Politechnika Śląska
Wydział Automatyki, Elektroniki
i Informatyki

Godlove Suila Kuaban

Transient Markovian and Diffusion Approximation Models for Performance Analysis of Computer Networks and Battery Energy Storage Systems

Doctoral Dissertation prepared under the supervision of
Prof dr hab. inż Tadeusz Czachórski

Gliwice 2023

Contents

Abstract

Quality of Service (QoS), security, energy consumption, and cost (deployment and operational cost) are key constraints in the design and provisioning of computer systems, networks, and ICT (Information and Communication Technology) infrastructures. These metrics are tightly connected. Thus, modern computer systems and networks should be designed and deployed in such a way as to find a reasonable tradeoff between QoS, security, energy consumption, and cost. Queueing theory is a commonly used tool in such studies. Queueing models based on Markov chains and diffusion approximation have been extensively used to model problems in computer systems and networks. One such application of these models is to evaluate the performance of queues (waiting lines of jobs, processes, data packets, energy packets, etc.) in computer systems and networks. Diffusion approximation is well suited for the transient analysis of queueing systems in computer systems and networks. It provides a methodology to perform a time-dependent analysis of the performance metrics (queue size, waiting time, and probability of rejection when the storage memory is full and subsequently arriving customers are rejected) as the parameters of the interarrival and service (processing) times changes over time. Another advantage of the diffusion approximation modelling methodology is the possibility of using realistic distributions of the interarrival and service times obtained from measured data.

The packet sizes generated from access networks vary from a few bytes in IoT and wireless sensor networks to 1500 bytes in Internet Protocol (IP) networks. The transmission of massive amounts of small packets (sometimes with randomly varying sizes) generated by access networks through high-speed Internet core networks to other access networks or cloud computing data centres has introduced several challenges such as poor throughput, underutilisation of network resources, and higher energy consumption. Packet aggregation mechanisms were developed to resolve these challenges. Packet aggregation mechanisms aggregate smaller packets into a larger payload packet, and these groups of aggregated packets will share the same header, hence increasing throughput, improving resource utilisation, and reduction in energy consumption. In Chapter 2, we present a review of packet aggregation applications in access networks (IoT and 4G/5G mobile networks), optical core networks, and cloud data centre networks. We also propose diffusion-based analytical models that can be used to design and evaluate the performance of packet aggregation mechanisms. We also demonstrated the use of measured traffic from real networks to evaluate the performance of packet aggregation mechanisms using simulation and analytical models. Despite its benefits, packet aggregation increases the packets' delays and may not be suitable for traffic belonging to real-time applications.

Queues of packets or jobs are unavoidable in computer network devices (e.g., routers and switches) due to the stochastic nature of the interarrival times of packets, processing and transmission times of packets, and sizes of packets. Queueing also results from sharing limited computational and communication resources. Queueing degrades the quality of service (QoS) experienced by users by increasing packet delays, packet loss probability, and jittering experienced by multimedia traffic. Queueing theory models (e.g., Markovian, fluid flow, and diffusion approximation models) are very useful tools for the analysis of the performance of computer systems and networks. In chapter 3, we present the architectures of hardware and software SDN switches and model the flow matching (lookup) mechanisms used in these switches. We propose a tractable diffusion approximation for both the transient and steady-state behaviour of a network router. Using these results, we show that when SDN switches change the paths of flows frequently, the network's behaviour may often be far from its steady-state behaviour. In chapter 4, we present an overview of flexible routing

in SDN-based networks. We extend the methodology developed in chapter 3 to the time-dependent analysis of multiple SDN switches using diffusion approximations, which are very convenient to analyze in a time-dependent regime.

Markovian, fluid flow, and diffusion approximation models have recently been adapted to model the energy depletion process in battery energy storage systems for computer systems and ICT infrastructures. One of the most important criteria is minimizing energy consumption in designing and deploying battery-powered computer systems (e.g., IoT devices, mobile phones, UAVs). Energy consumption in battery-powered computer systems and network devices can be reduced by using energy-efficient hardware, software, and protocols. If the energy stored in the battery is completely drained, the computer system or network device is shut down. Thus, modelling energy consumption in battery-powered computer systems and networks and modelling the energy depletion process in batteries is essential. In chapter 5, we apply a diffusion or Brownian motion process to model the energy depletion process of a battery of an IoT device. We use the model to obtain the probability density function, mean, variance, and probability of the lifetime of an IoT device. Also, we study the influence of the active power consumption, sleep time, and battery capacity on the probability density function, mean, and probability of the lifetime of an IoT device. We use numerical examples to study the influence of battery depletion attacks on the distribution of the lifetime of an IoT device. We also introduce in our model an energy threshold after which the device's battery should be replaced to ensure that the battery is not completely drained before it is replaced.

The energy-harvesting technologies to harvest energy from external energy sources in the environment such as solar, thermal, wind, and vibration to power computer systems or to replenish the energy drawn from the batteries attached to these systems enable them to operate longer with minimal energy-related interruptions. Thus, an effort to ensure higher efficiency of the harvesting and more economical performance of these devices is necessary. In chapter 6, we present an architecture of a green base station site. We develop Markovian and diffusion approximation models to analyze the steady-state and transient-state performance of battery energy storage systems. We apply the Markovian and diffusion approximation model to derive the time after which the battery energy storage system is completely discharged or fully charged. By assuming that the energy harvesting and the energy consumption processes are exponentially distributed, we compare the result obtained from the Markovian model to those from diffusion approximation models.

Therefore, diffusion approximation is a practical tool for analysing the performance of computer systems and networks as it allows the use of measured packet interarrival times and packet service times (processing and transmission times) distributions. It is useful for analysing the transient behaviour of QoS parameters in SDN-based networks resulting in frequent changes in the flow forwarding rules in the SDN switches. Transient Markovian and diffusion models are suitable for modelling the energy depletion process in battery energy storage systems for computer networks and networks with stochastic energy harvesting and energy consumption processes.

Chapter 1

Introduction

The intensity of traffic transported by computer networks has complex stochastic nature. Since the interarrival of packets into the buffers in network equipment and the time required to process or transmit packets are random, packets are queued up in buffers before they are processed or transmitted. The time that packets spend waiting in buffers and the time required to process or transmit the packets constitute the delays experienced by the packet along the path from the source to the their destination. The formation of queues degrades the Quality of Service (QoS) experienced by the users of the networks as it causes packets to be delayed significantly from the time they are generated to the time that they are delivered to their destination. When the delay experienced by the packets varies, it introduces a jitter, which degrades the QoS experienced by users receiving multimedia traffic. Queueing also causes packet losses due to the dropping of packets when the buffers are full or when packets are dropped earlier as an active queue management mechanism to prevent buffer overflows.

Queueing theory is a valuable tool in the design of computer networks and their performance evaluation. Usually, queueing models attempt to abstract the behaviour of a network of routers as a network of queues and then use queueing theory to analyse the performance of a network device or a given network of devices. They help to estimate the overall transmission time and quality of the transmission. They are still being developed and applied to evaluate the performance of newly proposed network architectures and protocols, e.g. Software Defined Networks, the Internet of Things, and Cloud computing. The majority of models are limited to the analysis of steady states. It means that flows of packets considered in models are constant, and the obtained solutions do not depend on time. It is in glaring contrast with the flows observed in real networks where the perpetual changes of traffic intensities are due to the nature of users, sending variable quantities of data, multimedia traffic, and also due to the performance of traffic control algorithms which are trying to avoid congestion in networks, e.g. the algorithm of congestion window used in TCP protocol which is adapting the rate of the sent traffic to the observed losses or transmission delays. However, the analysis of transient states is much more difficult and complex.

Network planning and optimisation engineers often use queueing theory to enable them to select the specifications for their hardware resources or upgrade the network resources to match the expected user demand with an acceptable QoS and return on their investments. The models are also used to study the influence of network or network device parameters on the QoS metrics of the network. The models may also reflect the software feature of transmission protocols, congestion avoidance rules, and routers' scheduling

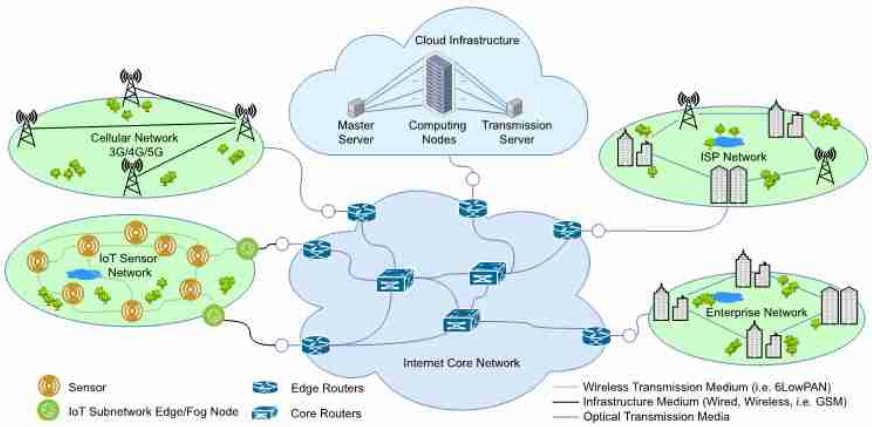

Figure 1.1: An architectural model consisting of access networks, Internet core networks and the cloud data centre.

rules. Traditionally, steady-state queueing models have been used for network design and optimisation as traffic changes in the network are not too frequent. After some time, the network attains a steady state. With the recent introduction of the Software Defined Networking (SDN) paradigm in which the routing decision is shifted from the router to a centralised server called the controller, the network may frequently be in a transient state. The data plane devices constantly collect data about the network's security, energy consumption and QoS parameters and send it to the controller, which then uses this information to perform route computations and update the packet forwarding rules in the routers. If this happens frequently, the network will constantly be in a transient state.

In this dissertation, we abstract a computer/telecommunication network architecture into a simplified architectural model shown in figure 1.1. It consists of the access, core, and data centre networks. Users access the services of telecommunication operators or internet service providers through access networks such as Digital Subscriber Line (DSL), Ethernet local area networks (LANs), Fiber-to-the-Home (FTTX), wireless LANs, mobile networks (e.g., 3G, 4G, and 5G), and the Internet of Things (IoT) access networks. The traffic generated by users connected to a given access network is transported through the core networks to users connected to other access networks or data centres (for users that want to access cloud services). The sizes of packets generated by various access networks vary significantly from tiny packets generated by IoT access networks (of just a few bytes) to IP packets (of more than a thousand bytes). The vast amounts of packets generated by access networks are usually aggregated into larger packets to increase throughput efficiency and reduce processing overhead.

Packet aggregation involves the assembly of smaller packets into larger ones. Packet aggregation provide many benefits at the level of core networks, such as increased spectral efficiency, energy efficiency, optimal resource utilisation [10], simplified traffic management, and significantly reduce protocol and signalling overhead. It significantly influences the networks' overall performance in terms of packet delay and packet losses. Therefore, packet aggregation and transmission queue management mechanisms must be carefully designed and parameterised.

Recently, queueing theory has been adapted to model battery energy storage systems for computer sys-

tems powered by batteries, e.g. [87, 88, 89, 100, 90, 95, 91, 73, 72, 125, 83, 123, 243]. The energy consumption of computer systems is not fixed. It varies with its computational demand and the number of packets processed or transmitted. One of the sources of randomness in the energy consumption of some computer systems is that users initiate computation, processing, or transmission of data at random times. Some computer systems are powered by batteries continuously recharged by energy harvested from renewable energy sources. The amount of energy generated by renewable energy sources sometimes fluctuates randomly with random environmental changes. The random fluctuation in the amount of energy harvested and delivered to the battery and the amount of energy drawn from the battery to power the computer system triggers the need to use stochastic models to analyse the dynamic changes in the battery's energy content. Stochastic modelling of the battery makes it possible to size the battery and estimate the time after which the energy stored in the battery could be completely depleted.

One of the approaches to analyse the dynamic changes in the amount of energy present in the battery at a given time with randomly changing recharging and energy consumption processes is to represent the energy present in the battery in discrete energy units, also called energy packets [87, 88, 89]. An energy packet is the minimum amount of energy required to transmit a single data packet or process a single job by treating the charging process as the delivery of energy packets into the battery and the energy consumption process as the departure of energy packets from the battery. Traditional queueing theory models can therefore be employed to estimate the battery's energy content at a given time, to size the battery, estimate the probability of completely draining or overcharging the battery, and the time required to drain the energy stored in the battery completely. Since energy is a continuous quantity, the changes in the amount of energy in the battery could be considered to be analogous to the changes of a fluid in a reservoir or can be approximated by diffusion or the Brownian motion process, as discussed in Chapters 5 and 6.

1.1 Contribution and thesis

Markovian and diffusion approximation models are not new and have been well developed and applied to model random processes in engineering, finance, physical science, and computer science. They are often used for the performance analysis or evaluation of computer systems and networks. In this dissertation, we adapt existing Markovian and diffusion approximation models to evalaute the performance of packet aggregation algorithms, SDN switches, a network of SDN switches, and battery energy storage systems for computer networks (with and without energy harvesting sources to replenish the energy drawn from the battery).

Most of the research work in designing and optimising SDN networks was based on the assumption that the network is always in a steady state. They are also based on the assumption that the interarrival time distribution follows a Poison process and that the service times are exponentially distributed. We analysed the functional behaviour of SDN switches and abstracted them into simple queueing models. We then adapted the G/G/1/N diffusion approximation model to evaluate the performance of the queues of packets in SDN switches. Since there is no restriction on the distribution interarrival and service time distributions, we used actual traffic data from the CAIDA data repository. The service time distribution was based on the SDN packet processing models we developed. We modelled the mechanism of searching software, and hardware flow tables in SDN switches to determine the service time distribution. We then extended the diffusion ap-

proximation model of a single SDN switch that we proposed to model the transient-state behaviour of a network of SDN switches (a queueing network of SDN switches).

We also applied the transient solution of diffusion or Brownian motion process to develop performance evaluation models of packet aggregation mechanisms that improve bandwidth and energy efficiency in IoT over IP networks, IoT over mobile networks (3G/4G/5G), IP over optical networks, and in data centre networks but introduces significant delays. We validated the accuracy of the proposed models using discrete event simulations. Our models are more realistic because we used actual traffic data from the CAIDA data repository to analyse the performance of the packet aggregation mechanism that has been implemented in commercially available routers, switches, and server machines.

We also applied the transient solution of Markovian and diffusion processes to develop models of the energy depletion process of battery energy storage systems used to power computer network devices (e.g., IoT devices, Fog computing nodes or access network nodes). Then, we used the models to estimate the device's lifetime (Time-To-Failure or Time-To-Shutdown) when all the stored energy is completely depleted. We also demonstrated the application of these models to analyse battery depletion attacks. We also proposed using these models to model battery energy storage systems in the presence of energy harvesting sources. We derived the time after which the battery energy storage system is completely discharged or fully charged. By assuming that the energy harvesting and the energy consumption processes are exponentially distributed, we compared the result obtained from the Markovian model to those from diffusion approximation models.

The thesis of this study can be stated as follows:
The diffusion approximation modelling formalism is a realistic and data-driven modelling technique that overcomes the limitations of other queueing theoretic techniques (e.g. Markov chains and fluid flow) for the transient analysis of Quality of Service (QoS) metrics in networks with time-dependent dynamic routing protocols (e.g., Software Defined Networking (SDN) core networks); and for the transient analysis of the energy depletion process in energy storage systems (ESS) in computer network infrastructure.

1.2 The tradeoff between QoS, security, energy, and cost

Quality of Service (QoS), security, energy consumption, and cost (deployment and operational cost) are key constraints in the design and provisioning of computer systems, networks, and ICT (Information and Communication Technology) infrastructures. In the early stage of the design, deployment, and provisioning of computer systems, networks, and ICT infrastructures, performance (QoS) and cost were the most critical constraints. The main objective was to improve the QoS experienced by the users to satisfy the users at an affordable cost.

Security became a significant problem as almost every computer system became connected through large-scale computer and telecommunication networks or the global internet infrastructure. However, implementing security mechanisms in designing and deploying computer systems and networks often limits QoS. It is because security mechanisms involve computationally expensive operations and limit the resources available for processing packets or user jobs [188]. Hence, the quality of service is degraded as reliable but computationally expensive security mechanisms are implemented in computer systems, networks, and ICT infrastructures.

Although security mechanisms protect computer systems, networks, and ICT infrastructures against se-

curity breaches that could result in service interruptions, loss of confidentiality, malicious data manipulation, information theft, and financial losses, the implementation of security mechanisms also increases capital or operational expenditures (Capex/Opex) [226]. Therefore, when designing and deploying secure computer systems, networks, and ICT infrastructures with acceptable QoS, we should consider the financial cost and ensure a reasonable tradeoff between QoS, security, and cost.

Since the 1980s, there has been a large-scale adoption of ICT technology to improve efficiency, QoS, and productivity in every sector of the modern economy and society (e.g., governance, finance, management, accounting, healthcare, manufacturing, telecommunication, retail, transportation, logistics, tourism, agriculture, entertainment, research, and development, etc.). The emergence of the World Wide Web and the modern Internet in the mid-1990s accelerated the digital revolution, with internet users increasing from 390 million in 2000 to 4.6 billion in 2021 [189]. Also, Internet of Things (IoT) technologies, together with other technological advances such as big data analytics, Artificial Intelligence (AI), automation, and unmanned electrical vehicles (e.g., self-driving vehicles and drones), are currently being used to transform every sector of the economy or society (e.g., agriculture, transportation, healthcare, manufacturing, security, etc.).

The increase in the number of computer networks and the size of networks (from access networks, core networks, to data centre networks) have caused the amount of energy consumed by ICT systems to increase significantly. In 2018, it was estimated in [71] that ICTs account for between 5% and 9% of the total electricity consumption and that by 2030, this figure could increase to between 10% and 20%. Mechanisms implemented to improve QoS and security of computer systems and networks also increase the amount of computation and communication; hence, the energy consumption of computer systems and networks also increases. Using dedicated hardware modules like Binary or Ternary Content Addressable Memory (BCAM/TCAM) to improve the processing speed of routers and switches significantly increased energy consumption as BCAMs/TCAMs are power-hungry hardware modules. Also, it should be noted that as the amount of computation and communication increases, the amount of heat generated by computer systems also increases, and an additional amount of energy is required for cooling. The increase in energy consumption often increases the energy bills incurred by ICT operators or service providers; hence, the operational cost increases in the case of mobile network operators, 70% of their energy bills resulting from the massive number of base stations deployed within the radio access networks[247].

In designing and deploying battery-powered computer systems (e.g., IoT device, mobile phones, UAVs), minimising energy consumption has been the most crucial design parameter. Because of the energy limitation in the design and deployment of battery-powered computer systems and networks, energy-efficient hardware, software, and protocols are used. Security mechanisms that provide an acceptable level of security without sacrificing the QoS and quicking draining the battery are used. Also, the mechanisms implemented to reasonably improve the QoS and security of computer systems and networks, and to minimise their energy consumption, should not drive up the cost or make the device unaffordable or less competitive. The critical energy constraint in the design and deployment of battery-powered computer systems can be relaxed by harvesting energy from the environment to replenish the energy drawn from the battery.

Figure 1.2 illustrates the relationship between QoS, security, energy consumption, and cost. These metrics are tightly connected. Modern computer systems and networks should be designed and deployed in such a way as to find a reasonable tradeoff between QoS, security, energy consumption, and cost. Therefore, a reasonable tradeoff should be maintained between QoS, security, energy consumption, and financial cost

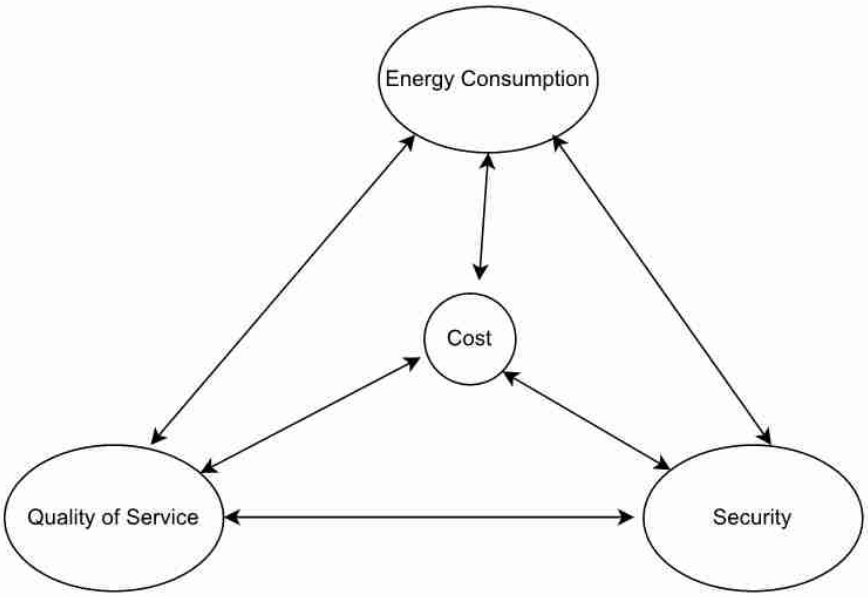

Figure 1.2: The QoS, security, energy, and cost tradeoff.

when designing and deploying computer systems and networks.

1.3 Background of performance modelling and motivation

Architects and designers of computer systems and networks often use experimental test beds, real computer systems (or networks), simulations, or mathematical modelling for planning, dimensioning, optimisation, and performance evaluations of their computer systems or networks. When studying or deploying computer systems and networks, they must carefully select design parameters and understand their relationship with the performance metrics of interest. Experimental testbeds are very costly and time-consuming to set up and conduct experiments. The difficulties of using experimental test beds can be overcome with computer simulations and mathematical modelling. However, the results obtained from experimental test beds are very realistic as simulations and mathematical models cannot reproduce or describe some behavioural aspects of computer systems and networks.

Simulation is a software approach to create a virtual environment that emulates (reproduces the function or action) the behaviour of a physical system to study its performance or properties. Computer simulations enable the architect or designer to create an abstract model of the computer system or network under study to emulate the interaction between the various relevant entities of the computer system or network. Mathematical modelling of computer systems and networks is the process of describing or analysing computer systems and networks using mathematical concepts, equations, or language. It provides a rigorous understanding of the relationship between the computer system's or network's design parameters and the performance metrics of interest.

Therefore, in situations where closed-form mathematical models can conveniently approximate the computer system or network under consideration, it is preferable to analyse or estimate its performance using

the closed-form analytical model without spending considerable time performing simulation studies or experimental setups. However, the limitation of closed-form mathematical models is that they are developed using simplifying assumptions that may deviate from reality. Also, some mathematical models are very difficult to solve analytically and are solved numerically using sophisticated numerical algorithms, requiring a significant amount of time. Thus, computer simulations are relatively time-consuming when compared to mathematical modelling is only sometimes true. In practice, both simulation and mathematical modelling are used for the analysis and evaluation of the performance of computer systems and networks. The accuracy of mathematical models is often validated using computer simulations.

Most mathematical models used for analysing and evaluating computer networks were developed using queueing theory. Queues are inevitable in computer systems and networks due to the occurrence of random events (e.g., the arrival of packets or jobs at a random time and random times required to process packets or jobs) and the sharing of scarce resources (e.g., CPU, the Random Access Memory (RAM), the buffers, the disk, the data bus, switching fabrics, and the ports) between the packets, jobs or processes. Queueing theory has been the cornerstone of network performance analysis. Still, network practitioners have always avoided queueing theory analysis in favour of simulations because of the need to understand the complex mathematics involved [16].

Queuing models were first proposed more than a hundred years ago by Agner Krarup Erlang for the evaluation of the performance of the Copenhagen telephone exchange. Also, the traffic analyst and the then director of Norwegian Televerket (now Telenor Group), Tore Olaus Engset, proposed queueing models for the analysis and evaluation of the performance of telephone exchanges. Their analysis was based on Markovian models. To apply Markovian models in the analysis of telephone exchanges, they assumed that the request for new connections follows a Poisson process and the duration of the connections is exponentially distributed. Many mathematicians, like Kolmogorov, Khinchin, Crommelin, Palm, and Takács, contributed to the development of queueing theory, and their models found many applications[48], one of them being the performance evaluations of computer systems and networks. They were later adapted for analysing and evaluating the performance of computer systems and packet-based computer networks. There were also adapted and used to analyse the transient behaviour of queues in computer systems and networks, as the intensity of traffic flows generated by users or applications (e.g. internet applications) is permanently changing [60]. Recently, Queueing models have been adapted for modelling battery energy storage systems for computer systems and networks, as discussed in chapters 5 and 6.

To use Markovian models to analyse or evaluate the performance of computer systems or networks, it is often assumed that the interarrival times of packets, jobs, or processes to a queue to be processed by a shared resource follows a Poison distributed and that the processing times are exponentially distributed. Markovian processes are used to model systems with limited memory. Therefore, Markovian processes are stochastic processes that have "memoryless" property in which the future of the process depends only on the present state, regardless of its history. They are used to model systems in many fields, including communication, transportation, image processing, bioinformatics, project management, mathematics, physics, finance, etc.[119]. A Marlovian process satisfies the "memoryless" property often expressed as:

$$\begin{aligned} P[X(t_{n+1}) = x_n + 1 | X(t_n) = x_n, X(t_{n-1}) = x_n - 1, + \cdots +, X(t_1) = x_1] \\ = P[X(t_{n+1} = x_{n+1} | X(t_n) = x_n] \end{aligned} \tag{1.1}$$

Where $t_1 < t_2 < \cdots < t_n < t_{n+1}$ and x_i is an element of discrete state space. It means that if the set of random variables (X_n) form a Markov chain, then the probability that the next state is (X_{n+1}) depends only on the current state (X_n) and not on the previous state. Hence, the Markovian process is an evolutionary time process in which the future states of the process depend entirely on the present state and are independent of its history or past states.

Markovian processes can be broadly classified into Discrete Time Markov Chains (DTMC) and Continuous Time Markov Chains (CTMC). In DTMC, the transitions from one state to another happen at fixed or discrete time instances. In CTMC, the transitions from one state to another happen at random times chosen from a continuous interval. There are two sources of randomness in CTMC; the first is an embedded discrete time jump chain (that is, considering the Markov process at the moment at which a change of state takes place) and exponentially distributed holding times (that is the sojourn which the Markov process spends at a particular state follow distribution). In this dissertation, we considered CTMC. Consider a CTMC in which a process can only jump forward to the next state or return to the previous state (from which it left); then, the transition rate from state *i* to state *j* is given by:

$$\begin{aligned} q_{ij} &= \lim_{\Delta t \to 0} \frac{P_{ij}(\Delta t)}{\Delta t} \\ P_{ij}(t) &= P[X_{t+s}| = j | X_s = i] \end{aligned} \tag{1.2}$$

Where $P_{ij}(t)$ is the state probability that the Markov Process will be in state *j* after the time interval *t*, given that it is currently at state *i* at time *s*. Whenever a CTMC enters a state *i*, it spends an amount of time called the dwell time or the holding time that is exponentially distributed. At the end of the expiration of the holding time, the process makes a transition (jump) to another state j with a probability P_{ij} [119] where

$$\sum_j P_{ij} = 1 \tag{1.3}$$

Suppose that after a short time Δt, the CTMC can make jumps into and out of the state *i* then the probability that the chain will be in state *i* at time*t* is the sum of all flows into and out of the state *i* given by Chapman-Kolmogorov equations for continuous time Markov chains as:

$$\begin{aligned} P_i(t+\Delta t) &= P_i(t)[1 - \sum_{j \neq i} q_{ij}\Delta t] + \sum_{j \neq i} P_j(t) q_{ji} \Delta t \\ P_i(t+\Delta t) - P_i(t) &= P_i(t) \sum_{j \neq i} q_{ij}\Delta t + \sum_{j \neq i} P_j(t) q_{ji} \Delta t \\ \frac{P_i(t+\Delta t) - P_i(t)}{\Delta t} &= P_i(t) \sum_{j \neq i} q_{ij} + \sum_{j \neq i} P_j(t) q_{ji} \\ \lim_{\Delta t \to 0} \frac{P_i(t+\Delta t) - P_i(t)}{\Delta t} = \frac{dP_i(t)}{dt} &= P_i(t) \sum_{j \neq i} q_{ij} + \sum_{j \neq i} P_j(t) q_{ji} \end{aligned} \tag{1.4}$$

$$\tag{1.5}$$

Equation ((1.4)) can be written in a compact form as

$$\frac{dP(t)}{dt} = P(t)Q \tag{1.6}$$

where $P(t) = [P_1(t), P_2(t), P_3(t) \cdots P_n(t) \cdots]$ is the state probability vector, which in the context of queueing systems represent the probability of having n number of customers in the queue at time t and.

$$\mathsf{Q} = \begin{pmatrix} q_{11} & q_{12} & q_{13} & q_{14} \cdots \\ q_{21} & q_{22} & q_{22} & q_{24} \cdots \\ q_{31} & q_{32} & q_{32} & q_{34} \cdots \\ \vdots & \vdots & \ddots & \vdots \end{pmatrix}$$

is the rate transition matrix or the intensity matrix. In steady-state (i.e., as $t - - > \infty$), $\frac{dP(t)}{dt} = 0$, $P(t) = [P_1(t), P_2(t), P_3(t) \cdots P_n \cdots]$ becomes $P = [P_1, P_2, P_3 \cdots P_n \cdots]$ the the system of equations in (1.4) and (1.6) becomes a system of linear equations. Its transient solution can be expressed in the form

$$P(t) = P(0)e^{Qt} = \sum_{k=1}^{\infty} \frac{(Qt)^k}{k!} \tag{1.7}$$

Where P(0) is the initial condition.

