanism which tends to leverage from the available bandwidth and reduces the sending rate when the residual bandwidth is not sufficient. More nodes enforce more traffic in the network which leads to more congestion, therefore, in such conditions, IDC-CoAP tends to minimize packet losses and save the battery life of constrained devices.

Similar to chain and ring topologies, IDC-CoAP attains better performance results in dumbbell topology and this is illustrated in Fig. 6.11g to Fig. 6.11i. The creation of a congested link between the RPL router and the CoAP receiver does not impact the relative performance of IDC-CoAP compared to the others. Here also we varied the number of nodes with similar results as in the chain topology.

The results of the grid topology and its sub-topologies Square-shape and U-shape are shown in Fig. 6.11j to Fig. 6.11l. Here, the position of congestion in the network topology becomes more variable and the residual bandwidth as perceived by senders can be more dynamic, especially in the full grid topology. Even in this case, IDC-CoAP is still showing a higher goodput, a lower application loss ratio and also a lower overhead. This reinforces the necessity of rate-based congestion control for CoAP rather than current backoff-based ones.

6.3 Other Proposed Algorithms

In order to improve our algorithms further, we developed and evaluated other versions for the congestion control. For instance, we developed another version of IDC-CoAP with few more instructions (lines 3 and 4), an incremental weight $iw = 1.7$, decremental weight $dw = 0.01$ and fast decremental weight $fw = 0.5$. The detailed pseudo-code of this version is presented in Algorithm 7. As matter of fact, the results obtained by this version are similar to the ones obtained by the final version, however, to maintain simplicity, we reduced the extra instructions and kept the final version of IDC-CoAP already presented in Algorithm 5. Thanks to our formula (6.1) that helped us to find the optimal parameters and reduce instructions in IDC-CoAP final version.



(a) Goodput - Ring Topology **(b)** Application Loss Ratio - Ring **(c)** Overhead - Ring

(d) Goodput - Chain **(e)** Application Loss Ratio - Chain **(f)** Overhead - Chain

(g) Goodput - Dumbbell **(h)** Application Loss Ratio - Dumbbell **(i)** Overhead - Dumbbell

(j) Goodput - Grid **(k)** Application Loss Ratio - Grid **(l)** Overhead - Grid

Figure 6.11: Performance evaluation results of IDC-CoAP (rate-based) vs. CoCoA+ and CoAP (backoff-based) using Cooja/Contiki

Algorithm 7 IDC-CoAP pseudo-code with more instructions

1: Wait for CoAP ACK or RTO expiration
2: **if** ack is true **then**
3: **if** previous packet is lost **then**
4: $spacing = spacing - \frac{3*(spacing-loss\_spacing)}{4}$
5: **else**
6: **if** spacing $\geq$ loss_spacing **then**
7: $spacing = spacing - dw * spacing$
8: **else**
9: $spacing = spacing - fw * spacing$
10: **end if**
11: **end if**
12: **else**
13: $loss\_spacing = spacing$ /* Save congestion level */
14: $spacing = iw * spacing$
15: **end if**
16: $spacing = max(spacing, current\_time - last\_send\_time)$
17: Send next packet (transmission or retransmission) at: $last\_send\_time + spacing$

6.3.1 IDC-CoAP with probing

Moreover, we worked on an approach by following the Additive Increase Multiplicative Decrease principle to control the rate with periodic probing of the bandwidth. In order to optimize the goodput, we need to probe the bandwidth. Now, if we probe the bandwidth frequently and the congestion is high, then the overhead will be high too due to losses. From one side we need to reduce the overhead and on the other side, we need to optimize the goodput. We need to find a good compromise between these two contradictory ideas. In this approach and similar to MBC-CoAP, if there is no packet loss after probing, the sending rate is increased. Also, similar to IDC-CoAP, we reduce the spacing using a decremental weight in order to increase the sending rate, and we increase the spacing using an incremental weight to decrease the sending rate when a packet is lost. This approach, named IDC-CoAP with probing, merges the concept of both algorithms: IDC-CoAP and MBC-CoAP with some difference in the design concept. For instance, we probe with $loss\_spacing$ which is saved at the loss event. Also, when a packet is lost, we immediately reduce the sending rate unless we are in the probing phase. In this case, we converge to the original sending rate. The pseudo-code of this approach is presented in Algorithm 8.

Similar to IDC-CoAP, when a loss is detected, the spacing is increased by multiplying it by the incremental weight iw (line 18). When ACK is received, we implement ten-phase cycle including the probing phase using $loss\_spacing$ to increase the sending rate and thus converge slowly (line 5). The spacing is reduced slowly (line 8) during the cycle. To ensure that the $loss\_spacing$ does not exceed $\frac{1}{ResidualBw}$ in terms of rate, we use a security margin of 1.01 (lines 9, 16, 18). When no losses

Algorithm 8 IDC-CoAP with probing

Wait for CoAP ACK or RTO expiration
if ack is true **then**
  if $cycle\_index = 10$ **then**
    $cycle\_index+ = 1$
    $spacing\_before\_probing = spacing$ /* Save spacing before probing */
    $spacing = loss\_spacing$ /* Probing phase - Spacing saved at the loss event */
  else if $cycle\_index < 10$ **then**
    $cycle\_index+ = 1$
    $spacing = spacing - \frac{3*(spacing-1.01*loss\_spacing)}{4}$ /* Slow decrease */
  else
    /* No losses are observed and Cycle index = 11*/
    $spacing = dw * spacing$ /* Fast decrease to increase the sending rate*/
  end if
  if previous packet is lost **then**
    if $cycle\_index \leq 10$ **then**
      $spacing = spacing - \frac{3*(spacing-1.01*loss\_spacing)}{4}$
    else
      $spacing = spacing\_before\_probing - \frac{3*(spacing\_before\_probing-1.01*loss\_spacing)}{4}$
    end if
    $cycle\_index = 1$
  end if
else
  if $cycle\_index \leq 10$ **then**
    $spacing = iw * spacing$
  else
    $spacing = spacing\_before\_probing$
  end if
end if
$spacing = max(spacing, current\_time - last\_send\_time)$
Send next packet (transmission or retransmission) at: $last\_send\_time + spacing$