The most simple Markovian queueing system is the infinite capacity queueing system written in Kendell's notation as M/M/1. The first and second M in the notation means that the interarrival time of customers (jobs) follows a Poison process, and service times are exponentially distributed. Notation 1 means that only a single server is processing the jobs. M/M/1 model is straightforward and has been used to model many queueing systems in many fields, including computer networks. In computer systems and networks, packets, jobs, or processes waiting to be processed are stored in memory that is limited in capacity. Thus, M/M/1/N queueing systems are preferable; N is the size of the memory or buffer. The packets or jobs coming when the system is full are dropped or lost. Markovian queueing systems with more than one server can be written in Kendall's notation as M/M/c and M/M/c/N, where c is the number of servers. Well-known Markovian models like M/M/1, M/M/1/N, M/M/c, and M/M/c/N have closed-form steady-state solutions and are widely used to model queueing-related problems.

For queueing models in computer systems and networks, the performance metrics that are often required include the mean number of packets or jobs waiting in the queue to be processed (mean queue size), the time that packets or jobs spend waiting in queues (waiting time), and the probability that the storage space will be filled and newly arriving packets will be dropped (blocking probability). After determining the state probabilities from the queueing model, the mean number of packets present in the queue can be determined. The mean waiting can be determined using Little's law (dividing the mean number of packets in the queue by the mean arrival rates of packets). The blocking probability is the state probability $P_N(t)$ or P_N (the probability that there are N packets in the queue at time t or in steady-state). When the number of packets waiting in the queue reaches N, then all the waiting space is occupied any subsequent packets or jobs that attempts to join the queue will be dropped or rejected.

The main disadvantage of using Markov chains to model queueing-related problems and other problems in computer systems and networks is the so-called "state explosion" problem, especially for complex systems with large states. Each state of the Markov chain corresponds to one state of the system, and the number of equations equals the number of states. As the number of states increases, the system of equations to be solved significantly increases. The system of equations describing the Markov chain of the modelled system can be solved using well-known numerical methods and software packages (e.g., MATLAB). The

steady-state solution of the Markov chains can be obtained using Markov solvers (QNAP [198], XMARCA [221], PRISM [152]). We used PRISM to solve a complex Markov chain (for a Markovian model) for the aggregation of shorter packets into larger ones in [232]. A software package called Olymp [191] was developed by some of our colleagues at the Institute of Theoretical and Applied Informatics, Polish Academy of Sciences (IITiS-PAN) to solve Markov chains with a large number of states. The transient solution of the Chapman-Kolmogorov equations describing the Markov chains can be solved using the Runge-Kutta Method (e.g., [148, 151, 147].

Another drawback of using Markov chains to model queueing-related problems and other types of problems in computer systems is the assumption that the interarrival times of customers or jobs follow a Poisson distribution and that their service times are exponentially distributed. In reality, these assumptions are not always valid as real systems exhibit complex behaviour behaviours that make the distributions of the interarrival and service times deviate from exponential distributions. As a result, queueing models with general interarrival and service time distributions were developed (e.g., G/G/1, G/G/1/N, G/G/c, and G/G/c/N), where G in Kendell's notation means that the distributions of the interarrival and service times are not restricted to any of the well-known distributions.

Since the assumption that the interarrival times follows a Poison distribution and that the service times are exponentially distributed is not always true in reality, approximate queueing models have been proposed to model queueing systems without any restrictions on the interarrival and service time distributions. For an M/G/1 queue, the approximation for the mean queue size was by Pollazcek and Khintichine in the 1930s (the so-called Pollazcek-Khintichine formula) [37]. Well-known approximations for the mean queue size for G/G/1 queueing systems were proposed by William G. Marchal [171] and also by W. Kraemer and M. Langenbach-Belz [142]. Approximate models also eliminate the "state explosion problem" as these models are scalable for any number of states or number of customers present in the queue and for studying network protocols (e.g., TCP-IP protocols)[234].

Another modelling method often used to approximate queueing models is the fluid flow approximation. Fluid flow approximation has been used in [48] to model queues in computer systems and networks. The limitation of fluid flow approximation is that it can be used to estimate only the average queue size. Still, information about the variance or the queue size distribution needs to be obtained. In modelling queues in computer networks, information about the variation of the queue size is important as can provide information about the maximum jitter (or peak-to-peak) jitter, which is an important performance metric for multimedia traffic. Fluid flow models are less complex and are, therefore, suitable for modelling large computer networks.

A more complex approximation that caters for the limitations of fluid flow approximation is diffusion approximation. Diffusion approximation queueing models are based on the approximation of the changes in the number of customers (packets or jobs) present in the queue by a Brownian motion or diffusion process. A diffusion process is a strong Markov process with continuous time and continuous space (continuous sample path), [106]. Unlike Markov processes discussed above, diffusion processes are stochastic processes whose state space is the continuum of real numbers, and their state changes or transitions occur at all times. That is, within a short time interval, a diffusion or Brownian motion process can only undergo a small displacement or change of state. The occurrence of Brownian motion was first observed by Robert Brown in 1827 when he observed that when pollen grains are suspended in a fluid, they undergo random displacements due to

collision or bombardments with molecules of the fluid. 80 years later, in 1905, Albert Einstein developed a satisfactory theory to explain the phenomenon of diffusion or Brownian motion. A rigorous theoretical foundation of Brownian motion was developed by Norbert Wiener in 1923 and is sometimes referred to as the Weiner process. Therefore, diffusion processes are Markov processes. The state space is continuous [44].

Diffusion processes are frequently used to model or approximate more complex and analytically intractable stochastic processes. Kobayashi proposed the use of diffusion approximation to solve non-product form queueing networks; he obtained the equilibrium queue distribution [133] and the nonequilibrium (transient) queue distribution [134] for queues in a network. The use of reflecting barriers (the probability of the empty queue is zero) limited the models to cases of heavy traffic. The introduction of the barriers with instantaneous jumps by Gelenbe [85, 101, 86] enhanced the method's precision, which has been used in this form since then. A proposition by Czachorski of an analytical-numerical algorithm to solve diffusion equations ([48, 52, 60, 61]) made the approach relatively easy in the case of transient states analysis.

Consider a queueing system in which customers (e.g., jobs or packets in the case of a computer system or network device) arrive with a mean rate λ and are processed with a mean service rate of μ. Let $A(t)$ be the cumulative number of customers that have arrived at the queueing station up to the time t and $S(t)$ be the cumulative number of customers that have been served up to time t (assuming that after a customer is served, it immediately leaves the server). The number of customers present in the queueing station at time t is

$$N(t) = A(t) - S(t) \tag{1.8}$$

The changes in the number of customers present in the queueing station within a small time interval $[t\ t+\Delta]$ is

$$\begin{aligned} N(t+\Delta) - N(t) &= A(t+\Delta) - A(t) - S(t+\Delta) - S(t) \\ \Delta N(t) &= \Delta A(t) - \Delta S(t) \end{aligned} \tag{1.9}$$

If the interarrival time and service time processes are both independent and identically distributed and Δ is sufficiently large such that many events (arrival of customers into the queue and departure of customers from the server) occur between the time interval t and $t + \Delta$, then the changes in the queue size $\Delta N(t)$ are approximately normally distributed with mean $E[\Delta N(t)] = (\lambda - \mu)\Delta$ and variance $var[\Delta N(t)] = (C_A^2\lambda + C_B^2\mu)\Delta$ [133]. Where $C_A^2 = \sigma_A^2\lambda^2$ and $C_B^2 = \sigma_B^2\mu^2$ are the squared coefficients of variation of the interarrival and service time distributions. The parameters σ_A^2 and σ_B^2 are the variance of the interarrival time and service time distributions, respectively.

Therefore, to model a queueing station using diffusion approximation, the discrete-state process, $\{N(t), t \geqslant 0\}$, of the number of customers in the queue is replaced by an appropriate continuous-state diffusion process, $\{X(t), t \geqslant 0\}$, the changes of which $dX(t) = X(t+\Delta) - X(t)$ are normally distributed with mean $(\beta\Delta)$ and variance $(\alpha\Delta)$, where β, and α are coefficients of the diffusion equation that describes the dynamic evolution of the process. The stochastic partial differential equation that describes the dynamic evolution of the diffusion process is given by [44]:

$$\frac{\partial f(x,t;x_0)}{\partial t} = \frac{\alpha}{2}\frac{\partial^2 f(x,t;x_0)}{\partial x^2} - \beta\frac{\partial f(x,t;x_0)}{\partial x} \tag{1.10}$$

Its solution defines the conditional probability density function (PDF) $f(x,t;x_0) = P[x \leqslant X(t) < x + dx \mid X(0) = x_0]$ of the process $X(t)$, approximating the number of customers present in the queue at time t. Both processes $X(t)$ and $N(t)$ have normally distributed changes with a mean (drift velocity or mean changes in the queue size) $\beta = \lambda - \mu$, and variance (variation of the changes in the queue size)$\alpha = \sigma_A^2\lambda^3 + \sigma_B^2\mu^3 = C_A^2\lambda + C_B^2\mu$. where λ is the arrival rate and μ is the service rate, ensures the same ratio of time-growth of mean and variance of these distributions (i.e., the queue utilization, $\varrho = \frac{\lambda}{\mu}$). Since the queue size must be positive, boundary conditions such as absorbing barrier [44], reflecting barrier [40], or elementary return boundary [85] are commonly used to constraint the diffusion process to the positive x-axis. The diffusion approximation principle and its application in modelling computer networks and battery energy storage systems for battery-powered computer network devices are discussed in chapters 2,3,4,5 and 6.

The features that are in favour of the diffusion approximation are:

- Diffusion model of a single server assumes general interarrival and service time distributions going beyond Markov models.
- Network models may have any topology, hierarchical, and having any number of nodes (easily scalable).
- The results are obtained in the form of queue distributions and waiting time distributions, making it easier to analyse the QoS of paths, e.g. jitter. An alternative method, the fluid-flow approximation, frequently used in large topologies and transient analysis, delivers only mean performance parameters and introduces larger errors.
- Easy separation (decomposability) of each node within a network model - the interactions among nodes are reduced to the computation of the input flow parameters at each node.
- Model may include classes of packets, which makes it possible to model queueing systems of computer systems and networks where the packets or jobs belong to different classes of service.
- Natural ease to analyse transient states based on the solution of diffusion equations.
- The transient state model is solved step-by-stem in short time intervals with parameters specific to these intervals, making them suitable for the analysis of the transient behaviour of the performance metrics (mean queue size, mean waiting time, or blocking probability).

Since diffusion approximation uses the mean and variance (or squared coefficient of variation) of the interarrival and service times distribution, they offer a more realistic approach to modelling practical queueing systems. These parameters can be derived from historical measurements in real systems and used for theoretical analysis and evaluation of the performance of queueing systems. Despite the numerous advantages of diffusion approximation models, they still have their limitations. One of the limitations of the diffusion approximation modelling methodology is the complexity of the transient solutions. However, they are more accurate than fluid flow approximations as demonstrated in [48] by Czachóski et. al.

1.4 An overview of the content of the thesis

This doctoral dissertation contains seven chapters. The remaining six chapters are as follows:
In chapter 2, we review packet aggregation applications in access networks (IoT and 4G/5G mobile networks), optical core networks, and cloud computing data centre networks is presented. We also present analytical models for designing and evaluating the performance of packet aggregation mechanisms. We demonstrate using measured traffic from real networks to evaluate the performance of packet aggregation mechanisms in simulation and analytical models.

In chapter 3, we present the architectures of hardware and software SDN switches and model the flow matching (lookup) mechanisms used in these switches. We propose a tractable diffusion approximation for a network router's transient and steady-state behaviour. In particular, we analysed the steady-state and transient-state delay and packet loss probability as a function of traffic load and other characteristics. Using these results, we show that when SDN switches change the paths of flows frequently, the network's behaviour may often be far from its steady-state behaviour. Therefore any network optimisation conducted with the help of SDN should not be based on steady-state behaviour but rather on some metric related to the time-dependent network behaviour. We determined a load threshold beyond which a slight increase in the load results in a sharp increase in the queue size and packet delay.

Chapter 4 presents an overview of flexible routing in SDN-based networks. We extend the methodology developed in chapter 3 to the time-dependent analysis of multiple SDN switches using diffusion approximations, which are very convenient to analyse in a time-dependent regime. Thus, we compute the transient behaviour of each SDN switch after changes occur in its input traffic rate.

In chapter 5, we apply a diffusion or Brownian motion process to model the energy depletion process of a battery of an IoT device. We use the model to obtain the probability density function, mean, variance, and probability of the lifetime of an IoT device. Also, we study the influence of active power consumption, sleep time, and battery capacity on the probability density function, mean, and probability of the lifetime of an IoT device. We use numerical examples to study the influence of battery depletion attacks on the distribution of the lifetime of an IoT device. We also introduce in our model an energy threshold after which the device's battery should be replaced to ensure that the battery is not completely drained before it is replaced.

In chapter 6, we present an architecture of a green base station site. We develop Markovian and diffusion approximation models to analyse battery energy storage systems' steady-state and transient-state performance. We apply Markovian and diffusion approximation models to derive the time after which the battery energy storage system is completely discharged or fully charged. By assuming that the energy harvesting and the energy consumption processes are exponentially distributed, we compare the result obtained from the Markovian model to those from diffusion approximation models.

In chapter 7, we conclude the dissertation.

1.4.1 Evaluation of the performance of packet aggregation mechanisms

The transmission of massive amounts of small packets generated by access networks through high-speed Internet core networks to other access networks or cloud computing data centres has introduced challenges such as poor throughput, underutilisation of network resources, and higher energy consumption. Therefore, it is essential to develop strategies to deal with these challenges. One of them is to aggregate smaller packets

into a larger payload packet. These aggregated packets will share the same header, increasing throughput, improving resource utilisation, and reducing energy consumption.

The Maximum-Time (MT), the Maximum-Size (MS), and the Maximum-Time-Maximum-Size (MTMS) packet aggregation schemes are the most popular packet aggregation schemes that have been implemented in commercial network equipment. Recently, MT, MS, and MTMS packet aggregation schemes have been implemented in the programming Protocol-independent Packet Processor (P4) [27] hardware switches in [250, 169, 251, 163], to exploit its programmability, hardware speed, and flexibility. In the time-based or size-based, or hybrid (a combination of time threshold and size threshold criteria) packet aggregation scheme, when packets arrive at the network node, they are classified based on their destination, Class of Service (CoS) or Quality of Service (QoS) parameters, and queued up in the input buffer. The smaller packets stored in the input buffer are aggregated into larger ones when the number of these packets stored in the buffer is greater than or equal to a defined maximum value or when a defined time threshold is reached. The main drawback of these packet aggregation mechanism is that in low traffic applications like in the case of IoT, the maximum defined size threshold may rarely be reached, resulting in excessive delays [250], which could be mitigated by setting the maximum time threshold to be within the maximum delay tolerance. Also, the MT packet aggregation scheme may result in large variation in the aggregated packet sizes; hence, poor resource utilisation at the level of the core network [41]. A novel slot-based packet aggregation scheme was recently proposed in [127, 126]. In this mechanism, the small packets stored at the input buffers are aggregated into larger ones and scheduled for transmissions during preallocated time slots.

The performance evaluation of time-based, size-based, or hybrid packet aggregation schemes at the edge node of IP over all-optical networks have been presented in [65, 183, 159, 114, 240, 145]. The major drawback of these studies is the assumption that the interarrival times of IP packets into the aggregation buffer follows a Poisson distribution, which is far from reality, as it differs significantly from the measured interarrival times from the Center for Applied Internet Data Analysis (CAIDA) repository which we used in [55, 57]. The authors in [163] used the Poisson assumption to analyse the performance of a time and size based packet aggregation scheme for IoT traffic over Software Defined Network, but the measured interarrival times for IoT traffic reported in [224] significantly differs from Poisson distribution. Diffusion approximation-based performance evaluations models which use measured interarrival times and sizes of packets measured from real networks have been proposed in [143, 18], and we extended these studies in [144] which is present in chapter 2. Diffusion approximation is a well-established modelling approach used to study non-Markovian queueing systems in which the arrival times and service times distributions are general, which was proposed in the current form by Gelenbe in [85, 86].

The amount of traffic generated by various access networks such as Digital Subscriber Lines (DSLs), ethernet Local Area Networks (LANs), wireless LANs, mobile networks (e.g. 3G, 4G, and 5G networks), and recently, the Internet of Things (IoT) networks is increasing exponentially, and have likely increased significantly since the beginning of the year 2020. The recent increase in the amount of traffic carried over the Internet could be attributed to the global reaction to the outbreak of the COVID-19 pandemic by transferring some key services (e.g. health care consultation, education, retail, entertainment, and business and work-related activities), and the recent increase in the rate of adoption of IoT. The packet sizes generated from these access networks vary from a few bytes in IoT and wireless sensor networks to 1500 bytes in Internet Protocol (IP) networks. The transmission of massive amounts of smaller packets, broadband access net-

works and high-speed core networks introduces some challenges such as bandwidth wastage due to protocol overhead, inefficient use of network resources, and increased energy consumption. It is, therefore, essential to develop strategies to deal with the huge amounts of traffic generated by access networks, especially IoT network [39]. One of the strategies to increase bandwidth efficiency, ensure efficient use of network resources, and reduce the extra energy consumed due to the presence of huge amounts of small packets is to incorporate packet aggregation modules in some nodes of the network. Despite its benefits, packet aggregation increases the packets' delays [62] and may not be suitable for traffic belonging to real-time applications [251].

The industry 4.0 trend is transforming every industry's production capabilities, including health care, energy, agriculture, food chains, logistics, retail, transportation and manufacturing, with IoT low power connectivity as its driving force [25]. IoT devices are designed to minimise their energy consumption and hence increase their battery life. The energy consumption in IoT devices is usually lowered by reducing its computing power by using microcontrollers or microprocessors, minimising their storage capacity, reducing the amount of energy consumed during communication with the use of low power communication protocols, and implementing energy-efficient encryption schemes and security protocols. The low-power communication protocols such as Constraint Application Protocol (CoAP) [35], Message Queueing Telemetry Transport (MQTT) [22], Advanced Message Queueing Protocol (AMQP) [245], and Light Weight Machine-to-Machine (LWM2M) [202] communication protocols that have been proposed are designed to deminish energy consumption by reducing the size of the IoT packet payload. The time required to receive or transmit a packet depends largely on its size, which is directly correlated with the amount of energy required to receive or transmit the packet. It implies that the smaller the packet's size, the smaller the communication duration, the smaller the amount of energy required to receive or transmit the packet, and hence, the longer the battery lifetime. Therefore, low power communication protocols are designed to keep the packet size as small as possible. However, with the recent large scale proliferation of IoT sensor devices that generate massive amounts of relatively small packets, it is necessary to think about the various ways that packet aggregation schemes can be deployed to deal with the challenges introduced at the level of access networks, Internet core networks, and data centres.

The recent generations of mobile networks (e.g. 4G and 5G networks) are designed to support IoT devices and satisfy the requirements of various IoT applications. However, mobile networks were initially designed to support user equipment (e.g. mobile phones, tablets), generating traffic where packet sizes are larger than those from IoT devices. The transmission of IoT traffic directly over mobile networks results in the inefficient utilisation of radio resources. The amount of IoT data carried over mobile networks is expected to increase significantly and poses challenges for service providers [130]. To ensure efficient utilisation of radio resource for IoT over mobile network deployment, a packet aggregation scheme can be implemented at an intermediate node between the IoT devices and the radio access network [175]. In this case, multiple IoT packets are aggregated into a larger packet whose size is almost equivalent to the maximum packet size that can be transmitted over the mobile network to efficiently utilise the radio resources. Instead of wasting resource blocks to handle individual small IoT packets, the IoT packets are aggregated, and the aggregated IoT packets share only one resource block. Also, the sizes of the packets from the user equipment in mobile networks (e.g. 4G and 5G) are smaller than the IP packet sizes handled in its transport networks. The aggregation and multiplexing of fronthaul and backhaul traffic into IP packets before transmission in the

transport network of 5G Cloud Radio Access Network (C-RAN) was discussed in [194, 241, 109].

Similarly, the packets from access networks that are transported over optical networks are smaller than standardised optical packets. Transporting smaller packets from the access networks directly over the optical network results in bandwidth wastage due to protocol overhead and poor utilisation of network resources. Also, at every node, the optical packet is converted from the optical domain to the electrical domain to perform routing and regenerates the signals and is then converted back to the optical domain for transportation; this is the so-called electronic bottleneck [211], that increases the energy consumption. The electronic packets from the access networks are aggregated into larger optical packets to ensure efficient bandwidth and resource utilisation in the optical core networks. The optical packets are transported purely in the optical domain through the optical core network without being converted to the electrical domain. They are transported transparently from the ingress edge nodes to the egress edge nodes and boosted using optical amplifiers when the signal power falls below acceptable limits.

A review of the application of packet aggregation in computer and telecommunication networks, from the access networks (e.g. IoT, wireless sensor, 4G/5G mobile networks), through Internet core networks to cloud data centre networks is presented in chapter 2. In the next chapter we present a detailed review of the recent application of packet aggregation in IoT and wireless sensor networks, IoT over SDN-based networks, IoT over mobile networks, in 5G radio access networks (C-RAN), IP over all-optical networks, and cloud computing data centre networks.

1.4.2 Evaluation of the performance of a Software Defined Networking Switch

In traditional networks, the routing protocols are proprietary and rigid, the routers are configured manually, and each router is responsible for both making routing decisions and traffic forwarding. The manual configurations of distributed proprietary network devices is very a cumbersome and error-prone process that can underutilise network resources [258]. These shortcomings of the current traditional networks have been addressed by introducing Software Defined Networking (SDN).

SDN is a dynamic, adaptable, and manageable paradigm that facilitates innovations in computer networks [253], together with the prototyping and deployment of flexible routing mechanisms [78]. An SDN network consists of the data plane, the control plane, and the application plane. The data plane consists of SDN switches or forwarding elements connected to the control plane through the OpenFlow protocol in the southbound interface. The control plane consists of network controllers, each of which is connected to a set of SDN switches. The controller is connected to the application servers in the application plane through the northbound interface. In classical networks, the routing protocols are proprietary and rigid, the routers are configured manually, and each router is responsible for both making routing decisions and forwarding data traffic. However, SDN offers a programmable architecture where routing decisions are moved from the data plane routers to centralised controllers. Therefore, instead of having a network of routers that perform both routing and forwarding of data plane traffic, we have simple SDN switches or forwarding devices that forward the data traffic depending on the controller's flow forwarding rules and collect network information and send it to the controller to optimise the routing decisions.

The stochastic nature of the interarrival times of packets into the input buffers in an SDN switch and the non-deterministic processing times of packets, results in the formation of queues at the input buffers,

which introduces significant delays and packet losses. Recent attempts have been made to develop performance evaluation models for an SDN switch. The authors in [170, 16, 178, 228] applied queueing models to evaluate the performance of an SDN switch. The authors evaluated that performance of the SDN switches under steady-state conditions. Thus, a major challenge in the performance evaluation of SDN switches is the analysis of their transient behaviour, when traffic rates vary under the effect of the SDN controllers' decisions and of the time-varying traffic peaks that travel through the network due to the intermittent or periodic decisions that are notified to the switches by the SDN controllers. However, the transient analysis of queueing networks that are usually used to model the performance of networks is particularly difficult, and the discrete event simulation of such systems can be very time-consuming due to the large number of samples needed to achieve a reasonable level of statistical accuracy.

The majority of the studies assumed that the interarrival process of packets into the buffer follows a Poisson process and that service times are exponentially distributed. However, packet interarrival and service times distributions do not precisely follow the usual "Poisson and exponential" assumptions, leading to computationally efficient results concerning the system's transient behaviour of the SDN switch. Therefore, to meet the need for a time-dependent analysis, diffusion approximation models for the performance modelling of a router or switch in an SDN network are proposed in this chapter. The diffusion approximation offers two important advantages: firstly, the packet arrival and service times distributions can be time-varying, and these models do not depend on the usual "exponential and Poisson" assumptions regarding service epochs in the queues and the arrival processes of packets. Additionally, the diffusion model only requires the first two moments of the interarrival and service times so that relatively realistic parameters can be based on measured traffic data, and it provides numerical results which are difficult to obtain with other techniques [24].

1.4.3 Evaluation of the performance of a network of Software Defined Networking switches

Recent studies have analysed SDN networks to optimise steady-state performance using queueing theory models of various complexity. Some of them represent the node as a single station, e.g. $M/Geo/1$ [228], $G/I/M/K$ [223], or M/G/1 station [260]. Others distinguish data node and controller, representing both as a Jackson network [170] or introducing high (for packets coming from the controller), and low priority queues in the data switch, including more complex input flows models based on Markov Modulated Poisson Process, [178]. Some studies are based on network calculus [19, 30]. All these models present only the steady-state analysis of SDN nodes.

The frequent changing of packet forwarding rules makes the analysis of the transient behaviour of the performance parameters of the switch, such as the delay, jitter, and packet loss probability. The usual tools for network performance analysis of computer networks are not well adapted for the performance analysis of SDN networks with frequently changing packet forwarding rules. It is because the transient analysis of queueing network models is particularly difficult. The discrete event simulation of the transient behaviour of networks is very time-consuming due to the large number of randomised repetitions needed to achieve a reasonable level of statistical accuracy.

Unfortunately, conventional queueing network models are difficult to use in the transient regime because of the computational burden associated with their analysis; indeed, analysing transients, even in a simple

single-server system with Poisson arrivals and exponential service times lead to the use of Bessel function expansions, and interconnected systems are quite hard to analyse in the transient case [192, 193, 231]. The analytical solution is known only in the case of single queues with Poisson input stream and exponentially distributed service times; see [36] for infinite and [235, 14] for finite queues. The models use Markov chains and solve Chapman–Kolmogorov equations (first-order linear differential equations), defining the state probabilities of n customers present in the system at time t. The equations are solved analytically in the Laplace domain, and then the original functions in the time domain are found. Even in these relatively simple models, the solutions are quite complex—e.g., in the case of the infinite queue, the state probabilities are given in the form of the infinite series of modified Bessel functions of the first type and various order; the Bessel functions are themselves the infinite series. Some simplifications were proposed—e.g., the generating function of the distribution in the Laplace domain may be replaced by expressions with simpler original functions in the time domain [141], or Bessel functions may be replaced by easier-to-compute functions [124].

These analytical results do not fit well with the problem of modelling computer networks, where the streams incoming to switches are not Poisson and the sizes of packets—and therefore also the service times—are not exponentially distributed. We may introduce to Markov models interarrival and service times distributions composed of exponentially distributed phases—e.g., Cox distributions or hyper-Erlang distributions; the state definition is extended to include the current phase. There are numerous tools—e.g., [205]—that can match a phase-type distribution to any empirical histogram. However, the initial number of states should be multiplied—in this case, by the number of phases. This substantially increases the number of equations to be solved numerically. The solution is usually obtained using an existing tool—e.g., [2, 152]. We have applied this approach in modelling the transient states of an IP router, [49]; to represent the service time distribution, we needed a hyper-Erlang distribution with three parallel Erlang distributions with 21, 1387, and 2 phases. This could be conducted in the case of a single queue, as we are able to solve systems of millions of equations numerically, but it is hard to use this approach in modelling a network of switches.

Although the assumption that the interarrival times of packets into the buffers of SDN switches and that the processing times of packets are exponentially distributed makes the analysis of SDN networks tractable, these assumptions deviate from reality. The typical method is to use either fluid flow approximation [181] or diffusion approximation [133, 47]. The fluid flow approximation is much simpler, as it considers only the time-dependent mean values of flows, queues, and delays. However, its errors are much larger than those of diffusion approximation; see a comparison in [60]. On the other hand, discrete event simulations of transients require many hundreds of independent repetitions of simulation runs to achieve sufficient statistical accuracy, making the computation times of such simulations prohibitive [52]. More details on fluid flow approximation and diffusion approximation and their application in the analysis of the performance of traditional computer networks was presented in [190]

The accuracy of diffusion approximations has been validated in industry-based research over many decades [207, 166, 172, 99], including for patented techniques [173], and also validated in many academic papers [255, 132, 137]. Their advantage includes a more accurate representation of interarrival and service processes, the ease of obtaining delay predictions from traffic measurements, and much faster numerics for transients than discrete queueing models [24] or simulations.