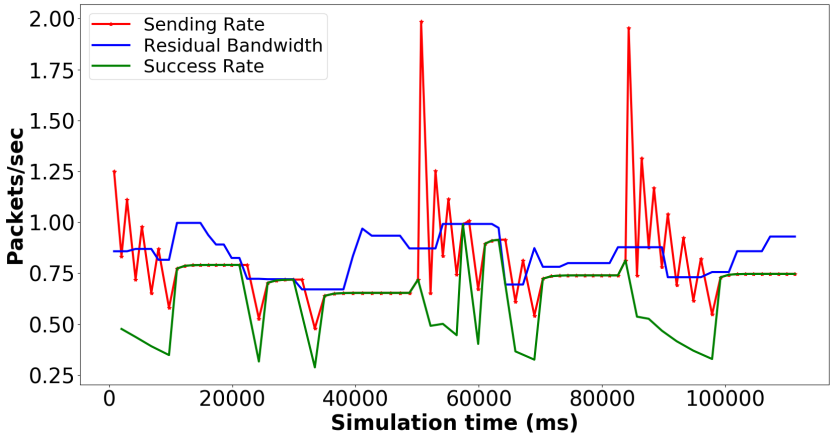
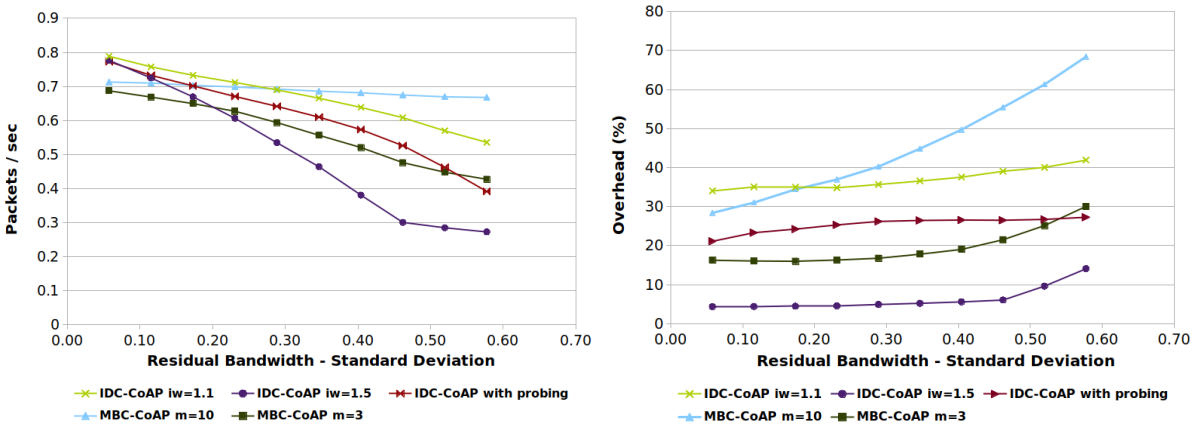


Figure 6.12: Instantaneous behavior of other proposed algorithms: IDC with probing



(a) Goodput: High variability residual bandwidth

(b) Overhead: High variability residual bandwidth

Figure 6.13: Simulation results of IDC-CoAP with probing

are observed during the probing cycle, spacing is reduced with a decremental factor dw (line 11). The instantaneous sending rate behavior of IDC-CoAP with probing is shown in Fig. 6.12 for $dw = 0.25$. The value of the sending rate can be reduced by choosing greater dw value. The blue plot represents the residual bandwidth while the red and blue plots represent the sending and success rates of the algorithm. Simulation parameters are summarized in Table 6.1. As can been seen from the instantaneous behavior of this approach, when the bandwidth increases, the sending rate becomes higher than the success rates which has a negative impact on the goodput. As per Fig. 6.13a, the goodput decreases when the bandwidth becomes highly variable. Although this approach provides better goodput than IDC-CoAP with $iw = 1.5$, however, this is achieved on the behalf of increased overhead as per Fig. 6.13b and additional number of instructions and complexity (Algorithm 8).

In our opinion, the code can be simplified more and the parameters can be tuned to further improve the results and achieve a better tradeoff between performance metrics.

6.3.2 IDC-CoAP: Exponential growth

Of equivalent importance and noteworthy is that we also implemented and evaluated the principle of IDC-CoAP with exponential growth. In this version of the protocol, we try to replace the additive increase principle of the proposed algorithm IDC-CoAP with a faster increase principle. According to our observation, when the sending rate converges slowly, some bandwidth will be wasted. However, if we converge faster, we will reduce the wastage of the bandwidth. As we have seen in IDC-CoAP, there was a wastage of the bandwidth. For instance, when the residual bandwidth is 2 packets per second, IDC-CoAP can not achieve more than 1.6 packets per second which is almost 80% of the available bandwidth. Actually, this is a drawback in the additive increase approach. Hence, when there is a congestion, it takes time to converge and transmit quickly. The new idea is to converge immediately. Therefore, instead of waiting t time, we wait $\frac{t}{2}$ then $\frac{t}{4}$ and so on until the sending rate becomes very close to the residual bandwidth. This fast convergence can be done using a binary or exponential approach. In addition to using $\frac{1}{2}$ in order to increase and decrease the sending rate, we might also use $\frac{3}{4}$ instead as an increase and decrease exponential factor. However, according to our analysis, the $\frac{1}{2}$ factor provides better convergence which will be confirmed in the instantaneous behavior. The outcome is a new approach called IDC-CoAP exponential growth, and the exponential factor can be either $\frac{1}{2}$ or $\frac{3}{4}$. The pseudo-code of this approach is presented in Algorithm 9. When ACK is received, we maintain a counter which is relevant to the target overhead. For instance, if the number of packets inside the period is n, the overhead is $\frac{1}{n}$. It is like we lose a packet after n times. We fix the counter n according to the target overhead. Here, we use $n = 10$ (line 3). The security margin used is 1.01 (lines 8 and 15). The spacing is exponentially reduced (line 8) to increase the sending rate. Here, we show the instructions with $\frac{1}{2}$ exponential factors (lines 8 and 15). When no losses are observed after n transmissions, spacing is reduced with a 0.5 factor (line 11). The instantaneous sending rates of IDC-CoAP exponential growth with $\frac{1}{2}$ and $\frac{3}{4}$ factors are presented in Fig. 6.14a and Fig. 6.14b respectively. The network scenario for the instantaneous behavior is similar to the one simulated in Fig. 6.1. We can see clearly the exponential growth that allows the sending rate to increase and converge quickly after a packet loss when the simulation time is between 10000 ms and 40000 ms in Fig. 6.14a) and between 20000 ms and 40000 ms in Fig. 6.14b). Due to the exponential increase, the sending rate exceeds the residual bandwidth in many cases which will cause packet losses.

Although such approaches will reduce the wastage of the bandwidth, however, it is very risky in case the bandwidth is highly variable. The main reason is that this fast behavior does not provide the network with sufficient time after congestion and hence worsen the situation. In addition to making the congestion worse, another drawback of such approaches is that they impose more losses. Fig. 6.15a

shows that IDC-CoAP exponential growth achieves better goodput than IDC-CoAP with $iw = 1.5$ when the residual bandwidth is very variable. The goodput of IDC-CoAP with $iw = 1.5$ decreases because of the slow convergence when the variability of the residual bandwidth increases. However, and as shown in Fig. 6.15b, IDC-CoAP exponential growth imposes more overhead up to 20% and 30%. The increase in the overhead is due to the increased number of losses caused by the fast convergence of the exponential growth.

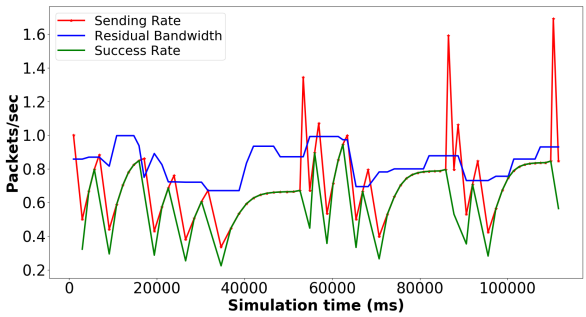
On the other hand, IDC-CoAP with probling and IDC-CoAP exponential growth are more complex than IDC-CoAP. However, a better tuning for these versions might improve the performance further. Hence, we consider these approaches as an open research problem.

Algorithm 9 IDC-CoAP with exponential growth

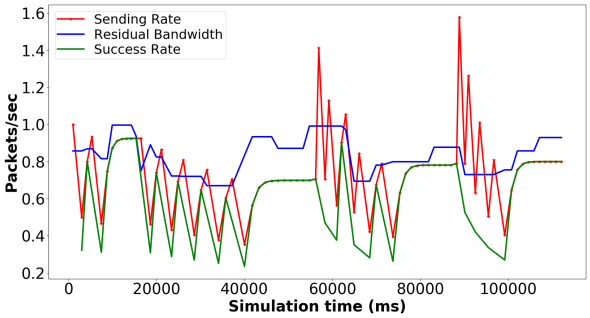
1: Wait for CoAP ACK or RTO expiration
2: **if** ack is true **then**
3: **if** $n = 10$ **then**
4: $n+ = 1$
5: $spacing = loss\_spacing$ /* Spacing saved at the loss event */
6: **else if** $n < 10$ **then**
7: $n+ = 1$
8: $spacing = spacing - \frac{(spacing - 1.01 * loss\_spacing)}{2}$ /* Exponential growth */
9: **else**
10: /* No losses are observed and n = 11*/
11: $spacing = 0.5 * spacing$ /* Increasing the sending rate*/
12: **end if**
13: **if** previous packet is lost **then**
14: $n = 1$
15: $spacing = spacing - \frac{(spacing - 1.01 * loss\_spacing)}{2}$
16: **end if**
17: **else**
18: $loss\_spacing = spacing$ /* Save congestion level */
19: $spacing = iw * spacing$
20: **end if**
21: $spacing = max(spacing, current\_time - last\_send\_time)$
22: Send next packet (transmission or retransmission) at: $last\_send\_time + spacing$

6.4 Summary

IoT devices are typically constrained by power capacity, memory and computational abilities in spite of the rapid changes in technology. As a result, these limitations increase the difficulty of data transmission and hence many algorithms may not perform efficiently if not well designed. Several research works have tried to improve further CoAP performance, in particular, its congestion control mecha-

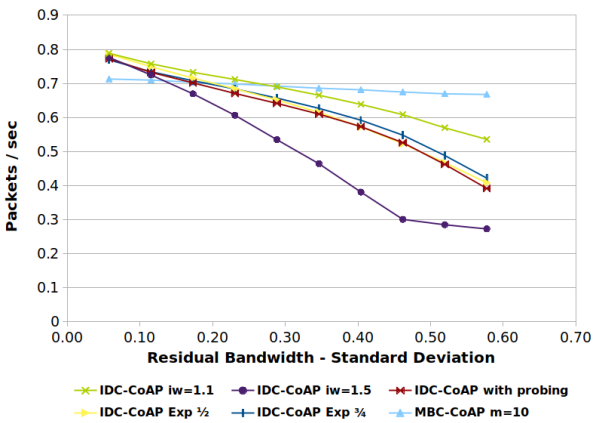


(a) IDC-CoAP: Exponential growth with $1/2$ factor

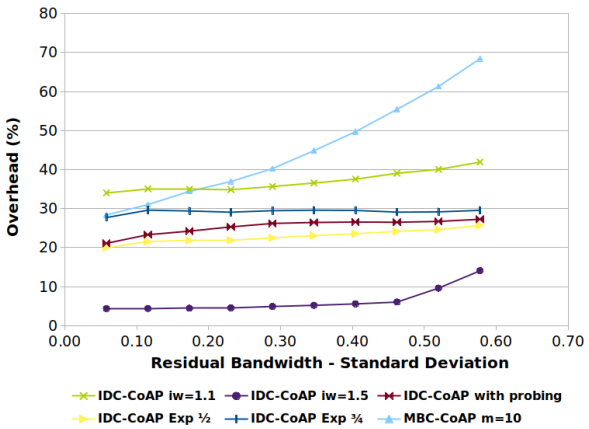


(b) IDC-CoAP: Exponential growth with $3/4$ factor

Figure 6.14: Instantaneous behavior of other proposed algorithms: IDC-CoAP exponential growth