1.4.4 Modelling of a battery of a computer system without energy harvesting

Energy modelling for battery-powered IoT devices

The Internet of Things (IoT) is a network of low-power computing devices (IoT devices) that are equipped with sensors to collect data from the physical environments and send it to a remote server (e.g., a fog computing or cloud computing server). The remote server performs some processing, and the results are either sent to the operator for decision-making or sent back to the low-power computing devices attached to an actuator capable of manipulating physical systems. The IoT devices are often powered by batteries [81, 212] with limited energy capacity. In the design and deployment of IoT devices and networks, the choice of the communication protocol, the amount of processing that can be performed, and the security mechanism that is implemented are constrained by the limited amount of energy present in the battery, which determines the performance and the lifetime of IoT devices and networks. The adoption of low-power communication protocols in IoT networks to reduce the energy drawn from the battery to power IoT devices (to prolong battery life) makes them unsuitable for applications that require the generation and transmission of multimedia data. Also, cyber security attacks can be conducted to quickly drain the battery of the IoT device and shorten the lifetime of the device [96].

Billions of wireless sensors devices are expected to be connected to the Internet through IoT access networks, and small, cost-effective batteries will power a majority of these sensors with limited energy capacity [117]. Recent advances in low-cost and low-power IoT technologies have enabled cost-effective, energy-efficient, data-driven, and flexible automation of cyber-physical systems. However, when hundreds of billions of IoT devices are connected to the Internet, the amount of energy required to power these systems and related fractures will be enormous. To ensure that IoT devices are small and cheap for commercial deployment in large numbers, they are generally designed to have limited battery capacity, low computational power, limited memory, and use low-power communication protocols [204, 120]. Therefore, when choosing batteries for IoT devices, it is essential to consider constraints such as cost, size, and capacity (energy rating of the battery in Wh, which determines the lifetime of the IoT device).

Also, the limited computational, communication, and energy resources in IoT devices, make it challenging to deploy complex security mechanisms and to implement traditional cryptographic algorithms as they require non-constant execution time [177]. Some IoT manufacturers and service providers design or deploy IoT devices without appropriate security mechanisms, making them vulnerable to cyber-attacks. Some of the possible cyberattacks that could be launched against an IoT device include identity theft, eavesdropping, man-in-the-middle attacks, and energy depletion attacks. One of the components of an IoT device that malicious attackers often target is the small, limited-capacity battery used to power the IoT device [112]. This kind of attack is often called an energy depletion attack and is analysed in this paper. Therefore, the key constraint in designing and deploying IoT devices and networks is to establish a reasonable tradeoff between power consumption, throughput, security, coverage area, battery lifetime, and financial cost.

The design specifications of an IoT device and those of the battery used to power the IoT device should be chosen in such a way as to ensure a longer lifetime; while ensuring acceptable QoS and security of the device. The lifetime of an IoT device is the time required to deplete the battery's energy completely. Mathematical modelling frameworks have been proposed to establish the relationship between the device and battery parameters with the lifetime of an IoT device. The authors in [215, 214] modelled the energy

depletion process of the battery of an IoT device (without energy harvesting) as a pure death Markovian process (since energy is drawn from the battery, and the battery is not recharged). They used their model to the impact of energy depletion attacks on the lifetime of an IoT device. The limitation of their model is limited by the assumption that the energy consumption process follows an exponential distribution, and they address this limitation by proposing a model in which the energy consumption can follow any process (and could even be deterministic).

Energy modelling for battery-powered UAVs

The recent advances in Unmanned Arial Vehicle (UAV) technologies (e.g., data collection, data storage, data processing, data transmission, data security, delivery of loads) [217] have increased their adoption rate for military and commercial applications. The fast adoption rate is partly driven by the decrease in the cost of drones and granting licenses to commercial service providers and hobbyists. Some of the industries that are being transformed through the application of drones include agriculture, environmental management, supply chains, law enforcement, surveillance, and photography [138, 139, 140]. At the beginning of the COVID-19 pandemic, drones were used for deliveries [75] and to enforce restriction rules (social distancing, no mass gatherings in open public spaces) designed to slow down the transmission of the virus.

It is essential to ensure that the batteries selected to supply an UAV is able to power it for the entire duration of its mission. The duration of a done's mission depends on the amount of energy required to perform some manoeuvering actions (takeoff, level flight, hovering, and landing) [6] and the energy required to power the ICT modules in the drone. The energy required to drive the drone depends on the manoeuvering action taken, the drone's speed, payload, and the wind. Although the amount of energy required to drive the drone is often far greater than the energy required to power the ICT modules, the influence of ICT energy consumption on the duration of the drone's mission could become significant (especially for drones that draw small amount of energy for flight but perform complex ICT functionalities). Also, cyber security attacks designed to increase the amount of transmission or computations executed by the drone and deplete its battery faster could rapidly deplete the energy stored in the battery.

Most drones are powered by batteries, making energy a critical resource that must be optimised during the mission of the drone. One of the responsibilities of a drone pilot is to ensure that the drone returns with enough energy in the battery that is sufficient for a safe landing after its mission. If the drone's battery is completely depleted during its mission, it will crash to the ground and could damage the drone or result in a lawsuit if it damages properties or causes harm to human life. Even the most experienced drone pilots sometimes encounter drone crashes due to battery depletion. It is difficult to estimate how much time is required to completely deplete the energy stored in the battery during flight because a complex interaction of multiple factors influences the battery energy depletion process in drones. These factors include weather (e.g., wind, temperature), drone speed, the ICT-related functionalities performed by the drone, and the weight of the drone and the load carried by the drone (if any). The energy stored in the battery could also be rapidly depleted due to cyber attacks, which are designed to induce the ICT systems of the drone to draw more energy from the drone unnecessarily. Some drones are configured to return to the operator at predefined battery levels and to land at 15% battery level automatically. Therefore, the drone operator should ensure the safe landing of the drone while preventing any harm to human lives.

To adapt a UAV to perform its functionalities for a given application, advanced on-board information and communication technology significantly increase its energy needs during a mission [257] because of the computationally intensive visual information processing before transmission or storage [45]. Using multiple cooperating UAVs to conduct a mission [118] also increases the computational burden and energy consumption of each UAV, in order to coordinate movements and create a consistent view of the events or scenes that are monitored [102], also leading to additional on-board energy consumption from communications [103], and more on-board software [195]. On-board computing and communication equipment cannot easily be put to sleep to save energy, to avoid compromising the real-time needs which would be impaired by "wake-up" delays [94].

Since careful usage of the UAVs energy budget is needed to achieve the best possible mission output from the battery storage and possible other on-board energy sources such as phtovoltaic and fuel cells, the optimisation of the power consumption of an UAV via its speed was studied in [239, 21, 38]. However, the energy used to perform functions such as encryption, compression of multimedia data, and communications is significant. In addition, the interplay of multiple factors influencing energy consumption implies that the energy drawn from the battery is not deterministic. Models that adequately capture the influence of some of the design parameters of drones that determine the flight duration of a UAV are important.

1.4.5 Modelling of a battery energy storage system of a computer system with energy harvesting

The adoption of renewable energy is driven by decarbonisation, digitisation, and grid decentralisation trends. Decarbonisation is the progressive reduction in carbon emission from energy generation and energy consumption systems. A decentralised energy system is one in which the energy production facilities are located closer to the consumers or energy consumption facilities [42]. The simplicity and the low cost of installing and maintaining renewable energy sources make them suitable for decentralised energy systems. They are increasingly being adopted as an energy source to power mobile network base stations located in challenging environments where conventional power sources are not available or expensive to operate.

Energy storage Systems (ESS) are systems that convert energy generated from various energy sources into energy forms that can be stored and used in the future. Battery energy storage systems are rechargeable battery systems that store electrical energy from renewable or traditional energy grids using electrochemical solutions for use at a later time. Other types of energy storage systems include pumped hydro, compressed air storage, and mechanical flywheels.

In an IoT network with hundreds, thousands, or millions of IoT devices, having too many device energy-related failures resulting from completely drained batteries increases the complexity and maintenance cost. These abrupt failures could lead to financial losses and loss of human lives. IoT devices are sometimes deployed in areas that are not easily accessible to change or recharge their batteries. The energy limitation in IoT networks is being addressed using energy harvesting [220]. It is the process of capturing or harvesting energy from one or more renewable energy sources and converting it into electrical energy that can be used to power IoT devices [81]. Energy harvesting is a sustainable and convenient way to ensure the continuous operation of IoT devices [69] without energy-related interruptions. The energy can be harvested and stored in the battery and drawn to power the IoT device [15], or the IoT device can be powered directly by a renewable

energy source, provided it is reliable enough. The authors in [13] presented a state-of-the-art review of sustainable green strategies in IoT networks, including energy harvesting and energy-saving practices.

Recently, some of the functionality (e.g., computation, communication, storage, and security) have been shifted from the cloud computing servers to the fog/edge computing servers, located closer to the end-users or data sources. It is to ensure acceptable quality of service (QoS), especially for real-time IoT applications and to reduce the energy consumption in the Internet core networks. For IoT network deployment in remote areas (e.g., farms located in remote areas not covered by the grid), a renewable energy source could supply the fog computing node. Also, renewable energy sources have been adopted to power ICT infrastructures with the persistent global agenda to adopt decentralised energy systems to supplement existing fossil fuel-based centralised energy systems to reduce carbon emissions. A decentralised energy system is one in which the energy production facilities are located closer to the consumers or energy consumption facilities [42]. The simplicity and the low cost of installing and maintaining renewable energy sources make them suitable for decentralised energy systems.

Green energy sources (e.g. solar and wind) present some challenges due to their intermittent power output [110]. Still, energy storage systems (e.g. batteries) are used to ensure a smooth supply of energy to the consumers. The amount of energy produced by green energy sources depends on the energy conversion efficiency, geographical location, time of day, season of the year, topography, and weather conditions. Also, the energy demands of fog computing nodes vary over time, depending on the computing, storage, communication, and security processes being handled by the fog node. Therefore, the amount of energy stored in the battery at a given time and the battery lifetime can be predicted using stochastic modelling techniques such as Markov models (e.g. in [46, 219, 17]) and diffusion approximation models (e.g. [4, 257]). If the battery's charging rate is far less than its discharging rate, then the battery could be completely discharged. In that case, the device supplied by the batter will be shut down, leading to a service outage. The authors in [43] proposed a reinforcement learning approach for battery management in a green fog computing node.

For UAVs that require long flight duration, constant manoeuvring, and carry heavy loads (like those designed for transportation), energy harvesting sources can be used to harvest energy from the environment to replenish the energy drawn from its battery during flight. The energy harvesting mechanisms are incorporated into drones to harvest energy from the environment and store it in the battery. Some of the energy harvesting mechanisms incorporated into drones include solar [206], and dynamic soaring [26]. Energy harvesting is environmentally dependent, and therefore, it is possible to use hybrid energy sources such as solar energy and hydrogen energy to ensure a continuous supply of energy for applications that require long flights [157]. Energy is drawn from the battery to drive the drone and to power its ICT systems. The optimisation of the power consumption of an UAV by choice of its speed was studied in [239]. However, the amount of energy used to perform ICT-related functions such as encryption, compression of multimedia data, and communication is becoming significant with the increasing complexity of electronic systems installed on the drone, which justify the need to consider ICT-related energy optimisation in drones during flight. Any energy harvesting mechanism used on-board is also influenced by the environment. Therefore, both the energy generation and consumption processes need to be modelled as stochastic processes.

The use of renewable energy system to power base stations in mobile cellular networks have been widely adopted as a promising solution to reduce the carbon footprint and operational expenditures by mobile operators. Mobile communications have contributed enormously to the social and economic development

of every society throughout the world, including the less developed or the remote parts of the world [92]. The authors in [92] proposed an energy packet-based Markovian model to analyse battery energy storage systems required to store energy in green base station sites supplied by renewable energy sources.

The interplay of multiple factors influencing energy consumption implies that the energy generated from the battery is not deterministic. The environment also influences the energy harvesting mechanism. Therefore, both processes are modelled as stochastic processes. Markovian stochastic models have been applied to model the changes in the energy content of a battery, e.g. in [46, 219, 17] However, the energy Poisson assumption in the arrival of energy packets into the battery and the removal of energy packets from the battery may deviate from reality.

This is why we apply here a diffusion model where the interarrival times and interdeparture times may follow any distribution, as already proposed in [4, 257]. An overview of the stochastic modelling of battery energy storage systems is presented in chapter 5.

Chapter 2

Design and Performance Modelling of Packet Aggregation Mechanisms and their Applications

The transmission of massive amounts of small packets generated by access networks through high-speed Internet core networks to other access networks or cloud computing data centres has introduced several challenges such as poor throughput, underutilisation of network resources, and higher energy consumption. Therefore, it is essential to develop strategies to deal with these challenges. One of them is to aggregate smaller packets into a larger payload packet, and these groups of aggregated packets will share the same header, hence increasing throughput, improved resource utilisation, and reduction in energy consumption.

In this chapter, a review of packet aggregation applications in access networks (IoT and 4G/5G mobile networks), optical core networks, and cloud computing data centre networks is presented. We also analytical models for the design and evaluation of the performance of packet aggregation mechanisms are presented. We demonstrate the use of measured traffic from real networks to evaluate the performance of packet aggregation mechanisms using simulation and anlytical models is demonstrated. The use of diffusion approximation allows us to consider time-dependent queueing models with general interarrival and service time distributions. Therefore these models are more general than others presented till now. This chapter is adapted from [145, 143, 18, 144, 59], which are the article that author published on the performance evaluation of packet aggregation algorithms.

2.1 Applications of packet aggregation

This section presents a review of the recent application of packet aggregation in computer and telecommunication networks, from the access networks (e.g. IoT, wireless sensor, 4G/5G mobile networks), through Internet core networks to cloud data centre networks. We present in Fig. 1.1 an abstract architecture of a network in which the IoT and wireless sensor networks, the cellular networks (3G/4G/5G), the Internet Service Provider (ISP) access networks, enterprise access networks, and data centre networks are connected by a high-speed internet core network. Many papers have been published on the application of packet aggregation to improve throughput efficiency, improve resource utilization, and reduce energy consumption in

computer networks, but this subsection is limited to the review of recent works on packet aggregation.

2.1.1 Packet aggregation in IoT and wireless sensor networks

A simplified architecture for IoT applications consists of the IoT layer, the fog layer (if fog computing is supported), and the cloud layer. The IoT sensors measure relevant data from the environment, securely transfer the data through an access point to the fog nodes for lightweight processing or the cloud data centre for advanced data analytics. The feedback from the data analytics platforms (fog and data centre applications) is sent back to perform appropriate actions to control some IoT actuators or to provide information to users for decision making. Well-known low power, reliable wireless access communication technologies such as LoRaWAN [74], Sigfox [155] have been widely adopted for the communication between the sensor devices and the access point. The sizes of the IoT packets generated are very small and result in spectral inefficiency, poor resource utilisation, and high energy consumption. Therefore, the IoT packets can be aggregated at the level of the access point or fog node (because they have higher computing resources than the IoT devices and do have energy limitations as they are continuously powered by a reliable energy source) before being transmitted through the Internet core network to the cloud computing data centres.

The authors in [136] proposed a packet aggregation scheme for the aggregation of IoT packets in wide area networks. In their proposed scheme, when packets arrive at the access point, they are classified based on their destination, and packets that belong to non-real-time applications are aggregated, but those that belong to delay-sensitive real-time applications are transmitted immediately to their destination. A "flag" field is added to the packets. It is checked by every node to determine whether the packet should be aggregated (if it does not belong to a real-time application) or not. They assumed that not every node in the network should possess the ability to aggregate and disaggregate packets, which should be considered when choosing the next node during packet forwarding. A similar dynamic mechanism for the aggregation of packets in IoT and Low power and Lossy Networks (LLNs) to decrease energy consumption and increase battery life was proposed in [116]. In the proposed scheme, each node is equipped with a Learning Automata [208], which grants permission to the node to aggregate small packets that need to be aggregated into a larger one, and denies aggregation permission for some packets that need to be transmitted immediately.

The packet aggregation and disaggregation process introduces an additional delay to both real-time and non-real-time packets. In the packet aggregation scheme proposed in [165] to reduce delays and energy consumption in body sensor networks, the IoT packets are stored in local buffers and aggregated into larger ones before forwarding them. When the number of packets stored in the buffer is greater or equal to the packet aggregation threshold or when the first packet's waiting time is greater than the waiting time aggregation threshold, the stored packets are aggregated into a larger one and sent to the forwarder node. The forwarder node is selected based on the expected queue size and waiting time to ensure the acceptable quality of service.

In the packet aggregation schemes proposed in [136, 116, 165], the authors classified the packets into packets that belong to real-time applications and those that belong to non-real-time applications. However, the authors did not clarify whether the packets that belong to real-time applications and those that belong to non-real-time applications share the same buffer or not. If they share the same buffer, then the QoS experienced by packets that belong to real-time applications may be degraded as the packet aggregation

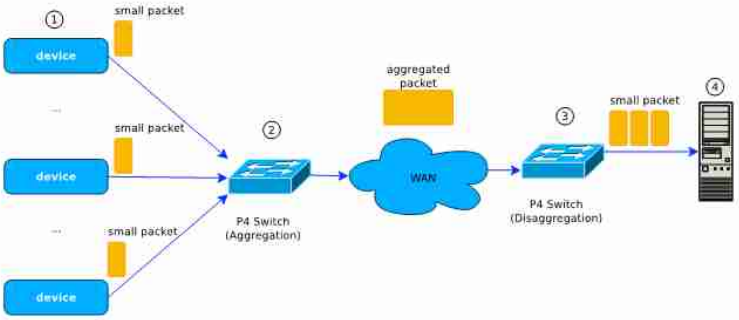

Figure 2.1: Architecture of a demonstration of IoT packet aggregation and disagreggation using P4 switches [250].

mechanism may introduce additional delays. The authors in [23] proposed priority-based channel access and data aggregation scheme to reduce packet delays in clustered Industrial IoT networks. When the IoT packets arrive at the access point or fog node, they are classified into high priority (that is, packets that belong to real-time applications) and low priority packets (those that belong to non-real-time applications). The high priority packets are stored in a high priority queue and transmitted without aggregation, but the low priority packets are stored in a low priority queue and aggregated into larger ones before transmission.

2.1.2 Packet aggregation in IoT over SDN-based network networks

The packet aggregation and disaggregation mechanisms implemented in most traditional network devices (servers, switches and routers) are performed by the Central Processing Unit (CPU), which execute software programs to perform packet aggregation and packet disaggregation operations in the control plane. The throughput rates achieved with CPU-based packet aggregation and disaggregation operations are lower than those obtained by using hardware ASIC switches. It is because software execution speeds are lower than those of hardware switching ASICs [251]. The authors in [250, 169, 251, 163] have demonstrated an SDN approach in which the data plane pipelines of P4 hardware switches can be programmed to perform packet aggregation and disaggregation operations at high speed. The authors conducted experiments using IoT traffic, which makes P4 SDN-based data plane switches a good choice for deploying networks that carry IoT traffic.

The recent introduction of the software defined networking paradigm has given network operators and service providers programmatic control over the networking equipment in their networks' data planes. The development of P4 technology has provided hardware leverage for manufacturers of network equipment and network operators. The P4 language is a high-level programming language used to program the data plane of hardware switches based on the SDN networking paradigm. It is used to program hardware switches similarly to Verilog, and VHDL (Very high-speed Hardware Description Language) hardware description languages are used to program FPGA(Field Programmable Gate Arrays)-based hardware. However, the P4 language does not require a detailed understanding of the underlying electronic design as in the case with VHDL, and Verilog [169]. Therefore, the ability to programmatically manipulate packet payload in P4 switches enables flexible and faster (higher throughputs) packet aggregation and disaggregation operations

than those obtained when using traditional (non-programable) network devices.

The implementation of packet aggregation of IoT traffic on P4 switches was first proposed in [163] as shown in Figure 2.1. The authors discussed its feasibility, presented performance evaluations models to evaluate the performance of packet aggregation and disaggregation operations. The first design and implementation of packet aggregation and disaggregation in P4 switched was presented in [251]. In their implementation, the packet aggregation and disaggregation operations are performed purely in the hardware switching ASIC pipelines, and the authors achieved a 100 Gbps packet aggregation throughput which is the highest so far reported. The limitation of these studies is that their implementation could only aggregate fixed-sized IoT packets, but we may have traffic from different sources with different packet sizes in a real network. The authors addressed this limitation in their recent implementation in [250], where they designed and implemented packet aggregation and packet disaggregation operations in P4 SDN data plane switches that enable the aggregation and disaggregation of IoT packets of different payload sizes with a throughput of 100 Gbps. Recently, the authors in [169] proposed a P4 implementation of an IoT protocol designed to ensure an adaptable aggregation of packets to reduce the number of packets sent over the network with an acceptable delay.

2.1.3 Packet aggregation in IoT over mobile networks (4G/5G)

With the widespread adoption of Low Power Wide Area (LPWA) technologies to enable long-range communication for IoT devices, NB-IoT (NarrowBand-IoT) [184] has been introduced and can coexist with existing mobile networks (e.g. 2G/3G/4G/5G). Existing mobile networks may be overwhelmed in the future by the massive growth in IoT traffic [210] when hundreds of billions of IoT devices are connected to the Internet via mobile networks. Allocating radio resources for each IoT packet will result in spectral inefficiency and inefficient radio resources utilisation. Multiples small IoT packets can be aggregated into larger ones so that a group of aggregated IoT packets can share the same radio resource. It will improve spectral efficiency and radio resource utilisation.

The authors in [210] proposed introducing a Relay Node (RN) between the IoT devices and the 5G radio access network. It receives small IoT packets, stores them in a buffer and then aggregates them into larger packets that are transmitted to the cellular radio access network through a wireless connection. The introduction of the Relay node to aggregate the small IoT packets into larger ones improve spectral efficiency and radio resource utilisation, but it also introduces a significant delay. In this case, packets from real-time IoT applications should not be aggregated. A similar approach was proposed in [131] for LTE-A(Long Term Evolution Advanced) mobile network in which an intermediate node is placed between the IoT devices and the 4G radio access network. It receives small IoT packets, store them temporally and then aggregate them into a larger packet when the queue size of packets in the buffer reaches a defined maximum threshold or when a defined waiting time threshold is reached. The authors concluded that their approach guarantees enhanced spectral efficiency, increase the capacity of the radio access network, and ensure an acceptable delay. The drawback of the approach proposed in [210] and [131] is that both real-time and non-real-time traffic share the same buffer, which causes real-time traffic to suffer from excessive delays due to the packet aggregation. Priority queueing should be considered. Real-time packets should be placed in high priority queues and relayed directly to the radio access network without aggregation, or traffic from real-time IoT

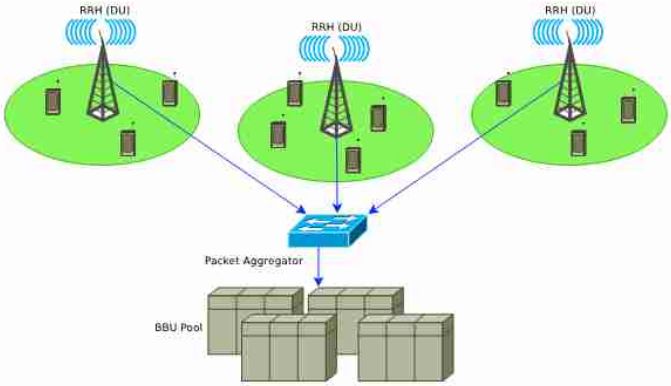

Figure 2.2: A Cloud Radio Access Network (C-RAN) architecture with packet aggregation at the fronthaul network [194].

applications should be transmitted directly to the radio access network.

2.1.4 Packet aggregation in 5G Radio Access Networks (C-RAN)

A 5G Cloud Radio Access Network (C-RAN) consists of a set of Remote Radio Heads (RRHs), a fronthaul network and a pool of shared Broadband Units (BBUs) as shown Fig. 2.2. The C-RAN paradigm is based on splitting functionalities by shifting complex signal processing from the RRHs to the BBUs. It leverages on the benefits provided by Network Function Virtualization (NFV) and SDN technologies to add flexibility and adaptability to fronthaul and transport networks of 5G mobile networks [241]. The RRHs receives the radio signals, digitise them and then transmit them to a pool of BBUs through the fronthaul network. The packets that belong to the flows coming from the RRHs are smaller than those in the backhaul network (packets from fronthaul networks are smaller than IP packets in the backhaul networks). An ethernet switch is deployed to aggregate packet flows from the fronthaul network and then multiplex them with those from the backhaul networks and transported through optical links to a pool of BBUs.

The authors in [194] proposed a C-RAN architecture shown in Fig. 2.2 in which an ethernet switch connected to the RRHs aggregates fronthaul traffic and forward the aggregated traffic to the cloud. The authors in [241] discussed the problem of multiplexing and aggregating fronthaul and backhaul traffic on C-RAN optical ethernet link. A strategy to aggregate fronthaul packet frames to improve throughput efficiency of the transport network of a 5G cloud radio access network was discussed in [109]. The analysis of the delay introduced by packet aggregation in 5G C-RAN has not been discussed so far. It will require the modelling of the packet aggregation process and considering the characteristics of 5G traffic and packet sizes.

2.1.5 Packet aggregation in IP over all-optical networks

The continuous rapid growth in the traffic generated by access networks and transported over long haul transport networks have led to the development and deployment of high-speed all-optical transport core networks. Transporting individual small packets from the access network over the optical network will result in poor throughput. Therefore, aggregating small packets from the access network into larger ones which are converted into optical packets and transported transparently across the optical transport networks.

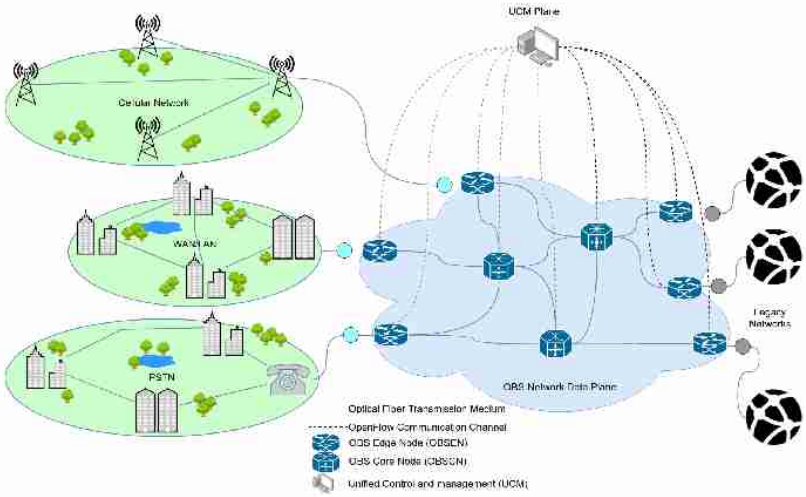

Figure 2.3: An architecture of an OBS network.

Due to the challenges in developing optical switches with optical memory, Optical Burst Switching (OBS) networks architectures (e.g. see Fig. 2.3) have been adopted as networking solution for optical networks.

In an OBS network, packets from the source access networks are aggregated into larger ones at the ingress node. The aggregated packet is converted from the electrical domain to the optical domain to form an optical packet and then transported across the optical network in the optical domain. At the aggress edge node, the optical packet is converted from the optical to the electrical domain. The larger packet is then disaggregated into smaller packets that are delivered to the destination access networks. Packet aggregation in optical networks have been discussed in [65, 164, 183, 114, 159, 182, 240, 146].

2.1.6 Packet aggregation in Cloud computing data centre networks

Cloud computing is a well-known dynamic, cost-effective and scalable computing paradigm that enables on-demand remote access of computing resources such as software, infrastructure, and platform over the internet. The large scale adoption of cloud computing is due to the introduction of virtualization technology which makes cloud computing scalable. Virtualization refers to the hardware or software methods that enable the partitioning of a physical machine into multiple instances that run concurrently and share the underlying physical resources, and devices. A Virtual machine monitor (VMM), also called a hypervisor, is used to manage the VMs and enable them to share the underlying physical resources including the network devices [28]. Some of the tools that enable the deployment of virtualization in cloud computing data centres include KVM, UMLinux, VMware, VirtualBox and Xen. [238].

In a virtualization environment like in the Xen environment, the driver domain hosts the physical device drivers, and network virtualisation is therefore essential to provide connectivity between the driver domain and the virtual machines (VMs) as seen in Figure 2.4. The I/O channel that transfers packets between the driver domain and the virtual machines creates a bottleneck due to its poor throughput [29, 238]. Packet aggregation has been proposed as a strategy to increase the communication throughput between the driver domain and the virtual machines by 700% in [29, 28, 238]. The packets to be transfered from the driver

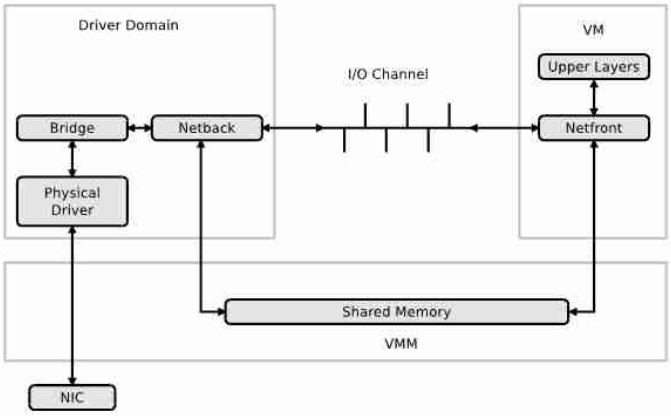

Figure 2.4: Driver domain based I/O virtualization [29].

domain to the virtual machines are classified based on their destination Mac addresses, aggregated and placed in containers which are transfered through the I/O channel to the virtual machines. At the virtual machine domain, the packets are dissggregated back into the small packets and relayed to the upper layers for processing. After processing, the packets are aggregated again, placed into containers and send back to the driver domain through the same channel.