(a) Goodput: High variability residual bandwidth



(b) Overhead: High variability residual bandwidth

Figure 6.15: Simulation results of IDC-CoAP exponential growth

nism which plays an important role in its efficiency in terms of reliability, energy consumption and rate performance. Based on our analysis in chapter 2 and according to the proposal of modifications of the evaluated algorithms, we developed, in this chapter, new rate-based algorithms, IDC-CoAP and MBC-CoAP. Most previous works have followed the backoff-based approach for congestion control while we proved in our performance evaluation section that a rate-based approach is much better than backoff-based approaches in most of the network topologies and scenarios. Our results, obtained from experiments, show significant advantages, with respect to goodput, loss ratio and overhead, in using rate-based approaches. IDC-CoAP provides lower overhead than MBC-CoAP and better goodput in scenarios where the residual bandwidth is not high variable, while MBC-CoAP is resilient against high variability of residual bandwidth and provides better goodput results than other algorithms. However, the linear approach of IDC-CoAP is less complex than MBC-CoAP and uses less instructions. In addition to the proposed algorithms, we have also explored and implemented different approaches that can be used as a guideline for further developments.

7

Conclusions and Future work

7.1 Conclusion

Since the design of the Constrained Application Protocol (CoAP), several research works have tried to improve its performance and increase its widespread usage in different IoT fields. In terms of efficiency, congestion control algorithms are particularly important in this protocol since they affect its reliability, energy consumption, and rate performance.

In this thesis, we developed novel models to represent the performance of CoAP under the Bernoulli loss model, the Simple Gilbert and Gilbert-Elliot loss models. We showed how to compute several performance metrics and validated these models via simulations in realistic network environments. The models are used to study precisely the performance of CoAP. The first observation that is brought up after this study is the importance of reducing the retransmission timeout in CoAP. Additionally, the performance analysis sheds light on the inadequacy of the backoff procedure in CoAP. In fact, it shows the necessity for replacing the default backoff procedure by a rate-based congestion control mechanism to improve bandwidth utilization, while maintaining simplicity.

Most previous works have followed the backoff-based approach for congestion control. However, we showed in this thesis that a rate-based approach is much more appropriate in most network scenarios. Indeed, the non fine-grained nature of the backoff-based procedure does not allow for precise control; this is true even when several backoff factors and weights are used. Thus, to be concretely efficient,

the proposed rate-based control must be both simple and well adapted to IoT networks and to devices that employ the CoAP protocol. Our two rate-based protocols IDC-CoAP and MBC-CoAP are able to leverage the available bandwidth in the network and thus reduce message losses and unnecessary retransmissions, which are very harmful to IoT constrained devices.

On the one hand, the design concept of MBC-CoAP allows to increase and reduce the sending rate to estimate the bandwidth accurately. However, this design concept imposes complexity when attempting to obtain and maintain the measurements. Also, another challenge is choosing the measurement window size m. When m is small, the sending rate converges fast and this is useful if the bandwidth is not variable. However, if the bandwidth is variable, then the value of m should be high to maximize the goodput. Now, how can one determine the variability of the bandwidth in the network? Equivalently, how should one set the measurement window size m? The answer to both questions is not clear because the status of the constrained networks is usually unpredictable.

On the other hand, IDC-CoAP design is simpler but the parameters should be well chosen to achieve a good tradeoff between goodput and loss ratio. Also, IDC-CoAP will be slow in convergence and the goodput is reduced when the decremental weights are small. Unfortunately, the risk of choosing high values of decremental weights is packet losses. In order to overcome this problem, we have developed a formula to generate the incremental and decremental weights of IDC-CoAP to meet a given performance objective. The parameters that we have selected achieved a good tradeoff between different performance metrics in the majority of the tested network scenarios. Our results, obtained from simulation and from the Cooja/Contiki environment, show that the more the available bandwidth or the network dynamics, the higher the gain in all performance metrics.

Another important aspect of our congestion control protocols is that, depending on the application, they can be tuned to optimize further the performance. We have shown in chapter 6 that a good combination of IDC-CoAP parameters is: incremental weight $iw = 1.5$, decremental weight $dw = 0.01$ and fast decremental weight $fw = 0.5$. For instance, in IoT health-monitoring applications, some information should be delivered with high reliability (such as the body temperature or the heart rate). In this case, the CoAP sending rate can be reduced conservatively to avoid losses. In particular, a higher incremental weight iw must be chosen. On the other hand, in a weather sensing application for example, a packet loss is less damaging because the next packet has an updated data of the weather. In this case, a lower value of incremental weight iw can be selected to increase the sending rate. Besides, if sensors are using wired power supplies, increasing the sending rate allows more data to be collected. Finally, since CoAP can be used also for file transfer, incorporating an effective rate-based control in it is beneficial.

7.2 Future Work

7.2.1 CoAP Evaluation

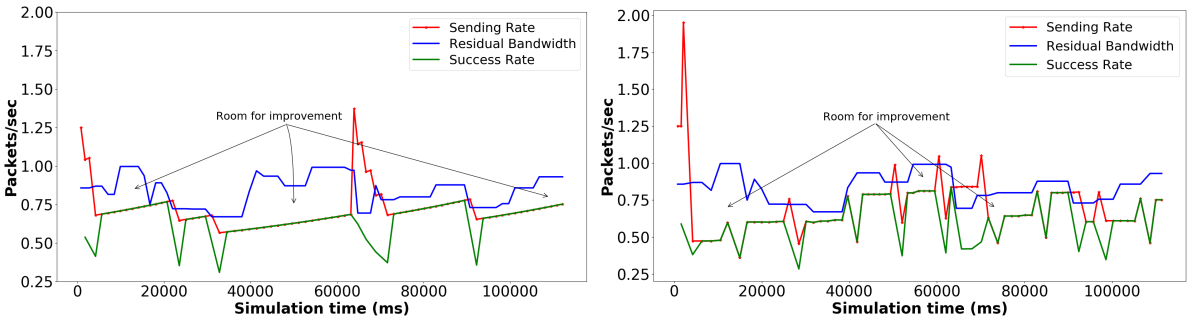
An important future work would be developing mathematical models, similar to the ones presented in this thesis, for IDC-CoAP and MBC-CoAP to complete their study from an analytical perspective. In this thesis, the developed models for CoAP evaluation assume that the bandwidth is one packet per RTT, the acknowledgement (ACK) of each packet is either received by the sender or not, and a backoff mechanism is involved when the ACK is not received. However, IDC-CoAP and MBC-CoAP assume that the bandwidth is variable and the sending rate is adapted according to the available bandwidth. Therefore, future work should take into account this behavior of MBC-CoAP and IDC-CoAP, which is different from the backoff-based approach. Besides, it would be interesting to include in the model other factors such as the impact of fragmentation. This may be done by modifying the loss probabilities of the Gilbert-Elliott model, which in turn can be obtained through measurements. Furthermore, other states can be added to our Markov chain to consider different packet sizes that would experience different loss ratios.

Another important future work is to compare our new CoAP protocols with other protocols that use Transmission Control Protocol (TCP) as a transport protocol, such as the Message Queue Telemetry Transport (MQTT). MQTT is an important application layer protocol designed for communications in IoT networks. It uses a topic-based publish/subscribe architecture, where reliability of messages is provided by three levels of Quality of Service (QoS). MQTT relies on TCP, and its reliance on this type of connection increases the overhead and thus the total packet size. Therefore, while MQTT can offer high reliability, it can lead to a harmful increase in power consumption [47]. Previous works have shown that it is appropriate to use MQTT in some specific IoT applications. Hence, performing an in-depth and relative analysis between our proposed CoAP protocols and protocols like MQTT is necessary to gain insight into their strengths and limitations, according to the application at hand.

7.2.2 Congestion Control mechanism

The results of this thesis prove that there is no congestion control mechanism that is optimal in all performance metrics and for all network scenarios. Our new proposals, IDC-CoAP and MBC-CoAP, perform better than other algorithms suggested by the literature in the majority of the considered cases.

There is no doubt that finding solutions that lead to better performance with less complexity is a difficult task. Indeed, one must take into account several aspects, such as (i) the capacity of the constrained devices, (ii) the traffic load on each node in the IoT network, and (iii) the ability to strike a balance between latency and reliability. In this thesis, we have presented different approaches to solve this problem. The first goal of our work is to keep the algorithms simple while achieving better perfor-



(a) Instantaneous behavior of IDC-CoAP algorithm

(b) Instantaneous behavior of MBC-CoAP algorithm

Figure 7.1: MBC-CoAP and IDC-CoAP: Future improvements

mance. In our future work, we aim to improve our proposed congestion control mechanism to provide even higher performance while maintaining simplicity. In particular, our objective is to estimate the bandwidth more precisely in order to improve goodput and reduce losses.

Fig. 7.1 presents the behavior of the instantaneous sending rate achieved by IDC-CoAP and MBC-CoAP algorithms in presence of a variable residual bandwidth. These graphs were introduced in details in Chapter 6. As can be seen from figures 7.1a and 7.1b, there is still room for improvement. Particularly, in Fig. 7.1a, we highlight points where the algorithm can better leverage the available bandwidth: at x-point 50000, the sending rate converges slowly although the bandwidth opens up. We see similar behavior at x-points 15000 and 120000. This takes place due to the gradual reduction in spacing via the decremental weight. In these instances, the available bandwidth is wasted and thus the goodput is reduced. Hence, one possible limitation of IDC-CoAP is that after 2 or more successive packet losses, its sending rate does not grow fast. Also, in Fig. 7.1b, the pacing cycle can be manipulated to engender additional gain of the goodput while minimizing packet losses, to achieve battery-life saving in constrained devices.

Since our results show that MBC-CoAP has robust stability and convergence properties, another future work aims to further reduce its complexity so that it can be incorporated in tiny IoT devices. Other approaches for congestion control may be adopted, but the addition of instructions will increase the overall complexity. While this problem has been explored in this thesis, better solutions can be sought after. On the other hand, we already started implementing MBC-CoAP in Cooja/Contiki. In fact, integrating MBC-CoAP in constrained IoT devices is one of the recent challenges we encountered. The initial results show that MBC-CoAP performs better than other algorithms on some nodes in our network topology but has negative impact on the performance of other nodes. Our first observation regarding this negative impact is due to MBC-CoAP's inability to converge quickly with sudden bandwidth changes. We think that tuning the measurement window size m will improve the overall performance of the algorithm on all the nodes. This is an area to be explored.

Chapter 6 was concluded by introducing new approaches – considered as open research problems – which are mainly IDC-CoAP with probing and IDC-CoAP with exponential growth. In the former, we followed the Additive Increase Multiplicative Decrease approach with probing of the bandwidth. In the latter, we considered exponential increase to increment the sending rate. Firstly, we did not consider reducing the number of instructions. Secondly, and in some cases, the sending rate exceeded the residual bandwidth due to the exponential growth; more induced packet losses is the consequence. Also, packet retransmissions are increased which is a waste of significant energy. Hence, another important future work is to consider tuning the parameters of these algorithms. More importantly, a reduction in these algorithms' complexity while maintaining a good tradeoff between performance metrics is a must. Another important area to study is the dynamic update of the proposed algorithms' parameters according to the network's state. One way to accomplish this is by computing the loss probability p of the network periodically and update the parameters accordingly. This is an area of more research.

In addition to tuning the parameters of the previous algorithms and updating them dynamically, we may adopt a hybrid protocol that switches between the Additive Increase Multiplicative Decrease approach and the exponential growth-based approach, depending on the network's state of congestion. For instance, in Chapter 6, we have seen that some proposals for IDC-CoAP perform well in cases where bandwidth variation is not high. Therefore, switching between different approaches according to bandwidth variability will be considered as well.

Finally, we state that loss is sometimes the result of the algorithm's 'aggressiveness'. As a protocol gets more aggressive (e.g., by using a fast decremental weight to increase the sending rate), it creates more loss and thus, needs to send at a lower rate. However, a less aggressive algorithm is less responsive to the available bandwidth (e.g., one that employs a slow decremental weight to reduce the sending rate). Therefore, we can see that there are many facets to the problem, requiring more research efforts. We believe that fast convergence to efficiency requires a mechanism that detects the availability of unused bandwidth, which is an active research area. In general, we foresee that advances in this field will greatly benefit the congestion control research.