2.2 Traffic Models: theoretical and measured traffic models

To design and evaluate the performance of packet aggregation algorithms, network equipment designers and network operators often use discrete event simulation and mathematical modelling. Mathematical models make it possible to develop mathematical relationships between the design parameters of the algorithm and its performance parameters. The limitation of most of the proposed mathematical models for packet aggregation algorithms (e.g.[65, 164, 183, 114, 159, 182, 240, 146]) is that they are based on the assumption that the distribution of the interarrival times of packets into the buffer follows a Poisson distribution. It is often assumed that the distribution of the packet sizes is fixed or exponentially distributed.

Figure 2.5 shows a comparison of theoretical distribution of the interarrival times of packet based on the Poisson arrival process assumption and the distribution of measured interarrival times from the CAIDA traffic data repository [3]. CAIDA routinely collects traces on several backbone links and make them available for research purposes. The data sets contain timestamps provided with up to nanosecond precision but truncated and stored in pcap (traffic capture) format with microsecond timescale. They also provide a dataset of the packet sizes. It can be observed in Fig. 2.5 that for the values of the interarrival time that are less than $o.002$ seconds, the distribution of the probability density of the interarrival time from the CAIDA data sets significantly differs from the theoretical one obtained using the Poisson assumption. However, for the values of the interarrival times that are greater than 0.002, the distributions from the CAIDA data sets and that from the Poisson traffic are the same. The authors in [82] performed a statistical study of the interarrival times of measure IP traffic to determine from among theoretic traffic models (such as Weibull, Pareto 2, Gamma, exponential, and lognormal). The authors realised that the best theoretical traffic model that best fits the distribution of the interarrival times is the Pareto 2 distribution. Therefore, even though assuming

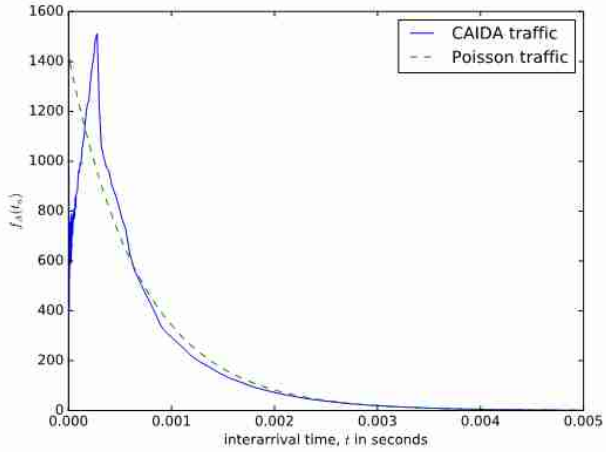

Figure 2.5: Interarrival time distribution $f_A(t_a)$, from the CAIDA measurement of the Equinix Chicago link.

that the interarrival times of packets follow a Poisson process facilitates performance analysis, the limitation of this assumption should be noted. Fig 2.6 shows the distribution of the packet sizes, with a sharp spike at about 64 bytes (for signalling packets) and another spike at about 1500 bytes (the maximum IP packet size). The average IP packet size from the presented dataset is 698 bytes. The distribution of the packet sizes is completely different from the usual assumption that it is exponentially distributed (e.g see [159]) The dataset present in Figures 2.5 and 2.6 are datasets of IP v4 packet sizes and their interarrival times from the Equinix Chicago link collected during one hour on 18 February 2016, having 22 644 654 packets belonging to 1 174 515 IPv4 flows (see [3]).

However, the coefficient of variation of the distribution of the interarrival times of the CAIDA traffic shown in figure 2.5 is 1.02, which is closer to that of a Poison distribution, which could justify the use of the Poisson assumption. The authors in [163] evaluated the performance of a packet aggregation mechanism for IoT traffic over SDN data plane made up of P4 switches. They assumed that the distribution of the interarrival time of IoT packets into the aggregation buffer follows a Poisson process but the measured distribution of the arrival times of IoT packet into a buffer in an access point is shown in Figure 2.7. The IoT traffic trace in Figure 2.7 was generated from a smart IoT environment with 28 different IoT devices such as cameras, light bulbs, motion sensors, health monitors etc. It was collected for six months [224]. It can be observed that the characteristics of IoT traffic are completely different from that of IP traffic. The IoT traffic is made of few spikes because the measurements from a group of IoT sensors for a given IoT application are updated at a predefined time simultaneously. At some time instants, there is heavy traffic from the sensors followed by prolonged silence. Therefore, analysing the performance of network devices carrying IoT traffic using the Poisson arrival assumption will yield inaccurate results.

It can be observed from Figures 2.5, 2.6 that the distribution of the interarrival times and packet sizes of traffic from real networks differ from usual Poisson assumptions. In [143, 18], the authors used diffusion approximation, which uses real traffic distributions. Diffusion approximation requires the mean and variance of the real traffic traces collected from a real network, which are used to estimate the diffusion approximation

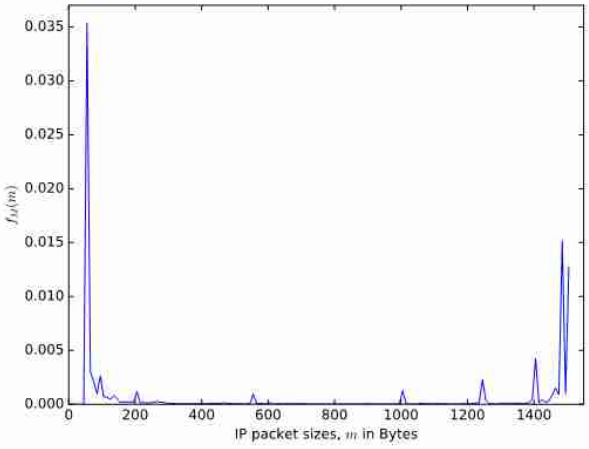

Figure 2.6: Distribution of the IP packet sizes, $f_M(m)$ from the CAIDA measurement of the Equinix Chicago link.

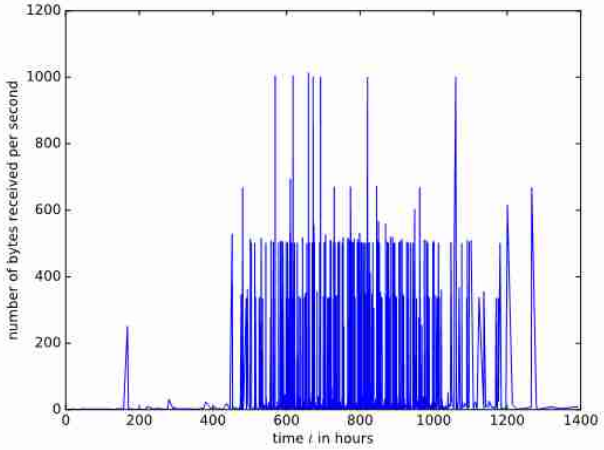

Figure 2.7: Measured IoT traffic trace from a real IoT network collected in [224].

process parameters that model the packet aggregation process.

2.3 Performance analysis of time-based and size-based packet aggregation mechanisms

In this section, we present diffusion approximation models for the time-based and size-based packet aggregation mechanism. When small packets that need to be aggregated into larger ones arrive in a network node that supports time-based or size-based packet aggregation mechanism, they are stored in an aggregation buffer and then aggregated into a larger packet when the size of the buffer content is greater than a defined maximum threshold or when a defined waiting time threshold is reached. When the first packet arrives, the byte counter and time counter are initialized. The byte counter tracks the number of bytes accumulated in the buffer, and when the amount of bytes accumulated in the buffer is greater than or equal to the defined threshold, then the content of the buffer is aggregated into a larger packet and sent to the transmission module for transmission. The timer tracks the waiting time of the first packet in the buffer to ensure that packets do not wait for too long in the buffer during a low traffic period and when it reaches a predefined waiting time threshold, the content of the buffer is aggregated and sent to the transmission module for transmission.

We propose analytical models for the evaluation of the time-based, size-based and hybrid packet aggregation mechanism. We compare the results of the analytical models with simulations. A simulation is a technique in which a virtual environment that emulates the behaviour of a physical system is created in software. A very popular type of simulation used in the performance evaluation of computer systems and networks is called discrete event simulation. We used a discrete event simulator that was programmed using the Java programming language. The simulator consists of a traffic generator created using the CAIDA traffic data sets and a buffer which represents the buffer at the input port of the node where the aggregation is performed. The MT, MS and MTMS packet aggregation mechanism is programmed in the simulator. At each simulation run, we collect the data on the aggregated packet sizes, and the interarrival times are collected and plotted together with the distributions from the analytical models using Matplotlib (a plotting library based on the python programming language). For both modelling and simulation, we use the datasets of IP v4 packet sizes and their interarrival times from the Equinix Chicago link collected during one hour on 18 February 2016, having 22 644 654 packets belonging to 1 174 515 IPv4 flows (see [3]). The traffic parameters are: $m = 698$ bytes, $\sigma_M^2 = 449361$, $\lambda = 70$ packets per second, $\sigma_A^2 = 4.9358e^{-7}$.

2.3.1 Diffusion approximation of the packet aggregation process

When packets arrive in the buffer, the number of bytes in the buffer increases and the buffer content continues to grow until the maximum defined size threshold or the waiting time threshold is reached. We represent the growth process of the content of the buffer by a diffusion approximation [85], [135], [86] process. Suppose that a diffusion approximation process $X(t)$ represent the number of bytes stored in the buffer at time t, then the dynamic changes in the number of bytes accumulated in the buffer can be modelled by diffusion equation (which is a parabolic partial differential equation describing Brownian motion of tiny particles)[44].

$$\frac{\partial f(x,t;x_0)}{\partial t} = \frac{\alpha}{2}\frac{\partial^2 f(x,t;x_0)}{\partial x^2} - \beta\frac{\partial f(x,t;x_0)}{\partial x} \tag{2.1}$$

where βdt and αdt represent the mean and variance of the changes of the diffusion process at dt. The equation defines the conditional probability density function (pdf) of the diffusion process $X(t)$

$$f(x,t;x_0)dx = P[x \leqslant X(t) < x + dx \mid X(0) = x_0].$$

The diffusion approximation applied to queueing systems is based on the assumption that the number of arrivals of customers joining the queue during a random time T has a distribution that is close to normal and does not depend on the distribution of interarrival times but only on its two first moments. The mean and variance of this normal distribution are λT and $\lambda^3\sigma_A^2 T$ where $1/\lambda$ and σ_A^2, are mean and variance of interarrival times, [85]. Here, the position x of the process $X(t)$ corresponds to the number of bytes currently in the buffer. The number of bytes received at a unit of time is a product of two independent random variables: X – the number of packets and Y – the size of packets. The mean of a product variable XY is $E(XY) = E(X)E(Y)$ and the variance is

$$\begin{aligned} Var(XY) &= E(X^2Y^2) - (E(XY))^2 \\ &= Var(X)Var(Y) + Var(X)(E(Y))^2 \\ &+ Var(Y)(E(X))^2, \end{aligned}$$

the mean number of arrived at a time unit packets is $E(X) = \lambda$ and the variance is $Var(X) = \lambda^3\sigma_A^3$, and we denote by m the mean size of a packet (in bytes) and by σ_m^2 the variance of its size, therefore the mean number of arrived at a time unit bytes is

$$\beta = \lambda m$$

and the variance of number of arrived at a time unit bytes defining α in Eq. (2.3 is

$$\alpha = \lambda^3\sigma_A^2\sigma_m^2 + \lambda^3\sigma_A^2 m^2 + \sigma_m^2\lambda^2. \tag{2.2}$$

We consider the unlimited queue; therefore, the diffusion process is limited only by a reflecting barrier at $x = 0$ (the queue is never negative).

Without any barrier, the density of the unrestricted process defined by Eq. (2.3) and started at x_0 is

$$f(x,t;x_0) = \frac{1}{\sqrt{2\pi\alpha t}} \exp\left[-\frac{(x - x_0 - \beta t)^2}{2\alpha t}\right]. \tag{2.3}$$

It is the solution of the solution of the parabolic partial differential equation (equation 2.1) the describe the dynamics of the diffusion process.

2.3.2 Modelling of the time-based packet aggregation process

For a time-based packet aggregation mechanism, when the first small packet arrive into the packet aggregation buffer, the time is activated to start tracking the waiting time of the first packet in the buffer. When the value of waiting time of the first packet in the buffer reaches a defined maximum waiting time threshold, then the content of the buffer is aggregated into a larger packet and the time is reset to zero. The growth process of the number of bytes accumulated after time t, can be approximated by a diffusion process that starts at $x = 0$ at time $t = 0$ and with an absorbing barrier at $x = N$ (the maximum defined size threshold,

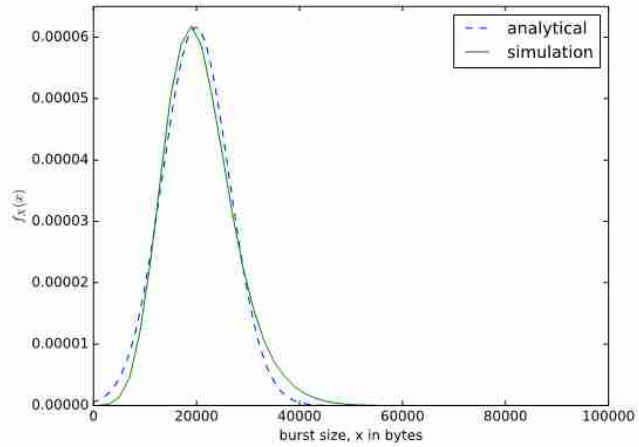

Figure 2.8: The distribution of the sizes of the aggregated packet in Bytes for a time-based packet aggregation mechanism.

and for a time-based packet aggregation mechanism, this value is set large enough that the time threshold is always reached). A diffusion process that starts at $x = 0$ and is absorbed at $x = N$ is [44]

$$\phi(x,t) = \frac{1}{\sqrt{2\pi\alpha t}}\Big\{\exp\left[-\frac{(x-\beta t)^2}{2\alpha t}\right] - \exp\left[\frac{2\beta N}{\alpha} - \frac{(x-2N-\beta t)^2}{2\alpha t}\right]\Big\}. \tag{2.4}$$

Since after a defined maximum waiting time threshold T, the content of the buffer is aggregated into a larger packet, the probability density function (PDF) of the diffusion process after time T is

$$\phi(x,T) = \frac{1}{\sqrt{2\pi\alpha T}}\Big\{\exp\left[-\frac{(x-\beta T)^2}{2\alpha T}\right] - \exp\left[\frac{2\beta N}{\alpha} - \frac{(x-2N-\beta t)^2}{2\alpha T}\right]\Big\}. \tag{2.5}$$

When the content of the buffer is aggregated into a larger packet, the timer is reset to $t = 0$ and is activated again when a new small packet arrives into the packet buffer, and the accumulation process starts again till the waiting time threshold is reached. Suppose that the first packet that arrives into the buffer to trigger the accumulation process is of random size M, with PDF $f_M(m)$, and that the number of bytes accumulated after time T is X (X is a random variable), then the actual number of the aggregated packet (the larger packet) is $X_B = X + M$, and its PDF is

$$f_{X_B} = \phi(x,T) * f_M(m) \tag{2.6}$$

and its mean and variance respectively is $\mu_{X_B} = \mu_X + \mu_m$ and $\sigma^2_{X_B} = \sigma^2_X + \sigma^2_M$

Figure 2.8 shows the distribution of the sizes of the aggregated packets in bytes. We set the size threshold or buffer size as $N = 100000$ bytes to ensure that only the maximum waiting time threshold criteria is satisfied. The value of the maximum waiting time threshold is $T = 0.02$ seconds. The value of T should be

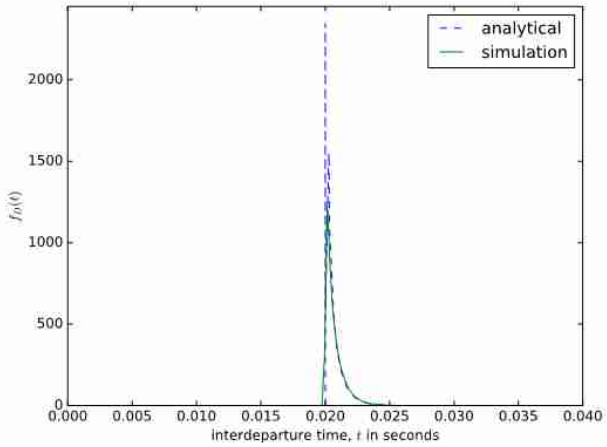

Figure 2.9: The distribution of the interdeparture times for a time-based packet aggregation mechanism.

carefully chosen to ensure that the sizes of the aggregated packets should not exit the maximum transmission unit of the target network, whose throughout efficiency is improved by the packet aggregation. For example, in aggregating voice or IoT packets to be transmitted over IP network, the value of T should ensure that the sizes of the aggregated packets should not exceed 1500 bytes. We compare the result from the analytical model of equation 2.6 with the results from the simulation. Since the small packets accumulated in the buffer are aggregated into a larger packet when the waiting time threshold is reached, regardless of the number of bytes present in the buffer, the sizes of the aggregated packets dispatched varies. The sizes of the aggregated packets depend on the rate at which the packets are arriving into the buffer, the sizes of the arriving packets and the value of the waiting time threshold, T. If the sizes of the aggregated packets are small, we have low throughput, and when they are too large, we have high throughput, but very large sizes of aggregated packets could lead to packet losses in the core network.

Since the content of the buffer is aggregated into a larger packet only when the waiting time threshold is reached, then the time from when the current threshold is reached, and the buffer content is aggregated to the moment when the next one occurs (the interdeparture time) is $t_a + T$ where t_a is the interarrival time. The accumulation time T is constant, but the interdeparture time is random because it depends on the interarrival time, since the timer is triggered only when the first packet arrives. The PDF of the interdeparture times is $f_D(t) = f_A(t_a - T)$, where $f_A(t_a)$ is the PDF of the interarrival times. Suppose that the interarrival time is exponentially distributed then, the interdepature times are exponentially distributed but shifted by T as in [159], e.g.

$$f_D(t) = \lambda e^{\lambda(t_a - T)} \tag{2.7}$$

Figure 2.9 shows the distribution of the interdeparture times for a time-based packet aggregation mechanism. The analytical results are obtained by shifting the distribution of the interarrival times from CAIDA by T. We compare the interdeparture times from the analytical model with those measured from the simulation in Figure 2.9. Since the content of the buffer is aggregated when the waiting time threshold is reached, the delay experienced by the packet is fixed, and its value is equal to the value of the waiting time threshold T.

Since when the buffer is emptied, the next accumulation process starts when the first packet arrives into the buffer, the interdepapature times distribution is similar to the distribution of the interarrival times but shifted by T as shown in Fig 2.9 (e.g., $T = 0.02$).

2.3.3 Modelling of the size-based packet aggregation process

For a size-based packet aggregation mechanism, the content of the buffer is aggregated into an aggregated (larger) packet when the maximum size threshold N is reached. When a small packet arrives, its size is compared to the difference between the maximum size threshold and the number of bytes accumulated. It is to ensure that after adding the arrived packet to the buffer, the number of bytes accumulated should not exceed the defined maximum size threshold. Therefore, the sizes of the aggregated packets are almost constant as the content of the buffer is aggregated into a larger packet when the size threshold is reached.

The times after which the maximum size threshold is reached varies since the interarrival times of packets into the buffer and the sizes of packets are random. The time from when the first packet arrives in the buffer to when the maximum size threshold is reached and the content of the buffer is aggregated into a larger packet can be considered as the first passage time of the diffusion process from $x = 0$ to $x = N$. The first passage time of the diffusion process from $x = 0$ to $x = x_0$ is [44]

$$\begin{aligned} \gamma_{0,x_0}(t) &= \lim_{x \to x_0} [\frac{\alpha}{2} \frac{\partial f(x,t;x_0)}{\partial x} - \beta f(x,t;x_0)] \\ &= \frac{x_0}{\sqrt{2\pi\alpha t^3}} \exp\left[-\frac{(x_0 - \beta t)^2}{2\alpha t}\right] . \end{aligned} \tag{2.8}$$

Therefore interdeparture time which the first passage time of the diffusion process that started from $x = 0$ and end at $x = N$ (when the maximum size threshold is reached). Hence, the PDF of the interdeparture time is

$$\begin{aligned} f_D(t) &= \lim_{x \to N} [\frac{\alpha}{2} \frac{\partial f(x,t;x_0)}{\partial x} - \beta f(x,t;x_0)] \\ &= \frac{N}{\sqrt{2\pi\alpha t^3}} \exp\left[-\frac{(N - \beta t)^2}{2\alpha t}\right] . \end{aligned} \tag{2.9}$$

Figure 2.10 shows a comparison of the interdeparture times from the diffusion approximation model and the ones obtained using a discrete-event simulator. The value of the size threshold should be carefully chosen to ensure that the delay experienced by the first packet that arrived into the buffer should not be too high. When the packets are aggregated, overhead bytes are added to them before sending the entire packet to the transmission module. Therefore, when choosing the value of N, the designer should bear in mind the value of the maximum transmission unit for the network over which the aggregated packet needs to be transmitted. The results presented in Figure 2.10 was obtained for $N = 10000$ bytes. When aggregating the voice traffic from mobile networks or smaller packets from IoT and wireless sensor networks, the value of N can be 1497 bytes, since the Ethernet maximum transmission unit is 1500 bytes (the rest is used for overhead bytes) [250]. Since the content of the buffer is emptied when the maximum size threshold is reached, regardless of the waiting time of the packets in the buffer, the waiting times of the packets in the buffer vary and are distributed as shown in Fig. 2.10. The waiting time experienced by the first packet that arrives into the buffer depends on the interarrival times of packets into the buffer, the packet sizes, and the value of the maximum size threshold N. If larger packets arrive at a faster rate (very short interarrival times), then the waiting time threshold will be reached very fast, accounting for the sharp spike in Fig. 2.10.

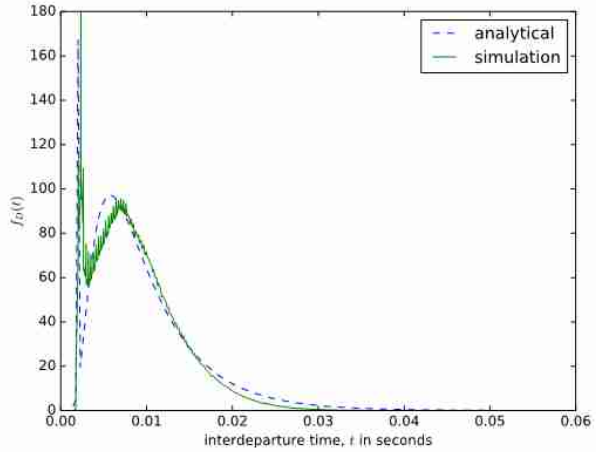

Figure 2.10: The distribution of the interdeparture times for a size-based packet aggregation mechanism.

2.3.4 Modelling of hybride packet aggregation process

For a hybrid packet aggregation mechanism, the content of the buffer is aggregated into a larger packet when either the number of bytes in the buffer is greater than or equal to the maximum size threshold or when the waiting time of the first packet in the buffer reaches the maximum waiting time threshold. Therefore, the values of N and T can be selected in such a way as to ensure any of the aggregation criteria is reached. Figure 2.7 shows a comparism of the analytical and the simulation results of the interdeparture times for a hybrid packet aggregation mechanism.

The sharp spike at the start of the interdeparture time distribution in Figure 2.11 is due to the frequent attainment of the maximum size threshold (perhaps due to fast arrivals or arrivals of packets with larger sizes). These spikes are very visible in the distribution from the simulation but analytically is approximated as the probability that the first passage time is less than or equal to the minimum filling time t_{min} e.g

$$F_D(t \leqslant t_{min}) = \int_0^{t_{min}} f_D(t)dt$$

Where $t < \frac{N-M_{max}}{\beta}$, where M_{max} is the maximum size of the arrival packets. The sharp spike at the end of the distribution is due to the frequent attainment of the maximum time threshold. This is the probability density that the diffusion process will end exactly when the deadline T is reached and is given by:

$$\int_T^{\infty} \gamma_{0,N}(t)dt = N\Big\{2 - \text{erfc}\left(\frac{N-T\beta}{\sqrt{2T\alpha}}\right) + e^{\frac{2N\beta}{\alpha}}\text{erfc}\left(\frac{N-T\beta}{\sqrt{2T\alpha}}\right)\Big\} \tag{2.10}$$

where

$$\text{erfc}(t) = 1 - \text{erf}(t), \qquad \text{erf}(t) = \frac{2}{\sqrt{\Pi}}\int_0^t e^{-\xi^2}d\xi.$$

In the case of the hybrid algorithm, the design parameters are the maximum burst size and the maximum time or deadline. Some design criteria such as the probability that the maximum burst size threshold is reached and the probability that the maximum time threshold is reached could be used to choose the design

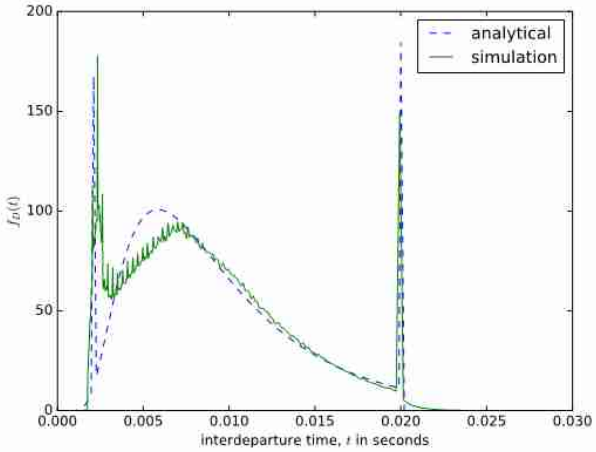

Figure 2.11: The distribution of the interdeparture times for a hybrid packet aggregation mechanism.

parameters, N and T. The probability of that time threshold will be reached, and the probability that the maximum burst size threshold will be reached respectively are

$$\begin{aligned} F_D(t \leqslant T) &= \int_0^T f_D(t)dt \\ F_{X_B}(x_B \leqslant N) &= \int_0^N f_{X_B}(x_B)dx_B \end{aligned} \tag{2.11}$$

If the value of the parameters N and T are chosen such that the probability that the time threshold is reached is 0.99, then the assembler is an MT assembler and if they are chosen such that the probability that the maximum burst size is reached is 0.99, then the assembler is an MS assembler [159]. The proposed diffusion approximation based performance analysis models for the time-based, size-based, and hybrid packet aggregation mechanism do not make any assumption about the distribution of the interarrival times of packets into the buffer and the distribution of the packet sizes as in most of the existing studies. Therefore, the distribution of the interarrival times and that of packet sizes is general; that is, any distribution can be used, including the distribution from traffic measurements such as those used in this studies.

2.4 Performance analysis of slot-based packet aggregation mechanisms

For a slot-based packet aggregation mechanism, the arriving small packets to be aggregated are stored in an over-dimensioned input buffer (a buffer with a large memory size that is sufficient to store the arriving packets). At each defined time slot Δ, small packets stored in the buffer are aggregated into a larger packet and then scheduled for transmission. It is a suitable packet aggregation mechanism in a wireless network environment in which access to the channel is shared by multiple devices, and each device is assigned a defined timeslot for transmission. A slot-based packet aggregation mechanism for enhancing VoIP performance on IEEE 802.11 wireless mesh networks was discussed in [160]. A novel slot-based packet aggregation scheme for the aggregation IP packets in Next Generation of Routers for Energy Efficiency Networks (N-GREEN)

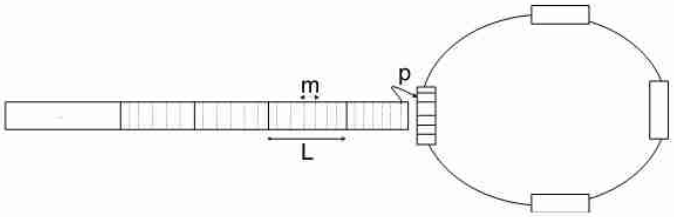

Figure 2.12: The N-GREEN packet aggregation system.

optical metro networks was proposed in [127, 126]. In this section, we present the performance evaluation models for a slot-based packet aggregation scheme in an N-GREEN network as we presented in [18].