A

Expressions of A^{r+1} elements: Cayley-Hamilton theorem

Through Cayley-Hamilton theorem, we found the following terms of $A^{r+1} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ as follows:

$$a = \frac{\lambda_2\lambda_1^{r+1} - \lambda_1\lambda_2^{r+1}}{\lambda_2 - \lambda_1} + (1-p)(1-k)\frac{\lambda_2^{r+1} - \lambda_1^{r+1}}{\lambda_2 - \lambda_1} \tag{A.1}$$

$$b = q(1-k)\frac{\lambda_2^{r+1} - \lambda_1^{r+1}}{\lambda_2 - \lambda_1} \tag{A.2}$$

$$c = p(1-h)\frac{\lambda_2^{r+1} - \lambda_1^{r+1}}{\lambda_2 - \lambda_1} \tag{A.3}$$

$$d = \frac{\lambda_2\lambda_1^{r+1} - \lambda_1\lambda_2^{r+1}}{\lambda_2 - \lambda_1} + (1-q)(1-h)\frac{\lambda_2^{r+1} - \lambda_1^{r+1}}{\lambda_2 - \lambda_1} \tag{A.4}$$

B

Expressions of A^{r+1} elements: Diagonalization/similarity

We provide in this section an alternative method to compute a, b, c and d using diagonalization and similarity transformation.

$$A = \begin{bmatrix} (1-p)(1-k) & q(1-k) \\ p(1-h) & (1-q)(1-h) \end{bmatrix}$$

$$\text{Define } P = \begin{bmatrix} (1-p)(1-k) & q(1-k) \\ p(1-h) & (1-q)(1-h) \end{bmatrix}$$

which is formed by eigen vectors. Then:

$$A^{r+1} = P \begin{bmatrix} \lambda_1^{r+1} & 0 \\ 0 & \lambda_2^{r+1} \end{bmatrix} P^{-1}$$

$$P^{-1} = \begin{bmatrix} p(1-h) & (1-q)(1-h) - \lambda_2 \\ -p(1-h) & \lambda_1 - (1-q)(1-h) \end{bmatrix}$$

However,

$$p(1-h)[\lambda_1 - (1-q)(1-h)] - p(1-h)[\lambda_2 - (1-q)(1-h)] = p(1-h)(\lambda_1 - \lambda_2) = 1$$

Substituting for $p(1-h) = \frac{1}{(\lambda_1 - \lambda_2)}$, we obtain:

$$P^{-1} = \begin{bmatrix} \frac{1}{(\lambda_1-\lambda_2)} & \frac{(1-q)(1-h)-\lambda_2}{p(1-h)(\lambda_1-\lambda_2)} \\ \frac{1}{(\lambda_1-\lambda_2)} & \frac{\lambda_1-(1-q)(1-h)}{p(1-h)(\lambda_1-\lambda_2)} \end{bmatrix}$$

Simplifying further, we obtain:

$$a = \frac{\lambda_1^{r+1}[\lambda_1 - (1-q)(1-h)] - \lambda_2^{r+1}[\lambda_2 - (1-q)(1-h)]}{\lambda_1 - \lambda_2}$$

$$b = \frac{\lambda_1^{r+1}[\lambda_1 - (1-q)(1-h)][(1-q)(1-h) - \lambda_2]}{p(1-h)(\lambda_1 - \lambda_2)} + \frac{\lambda_2^{r+1}[\lambda_2 - (1-q)(1-h)][\lambda_1 - (1-q)(1-h)]}{p(1-h)(\lambda_1 - \lambda_2)}$$

$$c = \frac{\lambda_2^{r+1}p(1-h) - \lambda_1^{r+1}p(1-h)}{\lambda_2 - \lambda_1}$$

$$d = \frac{\lambda_1^{r+1}[(1-q)(1-h) - \lambda_2] + \lambda_2^{r+1}[\lambda_1 - (1-q)(1-h)]}{\lambda_1 - \lambda_2}$$

Simplifying using the fact that

$$[\lambda_1 - (1-q)(1-h)][\lambda_2 - (1-q)(1-h)] = -pq(1-h)(1-k)$$

We found the same expressions for a, b, c and d of A^{r+1} calculated using Cayley-Hamilton theorem in Appendix A.

Bibliography

[1] MSPSim, Emulator of the MSP430 series. https://github.com/contiki-ng/mspsim. last accessed: 01.10.2020.

[2] Zolertia Z1, Low-power Wireless Sensor Network Platform. https://github.com/Zolertia/Resources/wiki/The-Z1-mote. last accessed: 01.10.2020.

[3] S. Abbasloo, Y. Xu, and H. J. Chao. C2TCP: A Flexible Cellular TCP to Meet Stringent Delay Requirements. *IEEE Journal on Selected Areas in Communications*, 37(4):918–932, April 2019. ISSN 1558-0008.

[4] Maykel Alonso-Arce, Javier Añorga, Saioa Arrizabalaga, and Paul Bustamante. A wireless sensor network PBL lab for the master in telecommunications engineering. In *2016 Technologies Applied to Electronics Teaching (TAEE)*, pages 1–8, 2016.

[5] E. Ancillotti and R. Bruno. BDP-CoAP: Leveraging Bandwidth-Delay Product for Congestion Control in CoAP. In *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, pages 656–661, April 2019.

[6] Mohsen Hallaj Asghar and Nasibeh Mohammadzadeh. Design and simulation of energy efficiency in node based on mqtt protocol in internet of things. In *2015 International Conference on Green Computing and Internet of Things (ICGCIoT)*, pages 1413–1417, 2015. doi: 10.1109/ICGCIoT.2015.7380689.

[7] M. Barton, H. Lemberg, M. Sarraf, and C. Hamilton. Performance analysis of packet loss concealment in mobile environments with a two-state loss model. In *2010 IEEE International Workshop Technical Committee on Communications Quality and Reliability (CQR 2010)*, pages 1–6, June 2010.

[8] Olaf Bergmann. libcoap: C-implementation of CoAP. https://libcoap.net/. last accessed: 01.03.2020.

[9] A. Betzler, C. Gomez, I. Demirkol, and J. Paradells. CoAP congestion control for the internet of things. *IEEE Communications Magazine*, 54(7):154–160, July 2016. ISSN 0163-6804.

[10] August Betzler, Carles Gomez, Ilker Demirkol, and Josep Paradells. CoCoA+: An advanced congestion control mechanism for CoAP. *Ad Hoc Networks*, 33:126 – 139, 2015. ISSN 1570-8705.

[11] R. Bhalerao, S. S. Subramanian, and J. Pasquale. An analysis and improvement of congestion control in the CoAP Internet-of-Things protocol. In *2016 13th IEEE Annual Consumer Communications Networking Conference (CCNC)*, pages 889–894, Jan 2016.

[12] Eric W. Biederman. veth - Virtual Ethernet Device. http://manpages.ubuntu.com/manpages/bionic/en/man4/veth.4.html. last accessed: 01.03.2020.

[13] A. Bildea, O. Alphand, F. Rousseau, and A. Duda. Link quality estimation with the gilbert-elliot model for wireless sensor networks. In *2015 IEEE 26th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pages 2049–2054, 2015. doi: 10.1109/PIMRC.2015.7343635.

[14] Simone Bolettieri, Giacomo Tanganelli, Carlo Vallati, and Enzo Mingozzi. pCoCoA: A precise congestion control algorithm for coap. *Ad Hoc Networks*, 80:116 – 129, 2018. ISSN 1570-8705.

[15] C Bormann, A Betzler, C Gomez, and I Demirkol. Coap simple congestion control/advanced. *ID: draft-bormann-core-cocoa-02*, 2014.

[16] Olivier Capp, Eric Moulines, and Tobias Ryden. *Inference in Hidden Markov Models*. Springer Publishing Company, Incorporated, 2010. ISBN 1441923195, 9781441923196.

[17] Neal Cardwell, Yuchung Cheng, C. Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. BBR: Congestion-Based Congestion Control. *ACM Queue*, 14, September-October:20 – 53, 2016.

[18] Ajay Chaudhary, Sateesh K. Peddoju, and Kavitha Kadarla. Study of internet-of-things messaging protocols used for exchanging data with external sources. In *2017 IEEE 14th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pages 666–671, 2017. doi: 10.1109/MASS.2017.85.

[19] Yuang Chen and Thomas Kunz. Performance evaluation of iot protocols under a constrained wireless access network. In *2016 International Conference on Selected Topics in Mobile Wireless Networking (MoWNeT)*, pages 1–7, 2016. doi: 10.1109/MoWNet.2016.7496622.

[20] M. Collina, M. Bartolucci, A. Vanelli-Coralli, and G. E. Corazza. Internet of things application layer protocol analysis over error and delay prone links. In *2014 7th Advanced Satellite Multimedia Systems Conference and the 13th Signal Processing for Space Communications Workshop (ASMS/SPSC)*, pages 398–404, Sept 2014.

[21] A. Dunkels, B. Gronvall, and T. Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *29th Annual IEEE International Conference on Local Computer Networks*, pages 455–462, 2004.

[22] E. O. Elliott. Estimates of error rates for codes on burst-noise channels. *The Bell System Technical Journal*, 42(5):1977–1997, Sept 1963. ISSN 0005-8580. doi: 10.1002/j.1538-7305.1963.tb00955.x.

[23] Simon Frohn, Sascha GüBner, and Christoph Lindemann. Analyzing the effective throughput in multi-hop ieee 802.11n networks. *Comput. Commun.*, 34(16):1912–1921, October 2011. ISSN 0140-3664. doi: 10.1016/j.comcom.2011.04.003.

[24] W. Gao, J. H. Nguyen, W. Yu, C. Lu, D. T. Ku, and W. G. Hatcher. Toward emulation-based performance assessment of constrained application protocol in dynamic networks. *IEEE Internet of Things Journal*, 4(5):1597–1610, Oct 2017.

[25] Weichao Gao, James Nguyen, Wei Yu, Chao Lu, and Daniel Ku. Assessing performance of constrained application protocol (coap) in manet using emulation. RACS '16, page 103–108. Association for Computing Machinery, 2016. ISBN 9781450344555. doi: 10.1145/2987386.2987400.

[26] E. N. Gilbert. Capacity of a burst-noise channel. *The Bell System Technical Journal*, 39(5):1253–1265, Sept 1960. ISSN 0005-8580. doi: 10.1002/j.1538-7305.1960.tb03959.x.

[27] Jorge Granjal, Edmundo Monteiro, and Jorge Sá Silva. Security for the internet of things: A survey of existing protocols and open research issues. *IEEE Communications Surveys Tutorials*, 17(3):1294–1312, 2015. doi: 10.1109/COMST.2015.2388550.

[28] Stephen Hemminger. NetEm - Network Emulator. http://manpages.ubuntu.com/manpages/cosmic/en/man8/tc-netem.8.html. last accessed: 01.03.2020.

[29] R. Herrero. Dynamic CoAP mode control in Real Time Wireless IoT Networks. *IEEE Internet of Things Journal*, pages 1–1, 2018. ISSN 2327-4662.

[30] R. Herrero and C. St-Pierre. Dynamic forward error correction in wireless real-time internet of things networks. *IET Networks*, 6(6):218–223, 2017. ISSN 2047-4954. doi: 10.1049/iet-net.2017.0110.

[31] Rolando Herrero. Analytical model of iot coap traffic. *Digital Communications and Networks*, 2018. ISSN 2352-8648. doi: https://doi.org/10.1016/j.dcan.2018.07.001.

[32] M. Hock, R. Bless, and M. Zitterbart. Experimental evaluation of bbr congestion control. In *2017 IEEE 25th International Conference on Network Protocols (ICNP)*, pages 1–10, 2017. doi: 10.1109/ICNP.2017.8117540.

[33] Adrian Hornsby and Rod Walsh. From instant messaging to cloud computing, an xmpp review. In *IEEE International Symposium on Consumer Electronics (ISCE 2010)*, pages 1–6, 2010. doi: 10.1109/ISCE.2010.5523293.

[34] S. M. Riazul Islam, Daehan Kwak, MD. Humaun Kabir, Mahmud Hossain, and Kyung-Sup Kwak. The internet of things for health care: A comprehensive survey. *IEEE Access*, 3:678–708, 2015. doi: 10.1109/ACCESS.2015.2437951.

[35] V. Jacobson. Congestion Avoidance and Control. *SIGCOMM Comput. Commun. Rev.*, 18(4):314–329, August 1988. ISSN 0146-4833.