2.4.1 The slot-based packet aggregation mechanism

In the slot-based packet aggregation mechanism that we modelled in [18], when the small packets arrive at each buffer, they are stored temporally. During a defined time slot, part of the content of the buffer (some small packets) are aggregated to a larger packet of size L and scheduled for transmission provided that the transmission channel is free. In the context of the N-GREEN slot-based packet aggregation that we modelled in [18], smaller electronic packets called Service Data Units (SDUs) from access networks (e.g. DSL, wired and wireless LANs, 2G/3G/4G/5G mobile networks, and IoT networks), are stored following a First-in-First-out (FiFo) queueing discipline. At every time slot allocated to the buffer, the SDUs are aggregated into larger packets called Packet Data Units (PDUs) that are converted into optical packets and then inserted into the optical transmission system, provided that the optical transmission channel is not occupied. Figure 2.12 illustrates the slot-based packet aggregation mechanism for an N-GREEN (Next Generation of Routers for Energy Efficiency Networks) in which at every time slot Δ, SDUs are aggregated into a PDU of size L and then inserted into any of the empty containers (of the same size) circulating in a ring so that it can be converted into an optical packet and transmitting through an N-GREEN metro network to its destination. If at each time slot the available container is empty, then the PDU is inserted into it with a probability p. Otherwise, the PDU will not be inserted and waits for the next time slot to try again.

The containers are circulating a ring, and each buffer expects the arrival of a container after every Δ seconds, and in a system with multiple buffers, a container can be empty or occupied with PDU loaded by another buffer. Therefore the PDUs are loaded or inserted by the buffers and they are converted into optical packets and then they are unloaded and transmitted by the transmission unit. Consider the following cases of aggregating the SDUs into a PDU and inserting the PDU into the containers at each time slot:

1. First case: At every time slot, the SDUs are aggregated and inserted into an empty container. If the number of bytes stored in the buffer is less than L bytes, all of its content is aggregated into a PDU and inserted into the container. However, if the number of bytes stored in the buffer is larger than L bytes, then the SDUs are aggregated into a PDU of size L bytes and inserted into the container, and the rest of the SDUs in the buffer waits for the next available container in the next time slot.

2. Second case: At every time slot, the SDUs are aggregated into a PDU and inserted into a container only when the number of bytes stored in the buffer is greater than or equal to L bytes

We modeled the packet aggregation mechanism for a single buffer and assumed that the behaviour is the same for all the buffers. We present a diffusion approximation based performance evaluation model for a slot-based packet aggregation mechanism.

2.4.2 Diffusion approximation model for a slot-based packet aggregation mechanism

As the SDUs arrive into the buffer, the number of bytes grow continuously. We represent the dynamic changes in the number of bytes stored in the buffer by a diffusion process. At each time slot, when the SDUs are aggregated into a PDU and inserted into a container, and the number of bytes in the buffer reduces instantaneously by an amount equivalent to the size of the SDU. The number of bytes present in the buffer at time t can be represented by a diffusion process whose dynamics is approximated by the parabolic partial differential equation (equation 2.1) in section 4 above. Since when the number of bytes present in the buffer is greater than L bytes, the SDUs are aggregated into a PDU of L bytes, we model the changes in the number of bytes stored in the buffer by a diffusion process jumps. We approximate the growth of the number of bytes in the buffer by a reflecting barrier at $x = 0$ (as the number of bytes stored in the buffer must be positive), whose PDF is [44]

$$f(x,t;x_0) = \frac{1}{\sqrt{2\pi\alpha t}}\left[a(t) - \exp(2\beta x/\alpha)b(t)\right]. \tag{2.12}$$

where

$$a(t) = \exp\left[-\frac{(x - x_0 - \beta t)^2}{2\alpha t}\right]$$

and

$$b(t) = \exp\left[-\frac{(-x - x_0 - \beta t)^2}{2\alpha t}\right]$$

After the PDU is inserted into the container at each time slot, the diffusion process jumps back and starts to increase again from the new initial point as the number of bytes stored in the buffer increases as the SDUs arrive into the buffer. We may also define the initial condition in a more general way, the starting point is not only at x_0, but it is at any point ξ given by a distribution $\psi(\xi)$, in this case

$$f(x,t;\psi) = \int_0^\infty f(x,t;\xi)\psi(\xi)d\xi. \tag{2.13}$$

It should be noted, when the PDU is inserted into the container, the decrease in the number of bytes in the buffer corresponds to an instantaneous jump back of the diffusion process $X(t)$. Therefore we concentrate on the diffusion description during constant intervals Δ and the definition of immediate changes of the process between these intervals. Next, we consider the two cases of aggregating SDUs into a PDU which is inserted into the container.

Case 1

At each time slot, if the number of bytes x present in the buffer is less than or equal L bytes (that is $x \leqslant L$), then all of the content of the buffer is aggregated into a PDU and inserted into the container with a probability p, otherwise, it is inserted during the next time slot. Therefore, for $x \leqslant L$, the entire content of the buffer is aggregated and inserted into the container, corresponding to a jump back of the diffusion process to $x = 0$, and continue to increase again as the queue of SDUs grows with the arrival of more SDUs into the buffer. However, if the number of bytes in the buffer is greater than L bytes (that is $x > L$), then the

SDUs are aggregated into a PDU of size L and inserted into the container, and the number of bytes in the buffer will decrease by L bytes, corresponding to a jump back of the diffusion process to $x = x - L$, and starts to increase again as the number of bytes in the buffer grows with the arrival of new SDUs. We treat the diffusion process from a given starting point till the point when it jumps back after the insertion of a PDU into the container during the next timeslot; that is we study the dynamic changes in the number of bytes in the buffer during the time interval Δ between timeslots. When the diffusion process jumps back to a given point after the insertion of the PDU into the container, that point becomes the starting the diffusion process that approximates the growth of number of bytes in the buffer.

Denote by $f^{(i)}(x, \Delta; \psi^{(i)})$ the PDF of the process during ith interval Δ. At the beginning of each interval, the time is set to zero, hence always $t \in [0, \Delta]$. The distribution of the number of bytes in the buffer at the end of each Δ, after the jump, if it occurs, defines the initial distribution of the number of bytes stored in the buffer for the next timeslot. Assume that the initial value of the process is $x_0 = 0$, i.e. the buffer is empty. At the end of the first interval, the position of the process, before a possible jump, is given by $f^{(1)}(x, \Delta; 0)$. The jump occurs with probability p giving the initial distribution for the next interval

$$\psi^{(2)}(0) = \int_m^L f^{(1)}(x, \Delta; m)dx \tag{2.14}$$

and for $\xi > 0$

$$\psi^{(2)}(\xi) = f^{(1)}(\xi + L, \Delta; 0) \tag{2.15}$$

or with probability $1 - p$ there is no jump and the new initial condition is given by the position of the process at the end of previous time-slot

$$\psi^{(2)}(\xi) = f^{(1)}(\xi, \Delta; 0). \tag{2.16}$$

Therefore, the complete initial condition for the second time slot is defined as

$$\begin{aligned} \psi^{(2)}(0) &= p \int_m^L f^{(1)}(x, \Delta; m)dx, \\ \psi^{(2)}(\xi) &= p f^{(1)}(\xi + L, \Delta; 0) \\ &+ (1 - p) f^{(1)}(\xi, \Delta; 0), \quad \xi > 0 \end{aligned} \tag{2.17}$$

and these initial conditions determine the movement of the process during the second time slot and its position at the end of it, $f^{(2)}(\xi, \Delta; \psi^{(2)})$.

In the same way for the next slots,

$$\begin{aligned} \psi^{(n+1)}(0) &= p \int_0^L f^{(n)}(x, \Delta; \psi^{(n)})dx, \\ \psi^{(n+1)}(\xi) &= p f^{(n)}(\xi + L, \Delta; \psi^{(n)}) \\ &+ (1 - p) f^{(n)}(\xi, \Delta; \psi^{(n)}), \quad \xi > 0 \end{aligned} \tag{2.18}$$

until the convergence, when $\psi^{(n+1)}(\xi) = \psi^{(n+1)}(\xi)$ and $f^{(n+1)}(x, t; \psi^{(n+1)}) \approx f^{(n)}(x, t; \psi^{(n)})$. This convergence is illustrated later in Figs. 2.13-2.4.4 for various valuess of p.

Since when the time slot occurs, and the number of bytes stored in the buffer is less than L bytes, the content of the buffer should be aggregated and inserted into the container, the sizes of the PDU may be less than L. Smaller PDU (aggregated packet) sizes result in lower aggregation throughput efficiency,

as the objective is to have more SDUs aggregated into a larger PDU and share the same header bytes during transmission, hence reducing protocol overheads. The probability of inserting a PDU of size L is $\int_L^\infty f(x,\Delta;\psi)dx$, but the probability of inserting a PDU of size $x < L$ is $f(x,\Delta;\psi)$. Therefore, the mean effective size of the packet is

$$L_{\mathit{eff}} = L\int_L^\infty f(x,\Delta;\psi)dx + \int_0^L f(x,\Delta;\psi)xdx. \tag{2.19}$$

The aggregation ratio is

$$\epsilon = \frac{L_{\mathit{eff}}}{m} \tag{2.20}$$

where m is the mean size of an SDU.

Case 2

At the occurrence of the time slot after the time interval Δ, the SDUs are aggregated into a PDU of size L bytes and inserted into the container only is the number of bytes in the buffer is greater than or equal to L bytes (that is $x \geqslant L$). The equations of Case 1 are adapted in the following way. As previously, at the end of the first interval Δ the PDF of the number of bytes stored in the buffer is $f^{(1)}(x,\Delta;0)$, and for any slot $n \geqslant 1$

$$\begin{aligned}
\psi^{(n+1)}(\xi) &= f^{(n)}(\xi,\Delta;\psi^{(n)}), \quad \xi < L, \\
\psi^{(n+1)}(\xi) &= pf^{(n)}(\xi+L,\Delta;\psi^{(n)}), \\
&+ (1-p)f^{(n)}(\xi,\Delta;\psi^{(n)}), \quad \xi \geqslant L.
\end{aligned} \tag{2.21}$$

When the steady state is reached, the initial distribution $\psi = \lim_{n\to\infty}\psi^{(n)}$ and the density of the number of bytes stored in the buffer at the end of Δ is the same e.g. $f(x,\Delta;\psi) = \psi(x)$.
The aggregation ratio is

$$\epsilon = \frac{L}{m} \tag{2.22}$$

Since the SDUs are aggregated into a PDU only when the number of bytes in the queue is greater than or equal to L bytes, the sizes of the PDUs that are converted into optical packets and transmitted are fixed. This ensures that the sizes of the PDU can be chosen by the designer such that it does not exceed the maximum transmission unit and the small PDUs that create throughput inefficiency in the transmission core network is avoided. Therefore, the designer or the network operator has control over the throughput, but in case one the throughput varies slightly as the size of the PDU can be less than L (when the time slot arrives and the number of bytes in the buffer is less than L bytes, the content of the buffer is aggregated into a PDU).

2.4.3 Queueing Delay

One of the major aims of packet aggregation is to improve throughput efficiency by reducing protocol overhead. The advantage of slot-based packet aggregation mechanism over other packet aggregation mechanism is that it produces higher throughput as the designer has control over the size of the PDU, L. That is, when the value of L is larger, more SDUs can be aggregated into a PDU and transported with just one header, which ensure that a large payload is transported with a relatively small overhead. However, high throughput is achieved at the cost of longer delays as some packets may wait for too long in the buffer. The

major parameters that influence the delay are the time interval between time slots (the time from when a buffer tries to insert the PDU into the container and the next trial), Δ and the channel availability probability (the probability of inserting the PDU into the container at a given time slot), p.

When an incoming SDU arrives into the buffer in which other SDUs that arrived earlier have been stored, it joins the queue at the tail end. We assume that all SDU (small packets) are treated the same without any prioritisation: when a SDU arrives it joins the queue at the tail end (SDUs join the queue sequentially). It should be noted that in some real implementation of this mechanism, the SDUs could be queued up based on their time sensitivity such that SDUs belong to real-time applications can be shuffled to the front (head) of the queue, aggregated, and inserted into the queue to ensure that they are transported immediately to satisfy their quality of service (QoS) requirements. However, to keep our analysis tractable, we have assumed that all SDUs have the same priority.

Suppose that an arriving SDU that arrives at time $t \in (t, \Delta)$ sees the queue distribution $f(x, t; \psi)$. With the probability

$$p_1 = \int_0^L f(x, t, \psi) dx$$

the number of bytes in the buffer is less than L. After the time interval Δ, the SDUs are aggregated into a PDU and inserted into the container with a probability p (if the container is empty), otherwise, it waits for the next time slot after the same time interval (e.g. Δ). Therefore, its waiting time will be $\Delta - t$ with probability p or $\Delta - t + \Delta$ with probability $(1-p)p$, or $\Delta - t + 2\Delta$ with probability $(1-p)^2 p$, $\ldots \Delta - t + n\Delta$ with probability $(1-p)^n p$ depending on the earliest arrival of a time slot with an empty container. This probability follows a geometric distributed, and its distribution density function is denoted as

$$\begin{aligned} f_{W1}(w, t) &= p\delta\left(w - (\Delta - t)\right) + (1-p)p\delta(w - (2\Delta - t)) \\ &\quad + (1-p)^2 p\delta(w - (3\Delta - t)) + \ldots \\ &\quad + (1-p)^n p\delta(w - ((n+1)\Delta - t)) + \ldots \end{aligned} \tag{2.23}$$

where $\delta(x)$ is Dirac delta function.

Assuming that the SDU arrival may happen at any moment t of the time slot with the same density $1/\Delta$, we determine $f_{W1}(w)$ as

$$f_{W1}(w) = \int_0^\Delta \frac{1}{\Delta} f_{W1}(w, t) dt. \tag{2.24}$$

Similarily, if the queue size is between L and $2L$ which will happen with probability

$$p_2 = \int_L^{2L} f(x, t; \psi) dx$$

then we should have two empty containers to insert two PDU in two consecutive time slots. It means that we add the delay incured by waiting for the arrival of the second empty container in second time slot to the waiting time (for the first time slot) defined above, This delay is equal to Δ with probability p if just the next container is empty, 2Δ if the next container is occupied but the one after it is empty – with probability $(1-p)p$, etc. The distribution of this additional delay $f_\Delta(w)$ is

$$\begin{aligned} f_\Delta(w) &= p\delta(w - \Delta) + (1-p)p\delta(w - 2\Delta) + \\ &\quad + (1-p)^2 p\delta(w - 3\Delta) + \ldots \\ &\quad + (1-p)^n p\delta(w - (n+1)\Delta) + \ldots \end{aligned} \tag{2.25}$$

Therefore, the waiting time for a SDU that arrives at time t and seeing the queue size between L and $2L$ is determined by the convolution

$$f_{W2}(w) = f_{W1}(w) * f_{\Delta}(w)$$

and the waiting time for the arriving SDU that sees the queue size between $2L$ and $3L$ is determined by

$$f_{W3}(w) = f_{W1}(w) * f_{\Delta}(w) * f_{\Delta}(w)$$

and the waiting time for an arriving SDU that sees the queue size between $(n-1)L$ and nL (i.e. the SDU is loaded at the n^{th} timeslot) is

$$f_{Wn}(w) = f_{W1}(w) * f_{\Delta}(w)^{*(n-1)} \tag{2.26}$$

The probabilities p_n, $n = 1, \ldots$ that an arriving SDU joins the queue and sees the queue size between $x \in [(n-1)L, nL]$ is

$$p_n = \int_{(n-1)L}^{nL} f(x, t; \psi) dx \tag{2.27}$$

Therefore, an SDU that arrives and joins a queue that is longer will wait longer, and its waiting time also depends on the probability that the circulating container that arrives to its buffer at the n^{th} timeslot is empty as shown in Figs 2.18 and 2.19 in the next section below.

2.4.4 Numerical examples

In numerical examples we use PDUs of length $L = 12.5$ KB (12500 bytes) and the time slots $\Delta = 10$ μsec at 10 Gb/sec, the same realistic parameters as considered in [127, 126]. The interarrival times have a general distribution with mean $1/\lambda$, variance σ_A^2, and the size of electronic packets is determined by a general distribution having density with mean m and variance σ_m^2. Assume $\lambda = 1$ packet/μsec, the average packet size $m = 700$ bytes, squared coefficients of variation $C_A^2 = \sigma_A^2 \lambda^2 = 1$ and $C_m^2 = \sigma_m^2/m^2 = 1$. It means that the parameters of the diffusion equation are: arrival rate $\beta = \lambda m = 0.7$ kB/μsec and $\alpha = 1.47$, as defined by Eq. (2.2)

Naturally, the variances C_A^2, C_m^2 may be different and represent any distribution, it is the advantage of diffusion approximation. Note that the squared coefficient of variation close to one does not mean necessarily that a distribution is resembling the exponential one. When analysing the distributions of packet sizes and times between packets given by CAIDA (Center for Applied Internet Data Analysis) repositories, we met distributions that are far away from exponential ones, but with $C^2 \approx 1$. The results presented are based on the PDU insertion mechanism in case 1.

Figs. 2.13-2.4.4 illustrate the convergence of the solution formulated in Eq. (2.19) for various values of p, it is visible that at each case, 25 iterations give satisfactory results.

Fig. 2.13 shows the distribution of the number of bytes in the aggregation buffer. Initially, the buffer is zero, and after the accumulation time Δ, the distribution of the number of bytes in the buffer is represented by a diffusion process that starts at $x = 0$ and grows as more packets arrive into the buffer(see the blue curve for $i = 1$) in Fig. 2.13. At the occurrence of the first timeslot, small packets in are aggregated into a larger packet of size L and inserted into the container with a probability $p = 0.25$, which shifts the diffusion process backwards by $x = L$ to a random point ξ which becomes the new initial point for the diffusion that represent the process that represents the distribution of the number of bytes in the buffer after the shift. At

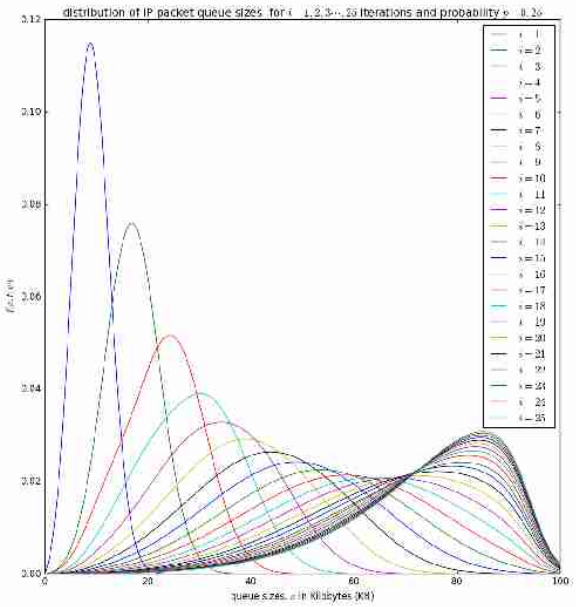

Figure 2.13: The distribution of the aggregation queue size, $f(x,t;\psi)$ if $p = 0.25$ for consecutive iterations as in Eq. (2.19) $i = 1 \ldots 25$

the occurrence of the timeslot, the PDU of size l may not be inserted into the container with a probability of $1 - p$. The distribution of the number of bytes in the buffer after the occurrence of the second timeslot can be represented by a diffusion process that starts at a random point ξ and grows as more packets arrives in the buffer (see the green cure for $i = 2$ in Fig. 2.13). After the 25 timeslots, the distribution obtained after the occurrence of the 25^{th} timeslot is similar to that obtained from the 24^{th} timeslot, which is some form of steady-state convergence behaviour of the distribution of the number of bytes in the buffer.

Figs. 2.14 and 2.15 show the distributed of the number of bytes in the buffer for various timeslots, and for $p = 0.5$ and $p = 0.75$ respectively. Similar to Fig.2.13, the distributions for higher timeslots is shifted to the right as the starting point of the diffusion process for higher timeslot may be slightly larger. After 25 timeslots, the distributions converge into a steady-state as in Fig.2.13. Unlike in Fig. 2.13, the distributions for $p = 0.5$ and $p = 0.75$ respectively are relatively shifted to the left because the probability of loading the PDU from the buffer to the container is larger.

Fig. 2.4.4 shows the distribution of the number of bytes in the buffer for $p = 1$. It can be observed that there is no significant shift of the distributions after the occurrence of various timeslots. For $p = 1$, it is certain that at the occurrence of a timeslot, SDUs are aggregated into a PDU and inserted into the container. The queue size for $p = 1$ is not as large as the case for $p = 0.75$, $p = 0.5$, and $p = 0.25$ as the PDU is loaded into the container at the occurrence of every timeslot when $p = 1$. Steady-state convergence is achieved after the 8^{th} timeslot.

Fig. 2.17 presents the impact of probability p (probability that at each timeslot, the circulating container

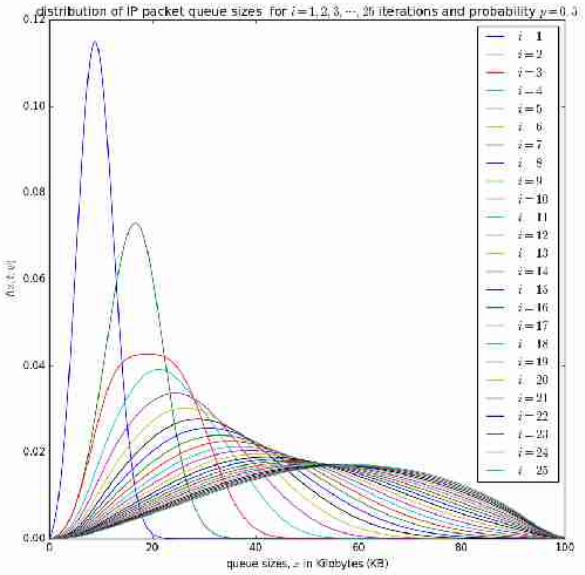

Figure 2.14: The distribution of the aggregation queue size, $f(x, t; \psi)$ if $p = 0.5$ for consecutive iterations as in Eq. (2.19) $i = 1 \ldots 25$

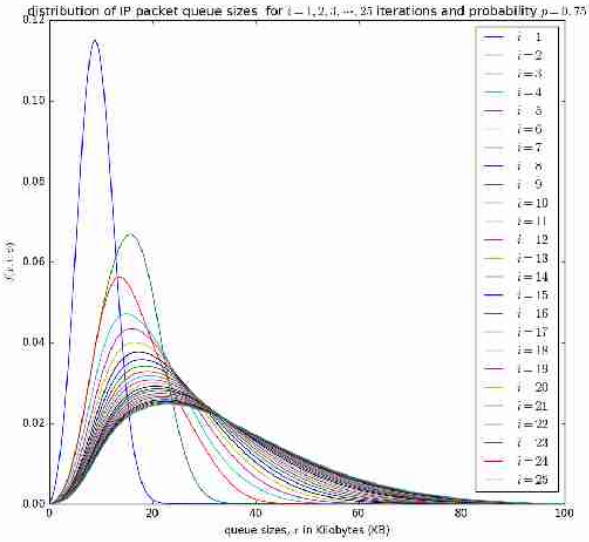

Figure 2.15: The distribution of the aggregation queue size, $f(f(x, t; \psi))$ if $p = 0.75$ for consecutive iterations as in Eq. (2.19) $i = 1 \ldots 25$

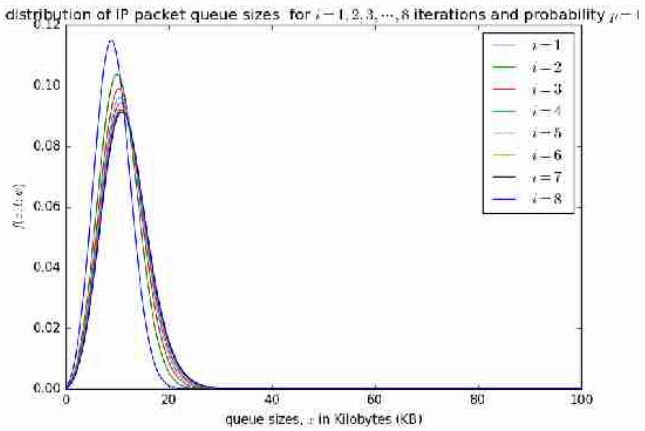

Figure 2.16: The distribution of the aggregation queue size, $f(x, t; \psi)$ if $p = 1$ for consecutive iterations as in Eq. (2.19) $i = 1 \ldots 25$

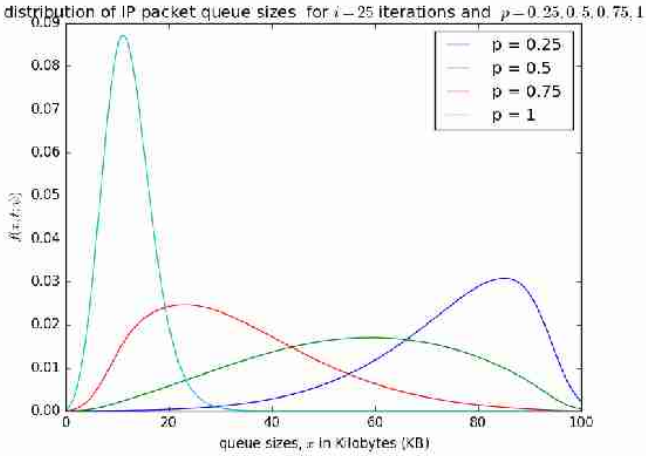

Figure 2.17: The distribution of the aggregation queue size, $f(x, t; \psi)$ for the $i = 25$ iterations and different values p

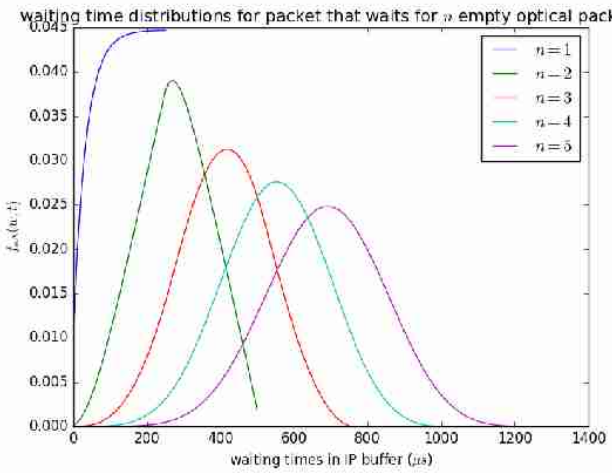

Figure 2.18: $f_{Wn}(w)$ as defined in Eq. (2.26) – the influence of the number of empty optical packets n needed to complete the transfer on the waiting time distribution, $p = 0.25$

that arrives at the buffer is empty) on the final distribution $f(x, t; \psi)$ of the queue length in bytes. If the container is not empty at a given timeslot, we try to load the PDU at the next timeslot. Fig. 2.17 shows that as p increases, the queue size in the packet aggregation buffer decreases, because as PDUs are inserted into the container at each timeslot, the queue size decreases. If the probaility p (that the circulating containers that arrive at the buffer at each timeslot are empty) decreases, then the PDUs are not inserted into the containers frequently, and the queue size of SDUs in the aggregation increases, and could likely lead to a buffer overflow, even though the aggregation buffer is over-dimensioned.

p_n	$p = 0.25$	$p = 0.5$	$p = 0.75$	$p = 1$
$n = 1$	0.000422	0.011993	0.085542	0.5581957
$n = 2$	0.004069	0.066236	0.287785	0.430287
$n = 3$	0.016506	0.130243	0.276826	0.011427
$n = 4$	0.049166	0.183986	0.185722	$8.86 * 10^{-5}$
$n = 5$	0.119076	0.210541	0.100210	$3.66 * 10^{-7}$

Table 2.1: Probabilities p_n, $n = 1, \ldots, 5$ that arriving SDU joins the queue before the interval $x \in [(n-1)L, nL]$, as in Eq. (2.27)

Table 2.1 presents probabilities p_n that that arriving SDU joins the queue before the interval $x \in [(n-1)L, nL]$ and will be aggregated and inserted into the container after n time slots.

Fig. 2.18 shows the influence of the number of timeslots n on the waiting time of an arriving SDU that arrives and sees the queue size between $(n-1)L$ and nL. When an SDU arrives and sees a queue size of about nL, it waits for n timeslots, and at each timeslot, SDUs are aggregated to a PDU of size L and inserted into the container, provided that the container is empty. The distribution in figure 2.18 ($f_{Wn}(w)$) is obtained from Eq. (2.26). It is described as the waiting time distributions for packets that waits for n empty

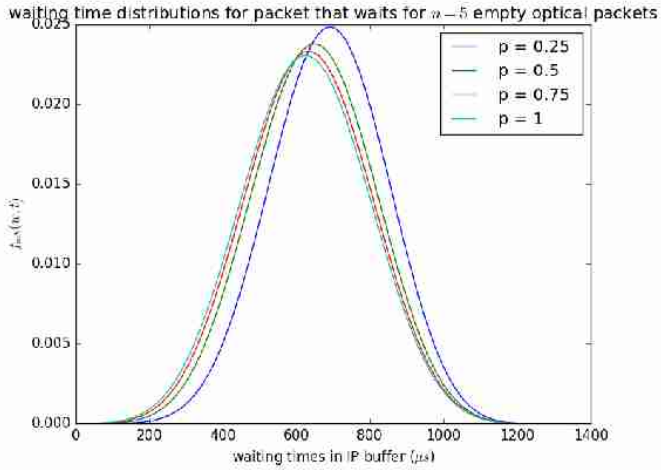

Figure 2.19: $f_{W5}(w)$ as defined in Eq. (2.26) – the influence of the probability p of the empty optical packet on the distribution of waiting time if $n = 5$

optical packets because the containers that the PDUs are inserted are similar to optical packets in size, and their content is converted to an optical packet and transmitted. The parameter n can also be understood as the number of timeslots that will occur for n PDUs that contain SDU that arrived earlier to be inserted into available containers before the container that contains the arriving PDU is inserted. Fig 2.18 shows that as n increases, the mean waiting time experienced by arriving SDU increases and can be observed by the distributions' shifts to the right as n increases.

Fig. 2.19 illustrates the influence of the probability p that the circulating container that arrives at buffer is empty at each timeslot. If the container that arrives at the buffer in each timeslot and the container that arrives is not empty, the PDU is lot inserted into the container, and it waits for the next timeslot. Fig. 2.19 shows that as the probability p that the container is available (i.e. the container is empty) at each timeslot increases, the waiting time experienced by the SDUs decreases, and if p is small (the container is not empty) then the waiting time experienced by the SDUs increases. The distribution in figure 2.19 ($f_{Wn}(w)$) is obtained from Eq. (2.26) for $n = 5$ and p is varied from 0.25 to 1.

2.5 The Tradeoff Between Throughput, Energy Consumption, and Delay

The main goal of sustainable network design is to achieve high throughput and minimise energy consumption with an acceptable QoS (delay, packet losses, and jitter). In this section, we discuss the network performance and energy consumption metrics and how they influence one another in the context of packet aggregation.

2.5.1 The Throughput Efficiency at the Core Network

Generally, the structure of a packet contains three main parts such as the header, the payload, and the trailer. The header contains information required to process a packet (e.g. packet length, synchronisation,

protocol, packet identification number, source address, destination address information), the payload contains the actual data delivered from a source user to a destination user, and the trailer which contains information which enables the receiving device to identify the end of the packet and to perform error checking. Since the header and the trailer part of the packet only carries information required to process the packet and not the user data intended to be delivered between two communicating devices, they constitute an overhead to the network and hence, termed overhead bytes. Transporting packets in which a significant proportion (percentage) of the total packet is occupied by overhead bytes results in bandwidth wastage. Suppose that the size of the overhead byte is O_b and the average size of the payload is L_p, then the percentage of the bandwidth wasted due to overhead per packet is

$$\varepsilon_o = \frac{O_b}{O_b + L_p} x100 \tag{2.28}$$

Consider a packet with a header (Ethernet, IPv4, and UDP headers) of 42 bytes, if its payload is 8 bytes (e.g. like the case of IoT packet), then the percentage of the bandwidth consumed by the header (percentage of bandwidth wasted) is 84%. If 100 of such packets are aggregated to share the same header, then the payload size of the aggregated packet is 800 bytes, and the percentage of the bandwidth consumed by the header becomes 5%. It shows that aggregation significantly reduces the percentage of the bandwidth consumed by the headers or overhead bytes. The bandwidth efficiency per packet is

$$\varepsilon_b = \frac{L_p}{O_b + L_p} x100 \tag{2.29}$$

Therefore, aggregating the smaller packets at the edge of the network significantly improves the throughput in the core networks. The more the number of bytes of smaller packets aggregated into larger packets, the higher the bandwidth efficiency.

2.5.2 The Core Network Energy Efficiency

One of the essential benefits of packet aggregation at the edge node is reducing energy consumption in the core network. The energy consumption of the core routers depends on both the number of packets received, processed, and transmitted and on the packet sizes. Packet aggregation reduces the number of packets handled by the core routers, but it increases the packet sizes making the energy benefits offered by packet aggregation not intuitive. It has been shown theoretically and practically in [246, 115, 105] that the power consumption of a core router or switch consists of a fixed baseline power P_B and a dynamic power P_D. The baseline power is the power consumed by some components such as the cooling Fans, routing engine cards (e.g., during signalling and updating of the routing tables) and other electronic components when they are idle. The dynamic power is the power consumed by the data plane (the line cards and the switching fabric) when it is receiving, processing, and transmitting data packets. Therefore, the power profile of a network router or switch is [225, 246, 115, 105]

$$P = P_B + P_D \tag{2.30}$$

Fig. 2.20 shows a simplified router or switch structure considered for theoretical analysis of the power consumption budget of a router or switch. The baseline power is fixed but the dynamic power varies with

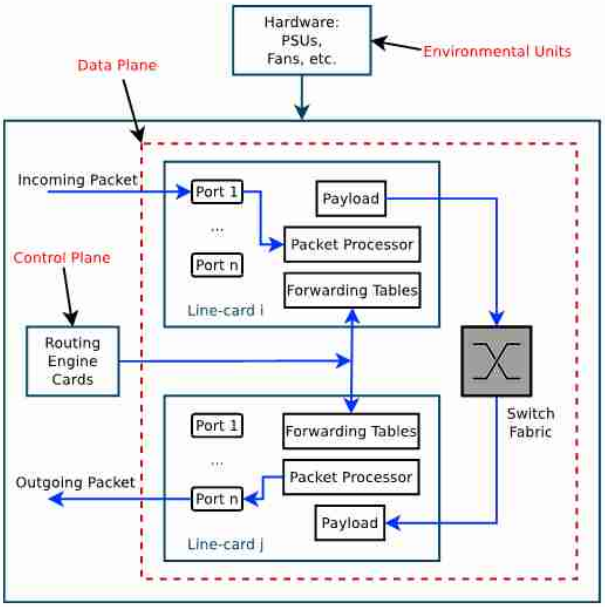

Figure 2.20: A structure of a Core Router or Switch [246]

the number of packets processed per second and on the sizes of packets. The power consumption budget for the n^{th} port of a high speed router or switch is

$$P_n = PE_n + R_{i,n}(E_{rx,n} + E_{rs,n}) + \lambda_{p,n}E_p + R_{o,n}(E_{ts,n} + E_{tx,n}) \tag{2.31}$$

Where PE_n is the power consumed by the n^{th} ethernet port when iddle (when there is no traffic on it), $R_{i,n}$ is the number of bytes received per seconds at the n^{th} port, $E_{rx,n}$ is the energy required to receive a byte on the ingress Ethernet interface of the n^{th} port, $E_{rs,n}$ is the energy required to process and store a byte on the ingress Ethernet interface of the n^{th} port, $\lambda_{p,n}$ is the number of packets from the n^{th} port processed per second, $E_{p,n}$ is the energy required to process each packet (parsing, route lookup, and forwarding), and it is the same for all packets irrespective of their sizes, $R_{o,n}$ is the number of bytes transmitted per second through the egress Ethernet interface of the n^{th} port, $E_{ts,n}$ is the energy required to process and store a byte on the egress Ethernet interface of the n^{th} port, and $E_{tx,n}$ is the energy required to transmit a byte on the ingress Ethernet interface of the n^{th} port. If the mean packet sizes at the ingress and egress interfaces of the n^{th} port are are $m_{i,n}$ and $m_{o,n}$ (in bytes) respectively, then the number of packets received per second through the ingress Ethernet interface of the n^{th} port is $\lambda_{i,n} = \frac{R_{i,n}}{m_{i,n}}$ and the number of packets transmitted per second through the egress Ethernet interface of the n^{th} port is $\lambda_{o,n} = \frac{R_{o,n}}{m_{o,n}}$ and equation 2.31 becomes

$$P_n = PE_n + m_{i,n}\lambda_{i,n}(E_{rx,n} + E_{rs,n}) + \lambda_{p,n}E_p + m_{o,n}\lambda_{o,n}(E_{ts,n} + E_{tx,n}) \tag{2.32}$$

Taking the partial dericatives of equation 2.32 with respect to the number of packets received, processed, and transmitted per second, we obtain the energy per packet. The energy per packet for the ingress Ethernet interface of the n^{th} port is

$$\frac{\partial P_n}{\partial \lambda_{i,n}} = m_{i,n}(E_{rx,n} + E_{rs,n})$$

the energy required to process a packet is

$$\frac{\partial P_n}{\partial \lambda_{p,n}} = E_p$$

It is independent of the packet size as the energy required to process (parsing, route lookup, and forwarding) a small packet and a large packet are the same. The energy per packet for the egress Ethernet interface of the n^{th} port is

$$\frac{\partial P_n}{\partial \lambda_{o,n}} = m_{o,n}(E_{ts,n} + E_{tx,n})$$

The energy per packet for the n^{th} port is

$$E_{n,p} = m_{i,n}(E_{rx,n} + E_{rs,n}) + E_p + m_{o,n}(E_{ts,n} + E_{tx,n}) \tag{2.33}$$

For $m_{i,n} = m_{o,n} = m_n$, then the energy consumption per packet is

$$E_{n,p} = m_n E_{rst} + E_p \tag{2.34}$$

where $E_{rst} = E_{rx,n} + E_{rs,n} + E_{ts,n} + E_{tx,n}$. Equation 2.34 shows that energy per varies linearly with the packet size as demonstrated in [225, 246, 115] using measurements. Therefore, aggregating smaller packets (e.g each of size m) into larger packets (e.g each of size $L = \sum_{i=1}^{K} m_i$) increases the energy consumption per packet in the core network. It means that by increasing the throughput (increasing the length of the aggregated packet, L), the energy consumption per packet at the core network also increases. For a core router or switch with K_i idle ports and K_a active ports the power consumption budget is

$$P = P_B + \sum_{n=1}^{K_i} PE_n + \sum_{n=1}^{K_a} [PE_n + m_{i,n}\lambda_{i,n}(E_{rx,n} + E_{rs,n}) + \lambda_{p,n}E_p + m_{o,n}\lambda_{o,n}(E_{ts,n} + E_{tx,n})] \tag{2.35}$$

The power consumption at the core router and switches increases with an increased number of packets received, processed and transmitted as in equation 2.35 which was demonstrated in [225, 246, 115] using measurements. In order to increase the routing or switching speeds, some core routers and switches contain Ternary Content Addressable Memories (TCAMs) in their hardware which are power-hungry electronic modules, and hence higher energy is required to process a single packet. If the core switches are SDN-based switches, then the power consumption budget considers the power consumed in searching the flow tables, installing the flow rules by the controller, and the communication between the switch and the controller [105]. Therefore, by aggregating a sufficiently large number of small packets into larger packets, the power consumption of the core routers and switches can be significantly reduced compared to increased energy consumption due to an increase in packet size. The core network energy efficiency is [242]

$$\text{Network energy efficiency} = \frac{\text{total useful traffic delivered}}{\text{total energy consumed}} \tag{2.36}$$

The objective is deliver more useful traffic (high throughput) with minimum amount of energy possible. Therefore, packet aggregation increases the throughput and reduces the energy consumption in the core network.

2.5.3 Network Delay

When a packet travels through a router or switch, it experiences a delay due to the time required to receive a packet, and the time spent waiting in the input buffer, the required to process the packet (parsing,

route lookup, and forwarding), the time spent waiting in the output buffers, and the time required to transmit the packet. The delay budget is

$$D = t_{rx} + t_{ib} + t_p + t_{ob} + t_{tx} \tag{2.37}$$

where t_{rx} is the time required to receive a packet, t_{ib} is the waiting time of the packet in the input buffer if it arrives and joins a queue, t_p is the time required to process a packet, t_{ob} is the waiting time of the packet in the output buffers, and t_{tx} is the time required to transmit a packet. For high-speed core routers, t_{rx} and t_{tx} are very small and could be ignored. A detailed analysis of the delay of a network of routers and switches is beyond the scope of this work but have been presented in [57, 57, 50].

If a port of an edge router is configured to support packet aggregation, then when the first small packet that needs to be aggregated arrives, it has to wait in the input buffer to be aggregated with other smaller packets into a larger packet. The delay introduced by the aggregation process is significantly larger than the delay experienced by a packet that joins a regular queue and is processed following a defined service discipline (e.g. first-come-first-serve or a priority-based service discipline). Therefore, equation 2.37 becomes

$$D = t_{ag} + t_{ib} + t_p + t_{ob} + t_{tx} \tag{2.38}$$

where t_{ag} is the aggregation delay discussed in sections 4 and 5 above. The aggregation delay depends on the parameters of the aggregation mechanism deployed, which also influence the throughput and energy consumption. Therefore, a reasonable tradeoff between the throughput, energy consumption, and delay should be made. A recent proposal to attain a reasonable tradeoff between QoS (high throughput and minimum delay) and energy consumption has been presented in [80]. The authors are proposing an SDN approach in which the QoS and energy consumption metrics are estimated and sent to a centralised controller that determines forwarding paths that minimise a goal function consisting of QoS and energy consumption metrics.

2.6 Conclusion

Packet aggregation is a useful strategy to increase throughput, improve resource utilisation, and reduce energy consumption in access networks, high-speed Internet core networks, and cloud computing data centre networks. The recent increase in the amounts of small packets generated by IoT networks, wireless sensor networks, and 4G/5G mobile networks has increased the need for more research on how to efficiently implement packet aggregation to meet the specific needs of these networks. The major drawback of packet aggregation mechanisms is the significant amount of delay that it introduces, making it unsuitable for packets that belong to real-time applications.

We have presented a detailed review of packet aggregation applications in access networks (IoT and 4G/5G mobile networks), optical core networks, and cloud computing data centre networks. We have also proposed diffusion approximation-based analytical models for the evaluation of the performance of packet aggregation mechanisms. We have demonstrated the use of measured traffic from real networks to evaluate analytically the performance of packet aggregation mechanisms. It is important to carefully tune the design parameters of the packet aggregation mechanism to obtain a reasonable tradeoff between throughput and energy consumption in the core routers or switches, and delay introduced at the edge router or switch by the packet aggregation process.

Chapter 3

Performance Evaluation Modelling of a Software Defined Networking (SDN) Switch

In traditional networks, each network device (e.g., router or switch) is fully or partially autonomous with respect to making routing decisions and fordwarding of packets. However, in SDN, a controller in the control plane makes routing or forwarding decisions and the data plane (which consist of SDN switches) is responsible for the forwarding of packets [213]. The SDN switches process and forward packet according to the rules stored and managed its flow tables [113].

When packets arrive at an SDN switch, they are received, processed, and transmitted. Packets can be queued up at the input and output ports they arrive and meet other packets waiting to be transmitted or processed. A packet, therefore, experiences delays at an SDN switch due to the time required to receive, process, and transmit it and the time spent waiting in queues. The queueing and processing delays significantly degrades the performance of SDN switches. The processing delay experienced by a packet results from the time required to search the flow tables (to find the flow rule that matches the content of the packet header) and apply the flow rule (either forward the packet to the output port for transmission or drop the packet).

The modelling of the flow matching process in hardware SDN switch is presented and the modelling of the flow matching process in a software SDN switch. The flow matching process is the searching the flow tables to determine the flow table entry or enteries whose matching field matches with the header fields of the packet. After matching the packet, the actions defined in the action field are applied (e.g., forward modify, or drop the packet). The majority of the research studies that have attempted to develop performance models for an SDN switch often assume that the packet processing times are exponentially distributed, which is not the case in reality. Realistic models of the packet processing process are developed and used as an input to the diffusion approximation model to obtain an analytical relationship between the SDN switch parameters and the some performance metrics. The performance of an SDN switch (especially a software SDN switch) significantly depends on the flow matching or flow lookup mechanism, which should be considered when analysing the performance of an SDN switch.

The broader use of Software Defined Network (SDN) controllers to create periodic changes in the network's topology sometimes lead to changes in traffic intensities at the various switches. Thus the transient behaviour of network components, particularly data switches, is becoming of great interest. Since standard

queueing models are difficult to analyze under time-varying conditions, we propose a tractable diffusion approximation for both the transient and steady-state behaviour of a network router. In particular, the analysis provides the steady-state and transient delay and packet loss probability as a function of traffic load and other characteristics. Using these results, we show that when SDN routers change the paths of flows frequently, the network's behaviour may often be far from its steady-state behaviour. Therefore any network optimization conducted with the help of SDN should not be based on steady-state behaviour, but rather on some metric related to the time-dependent network behaviour. A significant portion of the material presented in this chapter was published in [55, 57]. This chapter is focused on the performance modelling of an SDN switch.

3.1 Modelling of a hardware SDN switch

Hardware switches are often designed to process packets are lines rates using dedicated hardware resources. Software switches are designed to process packets using software programs deployed on general-purpose comodity hardware. Although software switches provide greater flexibility, they are very slow and introduces significant delays. The authors in [67] demonstrated emperically that the mean and variance of the delay experienced by packet in a hardware SDN switch is far smaller than that in a sofware SDN switch. Hardware switches are relatively fast compared to software switches, but their processing speeds or throughput still needs to be improved.

3.1.1 The architecture of a hardware SDN switch

Figure 3.1.1 describes the basic system architecture of a hardware SDN switch proposed in [253]. Arriving packets are temporarily queued at the input buffers and are then removed by the Arbiter and placed scheduled into the Packet Buffer. A copy of the packet header is forwarded to the Parser. The Parser parses the packet header to extract the header fields and then creates a tuple with the extracted information and forwards it to the Flow Match Unit. In the Flow Match Unit, the tuple is matched against existing flow rules stored in Flow Tables' flow entries. The flow entries in the Flow Tables are maintained under the controller's guidance and are updated when the controller installs new flow rules. The Flow Match Unit determines whether the packet is associated with a known flow and hence a known path.

In case of success (that is, the flow rule that matches the header fields of the packet is found in one of the flow table entries), the packet is then forwarded via the backplane. In case of failure (no flow table entry matches the packet header), a packet-in message will be sent by the SDN switch to the corresponding SDN controller [260] to notify the controller about the absence of a flow rule for the packet. The packet-in message contains either the packet or the packet's ID. The controller decides the correct action for the packet and then installs appropriate data in the flow table of the switch so that packets belonging to that particular flow can be forwarded subsequently. If there is no corresponding response from the controller, the packet will be dropped.

The flow table matching process involves searching the flow tables to find the entries containing corresponding action sets—i.e., flow rules that match the header of the packets of the input flows that pass through the Parser. If there is more than one flow table, the flow match process starts from the first flow

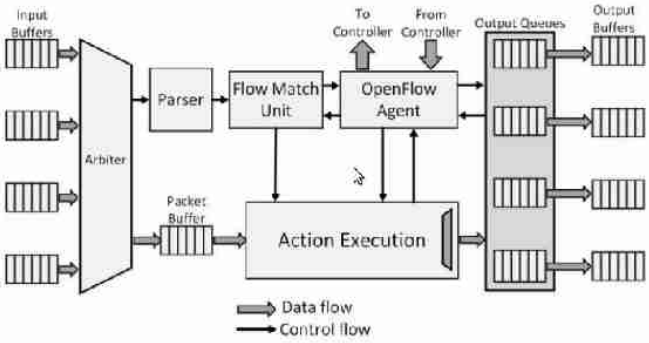

Figure 3.1: The architecture of an SDNswitch [253].

table, searches all of its entries, and jumps to the next flow table. The search process continues till an entry that matches the packet header is found. Otherwise, a “packet-in” message is generated.

In some SDN switch designs, an internal buffer or packet buffer can be implemented for the internal buffering of packets. In this case, when there is no matching flow entry for a data packet, it is internally buffered [222]. The packet ID is sent to the controller within a "packet-in" message. When the controller determines the flow rule for the packet and sends it to the switch within a "packet-out" message to update the flow tables of the switch. When the flow table is updated, the packet can then be removed from the internal buffer and processed. Then, subsequent packets that belong to the same flow can be processed according to the newly installed flow rule.

3.1.2 Modelling the flow matching process in hardware SDN switches

In a hardware SDN switch, packet processing function is embedded in a specialised hardware [222]. The layer two forwarding tables are implemented using Binary Content Addressable Memories (BCAMs). However, the layer three flow forwarding tables are implemented using Ternary Content Addressable Memories (TCAMs). The switching fabric of the a hardware SDN switch switch is often implemented using Application-Specific Integrated Circuits (ASICs). The flow tables of hardware switches are implemented using CAM and TCAM modules. The flow rules are stored in CAM and TCAM-based memory, and the packets are processed by ASICs at line speed. Making hardware SDN switches the preferred choice for high-speed or delay-sensitive networks compared to software SDN switches.The ASICs process the packets based on the flow rule.

CAM and TCAM memories are random memories. In a typical hardware network equipment, the MAC addresses used for flow lookup by the forwarding engine and TCAM stores IP addresses and subnet masks used for longest match lookups. They support read and write (update) operations and also supports parrell search operation in which the entire memory locations are searched within a single clock cycle. In each search operation, each bit of the search data (e.g., packet header) is compared with bits of the information store in the entry or memory location of the CAM (e.g., flow rules). The bits of the searched data are fed through the select lines and then compared with all the bits in the CAM cells. For a Binary Content Addressable Memory (BCAM), all the entries or memory locations are searched in parallel and all the outputs in the matched lines are passed through an encoder to obtain the matched location. Similarly, all the

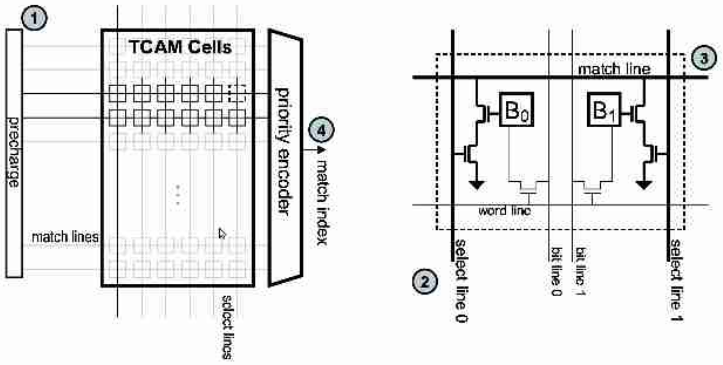

Figure 3.2: The architecture of a TCAM (left) and the structure of a TCAM cell [7].

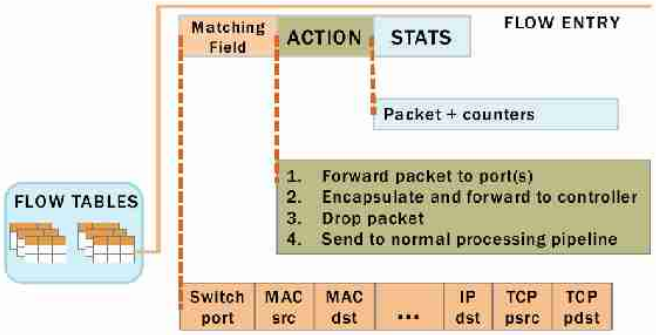

Figure 3.3: The structure of a flow table and flow entry fields [113].

entries of the tables stored in TCAM memories can also be simultaneously searched in a single clock cycle.

A TCAM memory differ from the BCAM memory in that it includes wildcard bits which will match both zero and one. That is, when searching the entries of TCAM based tables, some of the bits are don't cares and are not used in determine the matched entry. Because of the use of don't care bits in the searched data and on the content of the TCAM memory location, multiple matches may be obtained for a complete search of all the entries. In this case, the match with the highest priority is selected using a priority encoder. TCAM is also suitable for other high-speed networking applications such as packet clssification, access list control, and pattern matching detect intrusions in the network [7]. The architecture of a TCAM memory is shown in figure 3.1.2.

Packet processing consist of protocol analysis, extraction of the packet header, matching of the header fields with the flow table entries, and the execution of actions specified by the flow rules. Thus, when an SDN switch receives a packet, it needs to search or lookup flow tables to obtain the specifies how the packet should be processed. The flow table matching or lookup operation is the most slow and energy consuming packet processing operation in a TCAM-based hardware switch or router [261]. The performance of the flow matching process depends on the organisation of the flow entry data structures and on the execution of the flow lookup algorithms [70].

In practice, the flow tables are often very large, and therefore, the flow matching process could be longest

prefix matching, exact matching, or range matching. The flow matching process involves searching all the flow flow table entries to determine the flow entries that contains the flow rule that specifies the actions that should be performed on the packet. If the flow tables are searched sequentially, the time required to find the entry that matched the packet field wil be longer. For flow tables implemented using TCAMs, the flow match process can be performed within one clock cycle, with parallel access to all the entries of the flow table, resulting in a constant access time. That is, a fully parallel search of all the entries of the flow tables is performed within a single clock cycle to determine the flow entry that match the packet fields. If more than one match is found, the one with the highest prioty is considered. The structure of a flow table and flow matching fields is shown in figure 3.3.

The performance of a TCAM-based matching engine depends on the access time of the precharging circuit, matchlines, and the search line. Since all the entries of the TCAM are searched in parallel within a single clock cycle, the delay due to flow matching depends on the access time of the precharging circuit, matchlines, and the search line. It was estimated by the authors in [8] by adapting the Horowitz's approximation developed in [254]. The authors transformed the circuit of a typical TCAM cell structure with matchlines and searchlines shown in figure 3.1.2 (left figure) into simple RC circuit and then use the Horowitz's approximation to estimate the delay of the circuit. Assuming a rising input with a rise time, t_{rise}, the TCAM access delay is (see [8])

$$T_r = \tau * \sqrt{(log[v_{th}])^2 + \frac{2t_{rise}b(1 - v_{th})}{\tau}} \tag{3.1}$$

where V_{th} is the switching voltage, the constant b is the fraction of the input swing in which the output changes, and $\tau = R_{eq} * C_{eq}$ is the output time constant assuming a step input. The parameters R_{eq} and C_{eq} are respective the equivalence resistance and the equivalence capacitance of RC circuit representing a typical TCAM cell structure with matchlines and searchlines. The estimation of R_{eq} and C_{eq} a typical TCAM cell structure with matchlines and searchlines shown in figure 3.1.2 (left figure) was demonstrated in [7, 8]. Assuming a falling input with fall time t_{fall}, the TCAM access delay is (see [8])

$$T_f = \tau * \sqrt{(log[1 - v_{th}])^2 + \frac{2t_{fall}bv_{th}}{\tau}} \tag{3.2}$$

TCAMs do not only perform flow matching, they also perform other functions such as packet classification. Thus, the whole TCAM accessing bandwitch may not be available for flow matching. In some switch implementation, multiple line-cards may share the same TCAM-based matching mechanism to save cost. For large matching fields like the case with IPv6, longer flow table may be required due to distinctly increased key length. All these performance limitations result in the need for more powerful flow matching engines with scalable throughput that can ensure acceptable performance for next generation terabit routers [263].

The search operation is the most crutial operation in networking applications such as packet classification, matching the flow tables, and string matching for intrusion detection [8]. Although the search operation is performed at high speed using TCAM tables, a significant amount of energy is consumed. Also, TCAM memory is very expensive, which make the price of hardware SDN switches to be very expensive. Because of the expensive and power hungry nature of TCAMs, the capacity of flow tables implemented in TCAM is very limited. To cope with the issue of limited BCAM or TCAM-based hardware flow tables, the flow

rules that do not fit in the hardware flow tables are in a software flow table implemented in an SDRAM [209]. A high throughput but low energy consuption flow matching can be realised by storing table lookup results in an SRAM-based cach memory. Subsequent packets belonging to the flow can then be processed by the cach memory at a high throughput with low energy consumption. More sophisticated hardware means may also be designed to improve this performance (e.g., P4 switches [27]) but are not considered in this work. Therefore, throughput, energy, cost are important metrics to consider when designing TCAM-based hardware switches.

3.2 Modelling of a software SDN switch

Unlike hardware SDN switches that uses dedicated hardware resources (e.g., BCAM, TCAM, ASICs, and others) to process packets at line speed, software switches run on a general-purpose processor (e.g., CPU). Because of the expensive and power-hungry nature of BCAM and TCAM-based memories in hardware switches, the size of their flow tables are limited. Software-based switches are becoming an attractive alternative for certain types of networks (e.g., virtual networks at the data centre). The major limitation of software switches is that they are very slow in performance packet processing operations such as packet classification, flow matching, and intrusion detection. However, several algorithms and mechanisms are being developed to improve the packet processing speed of SDN-based software switches e.g., [113, 196].

In data centres, software SDN switches are used to provide flexible network services and on-demand resource provisioning. The most popular software SDN switch that is often implemented in servers to provide virtual switching services between virtual machines is the Open vSwitch (OVS). The Open vSwitch is a modular, open source, multi-platform, and OpenFlow compliant virtual switch.

3.2.1 Packet processing in a software SDN switch

To study the functional mechanism of an SDN software switch, we consider an Open vSwitch implemented on server machine and designed for flexibility and general purpose usage. An Open vSwitch consist of a userspace daemon (e.g., OVS-vswitchd) and a datapath kernel module. When a packet arrives at the physical or virtual port of the switch, it is sent the kernel datapath module.

The kernel datapath module matches the header fields of the packet with the flow table entries and then apply the actions corresponding the matched flow table entry. The action contained in the flow rule could be to forward the packet to the appropriate output port, modify the packet, or drop the packet. If no match is found, the packet is sent to the ovs-vswitchd in the userspace. The packet header fields are matched with the flow table entries in the ovs-vswitchd. If a match is found at the ovs-vswitchd, it updates the flow tables entries in the kernel datapath module. The packet and subsequent packes belonging to this flow are processed based on the updated flow rules.