[36] Shobhit Jain, Vinoth Kumar N., A. Paventhan, V. Kumar Chinnaiyan, V. Arnachalam, and Pradish M. Survey on smart grid technologies- smart metering, iot and ems. In *2014 IEEE Students' Conference on Electrical, Electronics and Computer Science*, pages 1–6, 2014. doi: 10.1109/SCEECS.2014.6804465.

[37] I. Jarvinen, I. Raitahila, Z. Cao, and M. Kojo. FASOR Retransmission Timeout and Congestion Control Mechanism for CoAP. In *2018 IEEE Global Communications Conference (GLOBECOM)*, pages 1–7, Dec 2018.

[38] Ilpo Järvinen, Iivo Raitahila, Zhen Cao, and Markku Kojo. Is CoAP congestion safe? In *Proceedings of the Applied Networking Research Workshop*, ANRW '18, page 43–49, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450355858.

[39] Jeong Geun Kim and M. M. Krunz. Delay analysis of selective repeat arq for a markovian source over a wireless channel. *IEEE Transactions on Vehicular Technology*, 49(5):1968–1981, 2000. doi: 10.1109/25.892598.

[40] Ming-Shen Jian, Jun-Yi Wu, Jing-Yan Chen, Yue-Jyun Li, Yi-Cheng Wang, and Hao-Yi Xu. Iot base smart home appliances by using cloud intelligent tetris switch. In *2017 19th International*

Conference on Advanced Communication Technology (ICACT), pages 589–592, 2017. doi: 10.23919/ICACT.2017.7890158.

[41] J. Joshi, D. Kurian, S. Bhasin, S. Mukherjee, P. Awasthi, S. Sharma, and S. Mittal. Health monitoring using wearable sensor and cloud computing. In *2016 International Conference on Cybernetics, Robotics and Control (CRC)*, pages 104–108, Aug 2016.

[42] V. Karagiannis, P. Chatzimisios, F. Vázquez-Gallego, and J. Alonso-Zarate. A Survey on Application Layer Protocols for the Internet of Things. *Transaction on IoT and Cloud Computing (TICC), 2015*, 1(1), January 2015.

[43] J. J. Lee, S. M. Chung, B. Lee, K. T. Kim, and H. Y. Youn. Round trip time based adaptive congestion control with CoAP for sensor network. In *2016 International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 113–115, May 2016.

[44] Jie Lin, Wei Yu, Nan Zhang, Xinyu Yang, Hanlin Zhang, and Wei Zhao. A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications. *IEEE Internet of Things Journal*, 4(5):1125–1142, 2017. doi: 10.1109/JIOT.2017.2683200.

[45] Aastha Maheshwari and Rajesh Kumar Yadav. Analysis of congestion control mechanism for IoT. In *2020 10th International Conference on Cloud Computing, Data Science Engineering (Confluence)*, pages 288–293, 2020.

[46] Biswajeeban Mishra and Attila Kertesz. The use of mqtt in m2m and iot systems: A survey. *IEEE Access*, 8:201071–201086, 2020. doi: 10.1109/ACCESS.2020.3035849.

[47] Nitin Naik. Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP. In *2017 IEEE International Systems Engineering Symposium (ISSE)*, pages 1–7, 2017. doi: 10.1109/SysEng.2017.8088251.

[48] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt. Cross-Level Sensor Network Simulation with COOJA. In *Proceedings. 2006 31st IEEE Conference on Local Computer Networks*, pages 641–648, 2006.

[49] Silvio Quincozes, Tubino Emilio, and Juliano Kazienko. Mqtt protocol: Fundamentals, tools and future directions. *IEEE Latin America Transactions*, 17(09):1439–1448, 2019. doi: 10.1109/TLA.2019.8931137.

[50] Vishal Rathod, Natasha Jeppu, Samanvita Sastry, Shruti Singala, and Mohit P. Tahiliani. Cocoa++: Delay gradient based congestion control for internet of things. *Future Generation Computer Systems*, 100:1053–1072, 2019. ISSN 0167-739X.

[51] Partha Pratim Ray, Mithun Mukherjee, and Lei Shu. Internet of things for disaster management: State-of-the-art and prospects. *IEEE Access*, 5:18818–18835, 2017. doi: 10.1109/ACCESS.2017.2752174.

[52] Pasi Sarolahti and Alexey Kuznetsov. Congestion Control in Linux TCP. In *Proceedings of the FREENIX Track: 2002 USENIX Annual Technical Conference*, pages 49–62, Berkeley, CA, USA, 2002. USENIX Association. ISBN 1-880446-01-4.

[53] Zach Shelby, Klaus Hartke, and Carsten Bormann. The Constrained Application Protocol (CoAP), RFC 7252. https://rfc-editor.org/rfc/rfc7252.txt, June 2014.

[54] Adika Bintang Sulaeman, Fransiskus Astha Ekadiyanto, and Riri Fitri Sari. Performance evaluation of http-coap proxy for wireless sensor and actuator networks. In *2016 IEEE Asia Pacific Conference on Wireless and Mobile (APWiMob)*, pages 68–73, 2016.

[55] Muhammad Ashar Tariq, Murad Khan, Muhammad Toaha Raza Khan, and Dongkyun Kim. Enhancements and challenges in coap—a survey. *Sensors*, 20(21), 2020. ISSN 1424-8220. doi: 10.3390/s20216391.

[56] D. Thangavel, X. Ma, A. Valera, H. X. Tan, and C. K. Y. Tan. Performance evaluation of MQTT and CoAP via a common middleware. In *2014 IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, pages 1–6, April 2014.

[57] Ubuntu. netns - Network Namespace Management. http://manpages.ubuntu.com/manpages/bionic/man8/ip-netns.8.html. last accessed: 01.03.2020.

[58] Diana C. Yacchirema, Carlos E. Palau, and Manuel Esteve. Enable iot interoperability in ambient assisted living: Active and healthy aging scenarios. In *2017 14th IEEE Annual Consumer Communications Networking Conference (CCNC)*, pages 53–58, 2017. doi: 10.1109/CCNC.2017.7983081.

[59] Dennis G. Zill and Michael R. Cullen. *Advanced Engineering Mathematics*. Jones and Bartlett Publishers, ISBN-13: 978-0-7637-3914-0, 2006.