If no match was found at the ovs-vswitchd, the a packet-in message is send to the controller through the OpenFlow protocol. The controller then determine the flow forwarding rule for the packet. The controller updates the flow tables stored in the ovs-vswitchd with the flow rules contained in the packet-in message that it sends to the Open vswitch through the OpenFlow protocol. The ovs-vswitchd then updates the flow tables stored in the kernel datapath modul. The packet and subsequent packets belonging to the flow are processed based on the flow rules stored in the kernel datapath module.

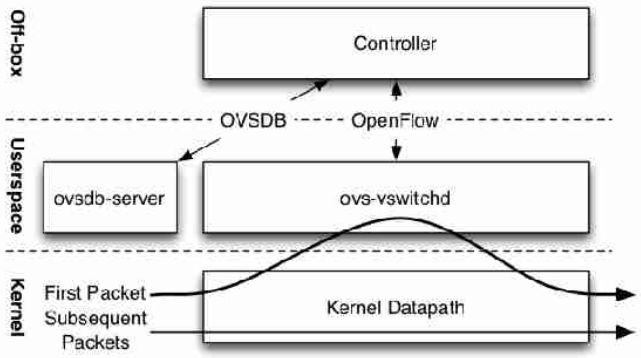

Figure 3.4: The structure of a software switch (specifically Open vSwitch) [196].

The controller only have access to manipulate (add, remove, update, search etc.) the flow tables stored in the ovs-vswitchd module. The kernel datapath module only processes that data packets based on the flow rules cached in it and is unaware of the internal details of the OpenFlow wire protocol. Thus, the seperation of the switch into the userspace and kernel datapath modules is invisible to the controller. From the point of view of the controller, the switch contains flow tables that are searched each time a packet arrives into it [196].

3.2.2 Modelling the flow matching process in software SDN switches

In a SDN software switch, the flow table entries are stored in a Synchronous Dynamic Random Access Memory (SDRAM). The packet processing locgic of a software switch is often implemented in software. One of the most popular software switch is the Open vSwitch installed in a commodity hardware such as a desktop or a Raspery Pi. When a new packet arrives, the content of its header is matched against the flow rules stored in flow tables entries in the SDRAM. If there is no matching flow table entry for the header of the packet, the packet is buffered internally and a packet-in message is generated and sent to the controller or the packet and its ID are encapsulated and sent to the controller with the packet-in message [222].

The flow matching process in a software SDN switch is performed by the CPU. The CPU can find the flow table entry that matches the content of the packet header by searching each entry of the flow table. If the flow rule for the packet is found in one of the flow entries, the actions contained in the flow rules are applied to the packet. The sequential search algorithm could be used to sequentially search through all the flow table entries or the binary search algorithm could be used.

Other sophisticated flow table search algorithms with lower time complexity implemented to improve the performance of the software switch may be considered considered. For a software switch with a small sized flow table (few hundreds), the sequential search algorithm may be consider due to its simplicity in implementation and the search time is better than that could be obtain using binary search algorithm or other sophisticated search algorithms which are complicated to implement. Software switches are cheaper as they can be implemented in a simple desktop computer or in a Raspery Pi, making them affordable. However, one of the disadvantage of using a software switch is that they are very slow and are not suitable for high

speed core networks or high speed data centre networks.

We consider the worst scenario where the flow tables are searched sequentially. Denote by p the probability that the flow rule of the arriving packet is not installed. The switch knows it after the examination of all K entries stored in the Flow Match Unit—i.e., time KT—where the examination of each entry requires time T. As a consequence, p is the probability that the service time is a constant KT, representing the case when all the flow entries in the table are examined without success, while with probability $(1-p)$ the packet's flow match is found in a time that is uniformly distributed in $[T, KT]$—i.e., on average in time $(K+1)T/2$ and variance $(K^2-1)T^2/12$. Therefore, the mean time required to search the flow tables sequentially is

$$E[T_s] = pKT + (1-p)\frac{(K+1)T}{2} \tag{3.3}$$

The the most popular software flow tables implemented using SRAM or SDRAM could be either hash-based flow table or wildcard-based flow table. In a hash-based flow tables, the match field information stored in the flow table entries are used as the input of a hash function. The hash function computes hash values which are used to store the information to be matched in the flow tables. Since the process of searching a hash-based flow table involves a single hash operation, its matching process is faster with cost of O(1). The drawback of the hash-based flow table is that it requires larger memory capcity [113] because the larger the match fields, the more flow table entries are required in the hash tables.

In wildcard-based flow tables, the flow entries are stored using wildcards. In wildcard based flow tables, some of the match fields are stored using wildcards. Some of the match fields are wildcard fields and are treated as "don't care" and are not considered during the matching process. It implies that packets belong to different flows could have the same matching rules. Thus, wildcard based flow table require fewer memory to store the flow table entries when compared to hash-based flow table. The draw back of wildcard-based flow table is that in the worst case (without any search optimization mechanism), all the flow table entries are searched sequencially.

To make a reasonable tradeoff between faster hash-based flow tables and slower wildcard based flow tables, both hash-based and wildcard-based flow tables are sometimes implemented in the kernel datapath module. The hash-based flow tables are implemented in a microflow cache but the wildcard-based flow tables are implemented in the megaflow cache. The microflow cache is implemented as a simple hash table. When a packet is received by the kernel datapath module, the packet header fields are hashed and exact matched with the match fields stored a hash value.

If no match is found in the microflow table, then the megaflow tables are searched. If a match is found at the megaflow table, then the packet is processed according the flow rules stored in the matched entry. The microflow cache is updated with the found rule so that subsequent packets of the same flow can be handle by the faster microflow cache. If no match is found, the packet or the packet ID is forwarded to the userspace module. The flow tables in the userspace module are searched. If a matched is found, then, the flow tables in the kernel datapath module are updated otherwise, the a packet-in message is sent to the controller. The controller determines the appropriate flow rule for the packet and then updates the flow table in the userspace module. The userspace module then updates the flow tables in the kernel datapath module.

The advantage of implementing the microflow cache and megaflow cache at the kernel datapath module is that it reduces the probability of a packet being sent to the userspace for flow table lookup to obtain appropriate flow rules. When the first packet of a flow enters the userspace and the appropriate flow rule is

determined, the found flow rule is cached in the microflow cache and megaflow cache. Subsequent packets will hit the microflow cache or the microflow cache in the kernel datapath module. Thus, speeding up the flow table matching speed.

Suppose that the time required to perform the hash peration to exact match the packet header fields with the flow information stored in the hash-based flow table is T_1 and the probability of hitting a table is p_1. Also, let us suppose that the time required to match a single flow entry of the tables at the megaflow cache and in the userspace is T_2 and T_3, respectively. The probability of hitting a table at the megaflow cache is p_2 and the probability of hitting a table at the userspace is p_3. If the flow tables at the megaflow cache and at the userspace are searched sequentially, then the average time required to find a match is

$$E[T] = p_1 * T_1 + p_2\frac{(K_2+1)T_2}{2} + (1-p_2)K_2T_2 + p_3\frac{(K_2+1)T_3}{2} + (1-p_3)K_3T_3 \tag{3.4}$$

where, $p_1+p_2+p_3 = 1$. K_2 and K_3 is the total number of flow entries in the megaflow cache and userspace module respectively. The time required to find the table entry that matched the hash-based flow table in the microflow cache is independ of its number of entries, K_1 becase it is performed with a single hash operation.

The performance of flow matching process mechanism at the megaflow cache and userspace module can be improved using other sophisticated algorithms. To improve the lookup performance of the flow tables in the megaflow cache and in the userspace, the authors in [196] implemented a Tupple Space Search (TSS) packet classification and lookup algorithm. The authors argued that the TSS algorithm is preferable for Open vSwitches deployed in data centres because it supports efficient constant-time table entry updates. In data centres, new services are often added and deleted, neccessitating frequent updates of the flow tables. If each tupple or hash table is equally likely to contain a match, then finding a match requires searching $\frac{(K+1)}{2}$ tables on average. In case there is no match, all the K tables must be searched. Although decision tree-based algorithms provide better lookup performance, TSS-based algorithm are still preferable in Open vSwitches [158], especially those used in data centres.

3.3 Queueing model of an SDN switch

The delay that can be experienced by packets consists of the queueing delay in the input buffer, the processing delay in the input buffer, queueing delay at the output buffers, and transmission delay. When the output ports' processing and line speeds are significantly greater than the Openflow processing time, which includes the time required to parse the packet, check the flow tables to find matching entries, and execute the flow rule actions, the switch can be represented by a single server queueing as in several recent papers [170, 16, 223, 153, 76, 178, 228]. Since the size of the input buffer is limited, we represent the SDN switch as a single server queueing model with finite capacity N.

A majority of previous papers model the packet processing time as an exponentially distributed random variable [170, 16, 223, 153, 76, 178]. The use of a diffusion process allows to use realistic packet processing models that consider practical flow table lookup mechanisms such as those discussed in the previous section. Also, diffusion approximation does not place any restriction on the distribution of the interarrival time of packets the queue. Real or measured data of the interarrival times on packets into the switch can be used. Therefore, an SDN switch with high speed transmission ports can be represented as queueing model with general interarrival and general service times.

Since the time required to check the flow tables and to forward the packets in hardware SDN switch is constant, theie input queues should be modelled as a queueing system with deterministic service times (e.g G/D/1/N), contrary to the popularly used markovian models. A software switch with any implemented flow lookup algorithm (including other packet processing algorithms) could be represented as a G/G/1/N queueing system.

3.4 Diffusion Model of an SDN Switch

To analyze this system, we use a continuous state space and continuous time diffusion process $\{X(t), t \geqslant 0\}$ to replace the discrete state-space buffer queue, where the increments $dX(t) = X(t+dt) - X(t)$ are normally distributed, with mean βdt and variance αdt, which appear in the diffusion Equation (1).

Assuming an arrival rate λ and average service time μ^{-1}, the changes in a small time interval ΔT tend to a normal distribution with mean $(\lambda-\mu)\Delta T$ and variance $(\lambda^3\sigma_A^2+\mu^3\sigma_B^2)\Delta T = (\lambda C_A^2+\mu C_B^2)\Delta T$, where σ_A^2 and σ_B^2 are the variances of the interarrival and service times, and C_A^2 and C_B^2 are the corresponding squared coefficients of variation. Therefore, for the diffusion process we have $\beta = \lambda - \mu$ and $\alpha = \lambda C_A^2 + \mu C_B^2$ [85].

The buffer's size is limited to N packets, therefore the diffusion process resides in the interval $[0, N]$, and we use a diffusion process with returns from the barriers at $x = 0$ and $x = N$ to represent the jumps that occur when the buffer queue is empty and a packet arrives, and when the queue is full and a service occurs as in [86], leading to the equations:

$$\begin{array}{lcl} \frac{\partial f(x,t;x_0)}{\partial t} & = & \frac{\alpha}{2}\frac{\partial^2 f(x,t;x_0)}{\partial x^2} - \beta\frac{\partial f(x,t;x_0)}{\partial x} + \lambda p_0(t)\delta(x-1) + \mu p_N(t)\delta(x-N+1), \\ \frac{dp_0(t)}{dt} & = & \lim_{x\to 0}[\frac{\alpha}{2}\frac{\partial f(x,t;x_0)}{\partial x} - \beta f(x,t;x_0)] - \lambda p_0(t) \\ \frac{dp_N(t)}{dt} & = & \lim_{x\to N}[\frac{\alpha}{2}\frac{\partial f(x,t;x_0)}{\partial x} - \beta f(x,t;x_0)] - \mu p_N(t), \end{array} \tag{3.5}$$

where $f(x,t;x_0)$ is the probability density function (pdf) of the diffusion process; $p_0(t)$ and $p_N(t)$ are, respectively, the probabilities that the process is at the barrier at $x = 0$ or $x = N$ at time t, corresponding to probabilities that the system is empty or saturated; and $\delta(x)$ is the Dirac delta function.

The first of the above equations defines the pdf of the diffusion process with jumps from $x = 0$ to $x = 1$ (arrival of the first customer after the idle period) with intensity λ and from $x = N$ to $x = N - 1$ (departure of a customer ending the saturation period) with intensity μ. The next two equations represent the probability balance of the barriers.

3.4.1 Steady-state analysis of the performance a SDN switch

In steady state, when $\lim_{t\to\infty} p_0(t) = p_0$, $\lim_{t\to\infty} p_N(t) = p_N$, $\lim_{t\to\infty} f(x,t;x_0) = f(x)$, Equation (3.5) becomes an ordinary differential one and its solution, for $\varrho = \lambda/\mu$, $\rho < 1$, can be expressed as [85]:

$$f(x) = \begin{cases} \dfrac{\lambda p_0}{-\beta}(1 - e^{zx}) & \text{for} \quad 0 < x \leqslant 1\ , \\ \dfrac{\lambda p_0}{-\beta}(e^{-z} - 1)e^{zx} & \text{for} \quad 1 \leqslant x \leqslant N-1\ , \\ \dfrac{\mu p_N}{-\beta}(e^{z(x-N)} - 1) & \text{for} \quad N-1 \leqslant x < N\ , \end{cases} \tag{3.6}$$

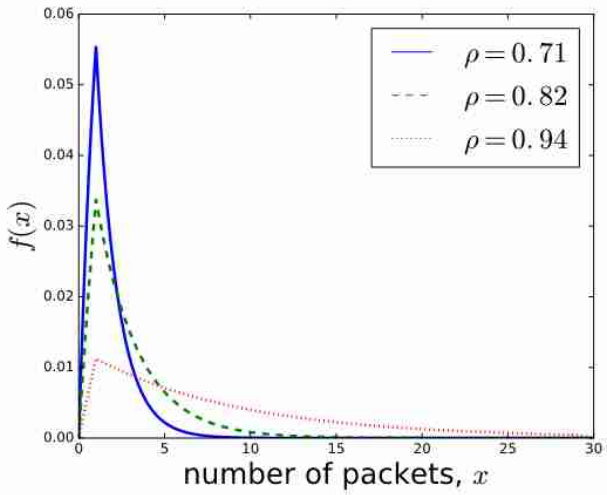

Figure 3.5: Steady-state pdf $f(x)$ of the diffusion process as an approximation of the queue distribution: illustration of the solution in Equation (3.6).

where $z = \frac{2\beta}{\alpha}$, and due to normalization:

$$\begin{aligned} p_0 &= \{1 + \varrho e^{z(N-1)} + \frac{\varrho}{1-\varrho}[1 - e^{z(N-1)}]\}^{-1} , \\ p_N &= \varrho p_0 e^{z(N-1)} . \end{aligned} \tag{3.7}$$

In this way, $f(n)$, given by Equations (3.6) and (3.8), approximates the steady-state distribution $p(n)$ in the Packet Buffer queue. A few examples of the curve $f(x)$ depending on $\varrho = \lambda/\mu$—i.e., the utilization of the system—are presented in Figure 3.4.1. The next figure presents $p_N(\varrho)$, which is the loss probability due to the buffer overflow.

The steady-state queueing delay can be modelled by the time it takes the diffusion process to drift from the point $x = x_0$, corresponding to the queue length at the moment of the packet arrival, to $x = 0$ when the packet is already on the head of the queue (its distance to the transmitter is equal to zero), and is removed to be forwarded. The density of the diffusion process $f(x)$ given by Equation (3.6) determines the queue distribution and, at the same time, the density of the initial point x_0 at Equation (3.12).

The density $\phi(x, t; x_0)$ of the diffusion process starting at x_0 and ending at the absorbing barrier at the origin is given in [44]. The method of images, usually applied to heat conduction problems, is used. We may imagine the barrier as a mirror with an image source placed at $x = 2x_0$, and the solution is a superposition of a source of unit strength, placed at the origin and a source of strength $-\exp(\frac{2\mu x_0}{\alpha^2})$ placed at $x = 2x_0$:

$$\phi(x, t; x_0) = \frac{e^{\frac{\beta}{\alpha}(x-x_0) - \frac{\beta^2}{2\alpha}t}}{\sqrt{2\pi\alpha t}} \left[e^{-\frac{(x-x_0)^2}{2\alpha t}} - e^{-\frac{(x+x_0)^2}{2\alpha t}} \right] . \tag{3.8}$$

The density function $\gamma_{x_0,0}(t)$ of the first passage time from $x = x_0$ to $x = 0$, i.e., probability density that the process enters the barrier at time t, is equal to the probability density that the process is leaving the

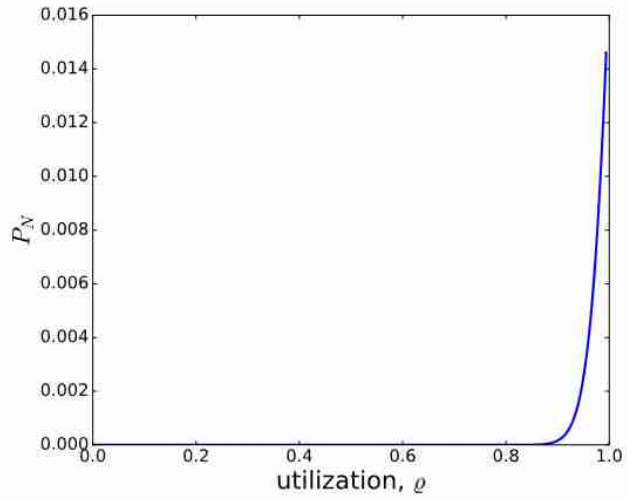

Figure 3.6: Probability p_N of the buffer overflow; illustration of the solution given by Equation (3.8).

diffusion interval ($x > 0$):

$$\gamma_{x_0,0}(t) \quad = \quad \frac{\partial}{\partial t}\int_{0+}^{\infty}\phi(s,t;x_0)dx = \lim_{x\to 0}[\frac{\alpha}{2}\frac{\partial}{\partial x}\phi(x,t;x_0) - \beta\phi(x,t;x_0)] = \frac{x_0}{\sqrt{2\Pi\alpha t^3}}e^{-\frac{(x_0+\beta t)^2}{2\alpha t}} \quad (3.9)$$

This density should be normalized to include only the cases when the process ends at the barrier, which is certain for $\beta < 0$. Therefore:

$$\int_0^{\infty}\gamma_{x_0,0}(t)dt = e^{\frac{2x_0\beta}{\alpha}}\ . \quad (3.10)$$

The first passage time of the diffusion process from the point $x = x_0$ to the barrier at $x = 0$ becomes:

$$\gamma_{x_0,0}(t) = \frac{x_0}{\sqrt{2\Pi\alpha t^3}}e^{-\left[\frac{2x_0\beta}{\alpha}+\frac{(x_0-\beta)^2}{2\alpha t}\right]}. \quad (3.11)$$

Suppose that a newly arrived packet joins the queue when the switch already contains x packets. Assuming the first-in-first-out service, the packet will be forwarded out from the switch after all the packets that arrived earlier have been forwarded, so that if the queue length probability density function is $f(x)$, the probability density function of the packet's queueing delay is:

$$f_R(t) = \int_0^{\infty}\left[\frac{x}{\sqrt{2\Pi\alpha t^3}}e^{-\left[\frac{2x\beta}{\alpha}+\frac{(x-\beta)^2}{2\alpha t}\right]}\right]f(x)dx. \quad (3.12)$$

Figure 3.4.1 illustrates this result with a few curves of $f_R(t)$ for different values of the traffic intensity ρ, and with the parameters; $C_A^2 = C_B^2 = 1$, which have been used in all the examples of Figures 3.4.1–3.4.1.

The mean delay experienced by a packet whose flow rule is contained in the flow table will be the sum of the queueing delay and processing time. If the mean queueing delay is D_q and the processing time is t_p, then the mean packet delay at the switch D_s is:

$$D_s \quad = \quad D_q + t_p = \int_0^{\infty} t f_R(t)dt + \frac{1}{\mu}. \quad (3.13)$$

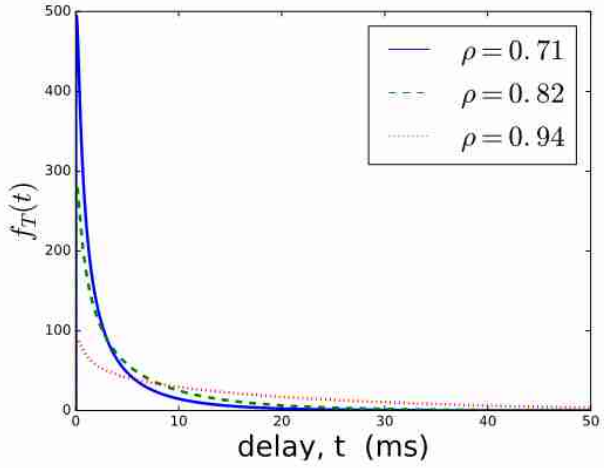

Figure 3.7: Density $f_R(t)$ of the queueing time; see Equation (3.12).

However, if a flow entry for an arriving packet is not found, then the packet is encapsulated and sent to the controller, which determines the flow rules, installs the flow entry for the packet, and sends back the packet in the packet-out message. In that case, the delay experienced by the packet, D is the sum of the delay in the switch and the delay at the controller D_c [170, 16, 223, 153, 76, 178]:

$$D = (1 - p)D_s + pD_c. \tag{3.14}$$

The modelling of the controller's working mechanism to determine D_c is beyond the scope of this work and will be considered in future works, and in our numerical examples all the flows are known, i.e., $p = 0$.

3.4.2 Transient-state analysis of the performance a SDN switch

In the case of steady-state analysis, the first two moments of the interarrival and service times used to calculate the diffusion parameters are constant. However, due to the unpredictable characteristics of user traffic and the use of adaptive routing protocols such as the self-aware routing protocol used in SerIoT SDN core network, the characteristics of the traffic arriving at the input and output buffers are dynamic. It requires the transient delay analysis within short time intervals, where the diffusion parameters are constant only with these interval time interval.

Consider a diffusion process with two absorbing barriers at $x = 0$ and $x = N$, that started at $t = 0$ from $x = x_0$ and that its probability density function $\phi(x, t; x_0)$ has the following form [44]

$$\phi(x, t; x_0) = \begin{cases} \delta(x - x_0 & \text{for} \quad t = 0\ , \\ \dfrac{1}{\sqrt{2\Pi\alpha t}} \displaystyle\sum_{n=-\infty}^{\infty} \{a(t) + b(t)\} & \text{for} \quad t > 0\ , \end{cases} \tag{3.15}$$

where

$$a(t) = \exp\left[\frac{\beta x'_n}{\alpha} - \frac{(x - x_0 - x'_n - \beta t)^2}{2\alpha t}\right],$$

$$b(t) = \exp\left[\frac{\beta x_n''}{\alpha} - \frac{(x - x_0 - x_n'' - \beta t)^2}{2\alpha t}\right],$$

and $x_n' = 2nN$, $x_n'' = -2x_0 - x_n'$.

Suppose that the diffusion process starts at point ξ with PDF $\psi(\xi)$, $\xi \in (0, N)$, $\lim_{\xi \to 0} \psi(\xi) = \lim_{\xi \to N} \psi(\xi) = 0$, then the PDF of the process has the form

$$\phi(x, t; \psi) = \int_0^N \phi(x, t; \xi)\psi(\xi)d\xi. \tag{3.16}$$

The Laplace transform of $\phi(x, t; x_0)$ can be expressed as

$$\bar{\phi}(x, s; x_0) = \frac{\exp[\frac{\beta(x-x_0)}{\alpha}]}{A(s)}. \tag{3.17}$$

$$\cdot \sum_{n=-\infty}^{\infty} \Big\{\exp\left[-\frac{|x - x_0 - x_n'|}{\alpha}A(s)\right] - \exp\left[-\frac{|x - x_0 - x_n''|}{\alpha}A(s)\right]\Big\}, \tag{3.18}$$

where $A(s) = \sqrt{\beta^2 + 2\alpha s}$.

Since the transient solution of equation (3.5) is not analytically tractable, the probability density function $f(x, t; \psi)$ of the diffusion approximation process with elementary returns boundaries can be obtained numerically. It is composed of the function $\phi(x, t; \psi)$, which is the probability density function of the diffusion process with absorbing barriers at $x = 0$ and $x = N$ and the functions $\phi(x, t - \tau; 1)$ and $\phi(x, t - \tau; N - 1)$ which are probability density functions of the diffusion processes that started at time $\tau < t$ at points $x = 1$ and $x = N - 1$ with densities $g_1(\tau)$ and $g_{N-1}(\tau)$ with instantaneous jumps [61][48][60]

$$f(x, t; \psi) = \phi(x, t; \psi) + \int_0^t g_1(\tau)\phi(x, t - \tau; 1)d\tau + \int_0^t g_{N-1}(\tau)\phi(x, t - \tau; N - 1)d\tau \ . \tag{3.19}$$

The densities $g_1(t)$ and $g_N(t)$ may be expressed with the use of functions $\gamma_0(t)$ and $\gamma_N(t)$:

$$\begin{aligned} g_1(\tau) &= \int_0^\tau \gamma_0(t)l_0(\tau - t)dt \\ g_{N-1}(\tau) &= \int_0^\tau \gamma_N(t)l_N(\tau - t)dt \ , \end{aligned} \tag{3.20}$$

where $l_0(x)$, $l_N(x)$ are the densities of sojourn times at $x = 0$ and $x = N$ respectively, while $\gamma_0(t)$ and $\gamma_N(t)$ are the probability densities that at time t the process enters to $x = 0$ or $x = N$ are

$$\begin{aligned} \gamma_0(t) &= p_0(0)\delta(t) + [1 - p_0(0) - p_N(0)]\gamma_{\psi,0}(t) \\ &\quad + \int_0^t g_1(\tau)\gamma_{1,0}(t - \tau)d\tau \\ &\quad + \int_0^t g_{N-1}(\tau)\gamma_{N-1,0}(t - \tau)d\tau \ , \\ \gamma_N(t) &= p_N(0)\delta(t) + [1 - p_0(0) - p_N(0)]\gamma_{\psi,N}(t) \\ &\quad + \int_0^t g_1(\tau)\gamma_{1,N}(t - \tau)d\tau \\ &\quad + \int_0^t g_{N-1}(\tau)\gamma_{N-1,N}(t - \tau)d\tau \ , \end{aligned} \tag{3.21}$$

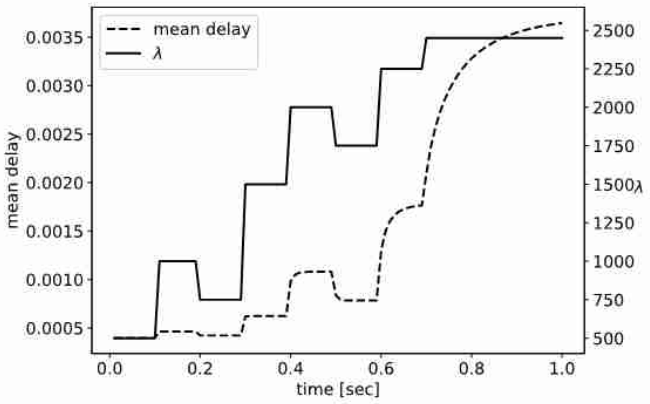

Figure 3.8: The effect of abrupt changes the traffic arrival rate λ on the time dependent behaviour of the expected packet delay at the router.

where $\gamma_{1,0}(t)$, $\gamma_{1,N}(t)$, $\gamma_{N-1,0}(t)$, $\gamma_{N-1,N}(t)$ are densities of the first passage time between corresponding points, e.g.

$$\gamma_{1,0}(t) = \lim_{x\to 0}[\frac{\alpha}{2}\frac{\partial\phi(x,t;1)}{\partial x} - \beta\phi(x,t;1)] \,, \tag{3.22}$$

and for absorbing barriers,

$$\lim_{x\to 0}\phi(x,t;x_0) = \lim_{x\to N}\phi(x,t;x_0) = 0 \,,$$

hence $\gamma_{1,0}(t) = \lim_{x\to 0}\frac{\alpha}{2}\frac{\partial\phi(x,t;1)}{\partial x}$. The functions $\gamma_{\psi,0}(t)$, $\gamma_{\psi,N}(t)$ denote the probability densities that the initial process that started at $t = 0$ at the point ξ with density $\psi(\xi)$ will end at time t by entering respectively $x = 0$ or $x = N$.

The Laplace transform of the density function $f(x,t;\psi)$ is

$$\begin{aligned}\bar{f}(x,s;\psi) = \bar{\phi}(x,s;\psi) + \bar{g}_1(s)\bar{\phi}(x,s;1) \\ +\bar{g}_{N-1}(s)\bar{\phi}(x,s;N-1),\end{aligned} \tag{3.23}$$

and the densities $\bar{g}_1(s), \bar{g}_{N-1}(s)$ are obtained from (3.20), (3.21), (3.22) after their Laplace transform. The probabilities that at time t the process has the value $x = 0$ or $x = N$ are

$$\begin{aligned}\bar{p}_0(s) &= \frac{1}{s}[\bar{\gamma}_0(s) - \bar{g}_1(s)], \\ \bar{p}_N(s) &= \frac{1}{s}[\bar{\gamma}_N(s) - \bar{g}_{N-1}(s)].\end{aligned} \tag{3.24}$$

The above solution gives the transient distribution of the queue length and the transient probability of packet losses when the buffer is full. The original functions of the Laplace transforms can be obtained numerically using Stehfest's algorithm [230], valid for constant diffusion parameters, i.e. constant traffic intensity λ. Therefore it is used for time intervals within which parameters are constant and the solution at the end of such interval serves as the initial condition, i.e. function $\psi(\xi)$ in (3.16) in the next interval with different parameters. The mean queueing delay was determined with the use of Little's formula but the first passage time approach is also possible.

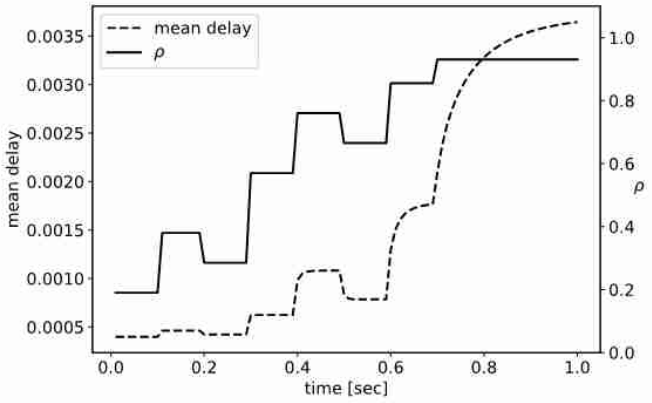

Figure 3.9: Impact of abrupt changes in the queue utilisation ρ on the time deendent behaviour of the expected packet delay.

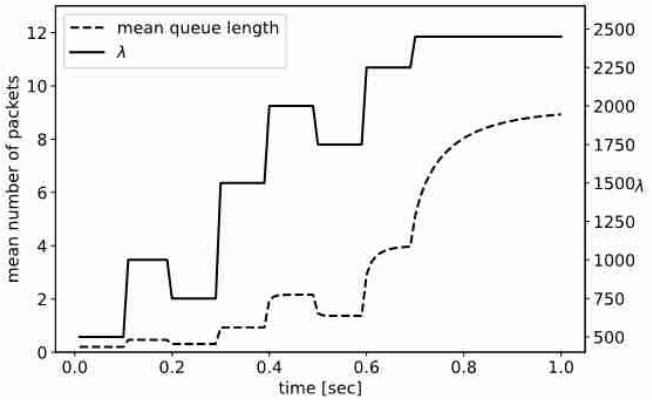

Figure 3.10: The effect of abrupt changes in the packet arrival rate on the time-dependent behaviour of the expected queue length.

Based on the previous analysis, we have examined the effect of changes in the levels of arriving traffic rates to a router which may result from path changes created by SDN controllers. We have assumed that the router's packet buffer is partitioned into $N = 100$ packet sections where each section is reserved for a given active packet flow. When a packet arrives at the buffer, the time it takes to scan the table that contains the list of flows is assumed to be uniformly distributed with average value $S = 0.038ms$ and squared coefficient of variation is 0.33; though these values vary with the router hardware, they are compatible with those of existing equipment.

In Fig. 3.4.2 the arriving traffic rates of a given flow vary in the range of 500 to 2500 packets per second, and the traffic level λ changes approximately every $100ms$ reflecting relatively frequent path changes. We notice that, while at low traffic values the mean delay of a packet closely matches the steady-state value which is reached rapidly, at high values the mean delay always remains in its transient state so that the steady-state value is a poor predictor of the actual delay experienced by packets. Similar results, for another sequence of changes in the traffic arrival rate are shown in Fig. 3.4.2, where the time-dependent mean packet delay is plotted against the queue utilization $\rho = \lambda S$. Confirming the results of the previous figure, we see here too that as ρ increases, the mean packet delay through the router never actually attains its steady-state value. Fig. 3.4.2 displays the changes of mean queue length together with changes of traffic intensity λ. The errors of the method of diffusion approximation were studied numerically in detail several times, e.g. in [61] and were found acceptable, therefore we do not here present any comparison of the diffusion results with discrete event simulation.

3.5 Performance threshold and load control

We have shown that the delay and packet loss probability increases slowly with the load parameter, ρ and then at a certain value of ρ, a slight increase in ρ will cause corresponding sharp increase in the delay or response time and packet loss probability. These performance thresholds values can then be used in the goal function in a self-aware route computation mechanism to ensure that the installation of new flow rules will not result in worst performance on some network paths.Therefore, the design of the SerIoT data plane and the SerIoT route computation mechanism must guarantee that the performance of any of the data plane forwarding devices does enter into regimes where a small increase in the load will lead to a large increase in the delay and packet loss and it becomes a bottleneck in the network.

If we take the parameter ρ, defined from the drift $\beta = \lambda - \mu = -\mu(1-\rho)$, so that $\rho = 1 + \frac{\beta}{\mu}$, considering that we can easily monitor the parameter $b_0 = [1 - p_0]$, it would be useful to study the sensitivity of b_0 on the load factor ρ, i.e. $\frac{\partial b_0}{\partial \rho}$

Consider a single service system with arrival rate λ, service rate μ, so that $\rho = \frac{\lambda}{\mu}$. In the simplest case we can assume Poisson arrival rates and exponentially distributed service times so that the average queue length in steady-state N and the average response time W are:

$$N = \frac{\rho}{1-\rho},\ W = \frac{1}{\mu(1-\rho)} \tag{3.25}$$

and:

$$\frac{dN}{d\rho} = \frac{1}{(1-\rho)^2}, \tag{3.26}$$

so that N is obviously an increasing function of ρ.

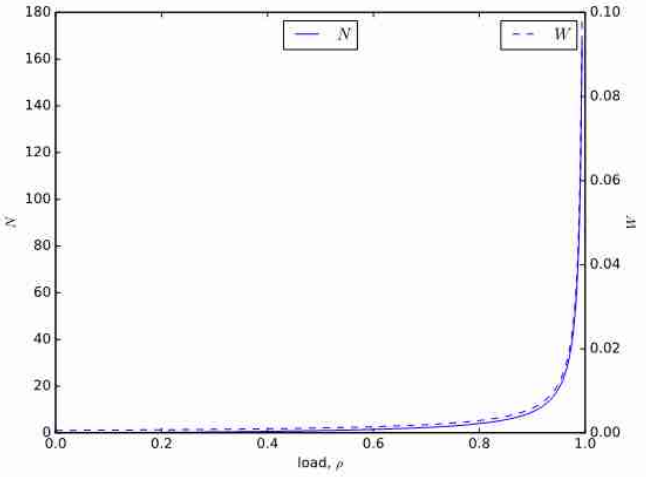

Figure 3.11: Load characteristics for an M/M/1 model

We can say that we wish W to be less than some value $\frac{B}{\mu}$ for some constant $B \geqslant 1$, in which case we have:

$$\rho = 1 - \frac{1}{B}. \tag{3.27}$$

This gives us the value that ρ should not exceed.

We are also trying to characterize those values of ρ for which the queue length N is very sensitive to small changes in ρ, because we do not wish the system to enter into regimes where a small increase in the load will lead to a large increase in the response time. We can state this as looking for those values of ρ for which:

$$N(\rho + \Delta) \approx N(\rho) + \Delta.\frac{dN}{d\rho} = N(\rho) + K.N(\rho), \tag{3.28}$$

for some value $K > 0$, or

$$\frac{\Delta}{K} = \rho(1 - \rho), \tag{3.29}$$

$$\rho^2 - \rho + \frac{\Delta}{K} = 0, \tag{3.30}$$

$$\rho = \frac{1}{2}[1 + \sqrt{1 - \frac{4\Delta}{K}}]. \tag{3.31}$$

For instance, if we set $K = 1$, it means that we are seeking the value of ρ for which an increase of ρ by an amount Δ results in an approximate 100% increase in N. If we set $\Delta = 0.1$ we see that we obtain the approximate value $\rho = 0.887$ as shown in figure 3.5.

If we choose $K = 0.5$ or a 50% increase in queue length, we will have the approximate value $\rho = 0.724$. Of course, similar calculations can be conducted for the average response time, and the calculations can be more accurate we do not limit ourselves to the first order approximation (3.28) but also use higher orders.

3.6 Conclusions

Networks that are controlled by SDN are subject to frequent changes in network state as the SDN controller modifies paths in the network to optimize Quality of Service, Security or Energy Consumption. These frequent changes may imply that rather than running at a steady-state regime, the network will mostly find itself in transitory states. Diffusions approximations are far more convenient for the transient analysis of service systems, rather than queueing networks and discrete event simulation. Therefore we examine the transient behaviour of a network router with a diffusion approximation model to evaluate both the transient and steady-state performance of a network router, in order to predict packet delay through the router, and its packet loss probability.

The diffusion approximation shows that the transitory behaviour of each router depends on the load, which results from the arrival rate of packets and the service process for each packet leaving the router. The service process , in turn, depends on the number of flows that the router handles because a possibly large flow table has to be searched to determine each incoming packet's outgoing link [228]. Thus our model also takes into account the dependence of the service time for each outgoing packet on the size of the flow table.

Our analysis we have presented in this chapter allows the prediction of the time-dependent behaviour of important performance metrics such as the mean delay experienced by a packet at the router, the packet queue length for each flow, and the packet loss probability. It also showed that the time-dependent behaviour tends much more slowly to its steady-state when the system is more heavily loaded. Numerical examples based on the analysis are also presented to illustrate these insights.

As a consequence of our analysis, we have seen that future work should consider SDN based network optimization techniques that focus both on the transient and steady-state behaviour, because the steady-state may not be attained in many cases. Future work should also compare these theoretical results with measurements and investigate the performance implications of the detailed interaction of SDN controllers with their connected routers. Also, we hope to use diffusion approximations to evaluate the performance of networks or systems where the objective of the controls is to optimize the performance of the system [249].

Chapter 4

Performance Evaluation Modelling of a Network of SDN Data Plane Switches

It has been recently observed that Software Defined Networks (SDN) can change the paths of different connections in the network at a relatively frequent pace to improve the overall network performance, including delay and packet loss, or to respond to other needs such as security. These changes mean that a network that SDN controls will seldom operate in steady state; rather, the network may often be in transient mode, especially when the network is heavily loaded and path changes are critically important. Hence, we propose a transient analysis of such networks to better understand how frequent changes in paths and the switches' workloads may affect multi-hop networks' performance. Since conventional queueing models are difficult to solve for transient behaviour and simulations take excessive computation time due to the need for statistical accuracy, we use a diffusion approximation to study a multi-hop network controlled by SDN. The results show that network optimization should consider the transient effects of SDN and that transients need to be included in the design of algorithms for SDN controllers that optimize network performance.

In this chapter we extend the approach we developed in [51] to the time-dependent analysis of multiple SDN switches using diffusion approximations , which are very convenient to analyze in a time-dependent regime. Thus, we compute the transient behaviour of each SDN switch after changes occur in its input traffic rate. Packet loss probabilities can also be computed even when they are "tiny" and impossible to estimate by conventional means. The analysis we undertake considers both single SDN switch and multiple interconnected SDN switches controlled by an SDN controller. A significant portion of the material presented in this chapter were published in [54, 57].

4.1 Flexible routing in SDN networks

Routing algorithms are implemented and communicated by each SDN controller to the SDN switches, which follow its instructions. Metrics such as hub count, delay, packet loss, bandwidth, jitter, and power consumption can be measured by SDN switches and sent to the controllers, which may use these metrics to determine the best routing paths and then install the flow forwarding rules in the data plane SDN switches. Indeed, the IoT[156] interacting Cloud Services [32] for the decision and control of the cyber-physical world create challenges for networks that achieve a better quality of service (QoS) and security and less energy

consumption, and can exploit the opportunities offered by machine learning [64, 98, 97]. These challenges can be met by SDN networks [176, 97], which offer greater flexibility and ease of implementation [78, 108].

These developments suggest that SDN is likely to become the preferred networking approach not only in core networks because of the centralized network intelligence and management that it enables but also for sub-networks of IoT devices and edge devices with specific QoS needs that can benefit from SDN programmability and flexibility. Thus, in [107], conventional routing protocols such as RIP, OSPF, EIGRP, and BGP, are compared with SDN with respect to convergence times after link failures, showing that SDN routing is better than conventional IP networks. Considerable work has also shown that SDN can select routing paths based on criteria such as quality of service (QoS) [161, 122, 68, 162, 79], while energy-aware SDN routing has also been discussed in several papers [168, 185, 11].

The scalability of SDN routers that conduct QoS routing has been studied in [128], where the authors propose an SDN-based scalable QoS routing scheme between autonomous systems. In [129], a survey of the scalability issues that arise when SDN's centralized scheme deals with relatively frequent path updates is conducted. Both hierarchical and concurrent (distributed) approaches are investigated to alleviate SDN controllers' additional workload. In recent work, [216] SDN is discussed as a means to choose the best paths based on a function of time-varying traffic in order to optimize the QoS metrics of interest. Other work [111] examines a broad class of QoS-based algorithms to assign paths to flows in SDN and analyzes the resulting performance. In addition, the work in [121] discusses the implementation of SDN based real-time QoS in industrial settings with mobile robots or palets, where motion and reliability requirements impose changes in paths to constantly meet real-time requirements. In [93], the use of AI-driven dynamic QoS routing in SDN is used to optimize QoS, reduce energy consumption, and improve security based on Autonomic Communications [64] and the Cognitive Packet Network algorithm [173].

However, in addition to scalability issues, QoS-driven SDN routing can create traffic and time-dependent changes in network topology and in the load and paths that are serviced by SDN switches. SDN network performance has been analyzed using queueing theory [170, 16, 178, 228] and network calculus [20, 19, 30], but these performance evaluations are based on the assumption that the network is in steady state—i.e., after a sufficiently long time—so that network metrics such as queueing delays, the length of packet queue buffers of SDN switches, and packet losses become stable (or time-independent). On the other hand, it is important to understand the time-dependent behaviour of SDN switches affected by changes in paths notified by the SDN controller. The controller can suddenly change the flows that an SDN switch receives, changing its input traffic. Furthermore, for a given switch some flows may be moved from one output port to another to comply with the new path that they must follow. These sudden changes will have performance consequences, including queueing delays and packet losses, which can only be understood via time-dependent transient analysis.

4.2 Time-dependent Modelling of a network of SDN switches

Consider a network of M stations with an arbitrary topology with routing probabilities $r_{ij}(t)$. We follow the approach of [101] developed for the steady-state network model then adapted to transient analysis in [66]. Additionally, we introduce time-dependent routing to model an SDN network.

The first step to solve the network model is to decompose the network—i.e., to determine the input

traffic parameters λ_i, C^2_{Ai} at every station i and then apply the single server model of the previous section to each station separately.

In the transient state, we should distinguish at any station i the input traffic $\lambda_{i-in}(t)$ and the output traffic intensities $\lambda_{i-out}(t)$:

$$\lambda_{i-out}(t) = [1 - p_{0i}(t)]\mu_i$$

which are different; $p_{0i}(t)$ denotes the probability that the station i is idle at time t, i.e., the diffusion process related to this station is inside the barrier at $x = 0$. The term $1 - p_{0i}(t) = \varrho_i(t)$ presents probability that the station i is busy and customers are leaving it with the rate μ_i.

The traffic equations balancing the flows of stations are:

$$\lambda_{i-in}(t) = \lambda_{0i}(t) + \sum_{j=1}^{M} \lambda_{j-out}(t) r_{ji}(t)\,, \qquad i = 1, \ldots, M, \tag{4.1}$$

where the first term λ_{0i} represents the traffic flows coming from the outside of the network directly to station i.

The routing probabilities $r_{ji}(t)$ change each interval Δ following the decisions of the controler, remaining constant inside the interval, and the flow parameters may change every interval $\delta < \Delta$; we assume $\Delta = n\delta$, in numerical examples below $n = 10$. This way all model parameters are constant witin intervals δ when the solution (3.20) is computed.

Denote by $f_{Aj}(x,t)$ and $f_{Bj}(x,t)$ the density functions of the interarrival and service times distributions at station j at time t. The pdf $f_{Dj}(x,t)$ of the interdeparture times from this node at time t may be expressed as:

$$f_{Dj}(x,t) = \varrho_j(t) f_{Bj}(x,t) + [1 - \varrho_j(t)] f_{Aj}(x,t) * f_{Bj}(x,t)\,, \quad j = 1, \ldots, M, \tag{4.2}$$

where * denotes the convolution with respect to x. The first term of the right side in (4.2) represents the interdepature times of packets when the node j is working, and the second term gives the interdeparture times when it is idle. The formula (4.2), known as Burke's theorem [31], is exact for Poisson input (the pdf of the idle period distribution that should be used in the second term of (4.2) is the same as $f_{Aj}(x,t)$) and approximate in other cases. From (4.2), we receive:

$$C^2_{Dj}(t) = \varrho^2_j(t) C^2_{Bj}(t) + C^2_{Aj}(t)(1 - \varrho_j(t)) + \varrho_j(t)[1 - \varrho_j(t)]\,. \tag{4.3}$$

where $C^2_{Dj}(t)$, $C^2_{Bj}(t)$, and $C^2_{Aj}(t)$ are time-dependent square coefficients of the variation in interdeparture, service, and interarrival times, respectively. Packets leaving the node j according to the distribution $f_{Dj}(x,t)$ choose any node i with probability $r_{ji}(t)$ and the times between two packets routed from node j to i has pdf $f_{ji}(x,t)$

$$\begin{aligned} f_{ji}(x,t) &= f_{Dj}(x,t) r_{ji}(t) + f_{Dj}(x,t) * f_{Dj}(x,t)[1 - r_{ji}(t)] r_{ji}(t) + \\ &\quad f_{Dj}(x,t) * f_{Dj}(x,t) * f_{Dj}(x,t)[1 - r_{ji}(t)]^2 r_{ji} + \cdots \end{aligned} \tag{4.4}$$

For example, a packet leaving station j goes to station i with probability $r_{ji}(t)$ or with probability $1 - r_{ji}(t)$ it goes elswhere but the second packet goes to i with probability $r_{ji}(t)$, hence the gap has has pdf $f_{Dj}(x,t) * f_{Dj}(x,t)$ with probability $[1 - r_{ji}(t)] r_{ji}(t)$, etc., or, after Laplace transform:

$$\begin{aligned} \bar{f}_{ji}(s,t) &= \bar{f}_{Dj}(s,t) r_{ji}(t) + \bar{f}_{Dj}(s,t)^2 [1 - r_{ji}(t)] r_{ji} + \bar{f}_{Dj}(s,t)^3 (1 - r_{ji}(t))^2 r_{ji} + \cdots \\ &= \frac{r_{ji}(t) \bar{f}_i(s,t)}{1 - [1 - r_{ji}(t)] \bar{f}_i(s,t)}\,, \end{aligned}$$

Then we compute the squared coefficient of variation:

$$C_{ji}^2(t) \quad = \quad r_{ji}(t)[C_{Dj}^2(t) - 1] + 1 \, ,$$

Hence:

$$C_{Ai}^2(t) = \frac{1}{\lambda_{i-in}(t)} \sum_{j=1}^{M} r_{ji}(t)\lambda_{i-out}(t)[(C_{Di}^2(t) - 1)r_{ji}(t) + 1] \; + \frac{C_{0i}^2(t)\lambda_{0i}(t)}{\lambda_{i-in}(t)} \, , \tag{4.5}$$

where the parameters λ_{0i} and C_{0i}^2 refer to the flows coming to station i from outside of the network.

The parameters of the input flow at station i are given by (4.1) and (4.5). Equations (4.3) and (4.5) form a system of linear equations yielding $C_{Ai}^2(t)$ and also the diffusion parameters $\beta_i(t)$, $\alpha_i(t)$ for every node i. At each interval δ, the functions $f_i(x, t; \psi_i)$ providing the queue length distributions at every station i for $t \in \delta$ are computed. Their values at the end of the interval yield, among others, the current utilizations ϱ_i used to determine the flow parameters and diffusion parameters for the next interval δ.

The pdf $f_{Ri}(x, t)$ of the time-dependant response time (waiting time plus service) is determined using the first passage time from the end of the queue to zero, as defined by Equation (3.12). If $f_{Ri}(x, t)$ is the response time pdf at node i, then the response time pdf $f_R(x, t)$ for the path $1, \ldots, n$ of n stations is:

$$f_R(x, t) = f_{R1}(x, t) * f_{R2}(x, t) * f_{R3}(x, t) * \cdots * f_{Rn}(x, t),$$

or:

$$\bar{f}_R(x, s) = \prod_{i=1}^{n} \bar{f}_{Ri}(x, s).$$

The loss probability $p_{loss}(t)$ for same entire path may be computed from:

$$1 - p_{loss}(t) = (1 - p_{N1}(t))(1 - p_{N2}(t))(1 - p_{N3}(t)) \ldots (1 - p_{Nn}(t)) \tag{4.6}$$

where $p_{Ni}(t)$ is the probability that the queue at station i is saturated at time t—i.e., the diffusion process for this station is at time t at the barrier $x = N$.

4.3 Transient analysis of the influence of changing forwarding flow rules on the SDN data plane

Consider a network composed of four SDN switches, $S1$–$S4$; see Figure 4.1. Their parameters are the same as the switch in Example 1, except for the SDN switch $S4$, which is twice as fast. Therefore $\mu_1 = \mu_2 = \mu_3 = 2628.8$ packets/sec while $\mu_4 = 5257.6$ packets/sec. At all the switches, the squared coefficient of variation of service time is identical to the value $C_{Bi}^2 = 0.33$.

Similarly to Example 1, the network's performance is investigated during 1 second. Host 1 is sending packet flows of intensity λ_{01} to Host 2, and the traffic rate is changing in the range 500–2500 packets/sec, as shown in Figure 4.2. Host 4 generates traffic at rate λ_{02}, as shown in Figure 4.2, which is forwarded to Host 2 via the SDN switches $S2$ and $S4$.

As in Example 1, the squared coefficient of variation of interarrival times in the flows λ_{01} is $C_{A1}^2 = C_{01}^2 = 1.02$, or $C_{A1}^2 = 4.08$ or $C_{A1}^2 = 8.16$. The second input traffic λ_{02} at $S2$ has only one parameter $C_{A2}^2 = C_{02}^2 = 1.02$.

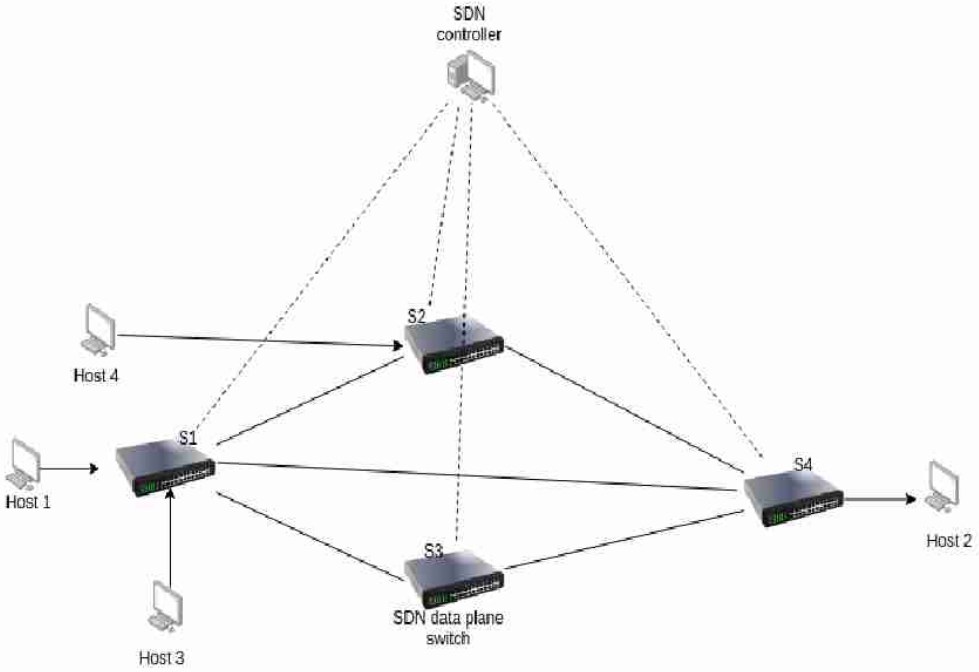

Figure 4.1: The example network being considered.

The SDN controller alters—if needed—the routing to balance the load of nodes every 100 msec; in this example, it refers to the routing probabilities r_{12} and r_{13}; see Figure 4.3.

Figures 4.4 and 4.5 illustrate the decomposition of the network model. They present the flows $\lambda_i(t)$ given by Equation (4.1) and the squared coefficients of variation $C^2_{Ai}(t)$ received from Equations (4.3) and (4.5). The service times at the stations have relatively small squared coefficients of variation $C^2_{Bi} = 0.33$. Therefore, the important variability of the first flow entering the network is reduced at the network interior, as defined by Equation (4.3); see Figure 4.5.

The transient solution of diffusion equations is computed in intervals of the length $\delta = 10$ msec—i.e., we have 100 intervals with fixed diffusion parameters; at the end of each δ, the Equations (4.1) and (4.5) are solved to determine the new parameters of flow for the single-station models in the next interval. The diffusion density function obtained for any station i at the end of an interval gives the initial conditions for the diffusion equation at the next one.

The curves in Figure 4.6 compare the loss probability (note here the minimal values computed by the model), and, in Figure 4.7, the mean queues for all four stations, in case of $C^2_{A1} = 1.04$. We may observe the changes in mean queues in $S2$ and $S3$ due to load balancing after the second flow becomes active. Observing the mean queues at $S1$ and $S2$, we can see that the transient periods may be longer than the time between the controller's decisions. As noted earlier, the length of the transient time increases with a load of a station and the variability of the input flow. For greater variabilities of the first flow, the path $S1 - S3 - S4$ becomes saturated; see Figure 4.8. This happens due to saturation in $S4$, as shown in Figure 4.9.

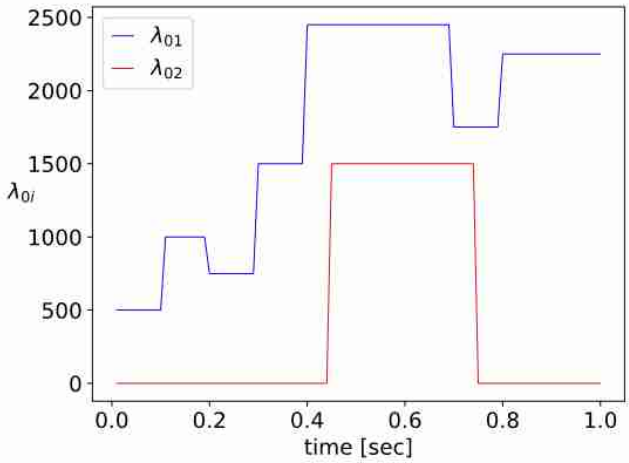

Figure 4.2: Input flows $\lambda_{01}(t)$, $\lambda_{02}(t)$, time in seconds.

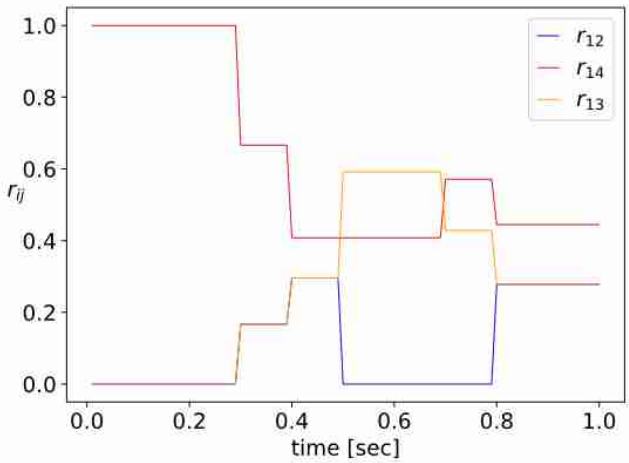

Figure 4.3: Routing probabilities $r_{12}(t)$, $r_{13}(t)$, $r_{14}(t)$.

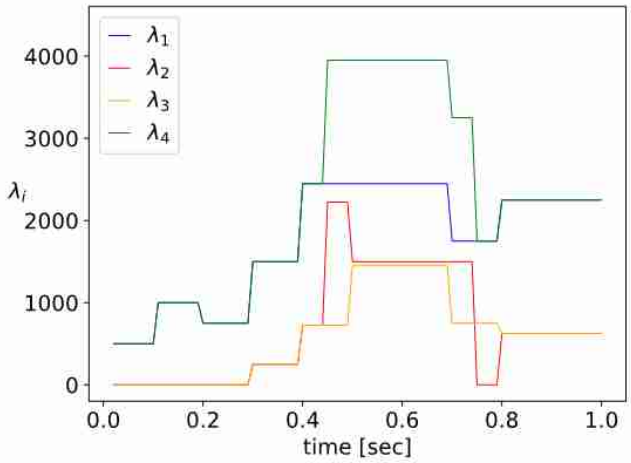

Figure 4.4: Input flows $\lambda_i(t)$ for stations $S1 \ldots S4$.

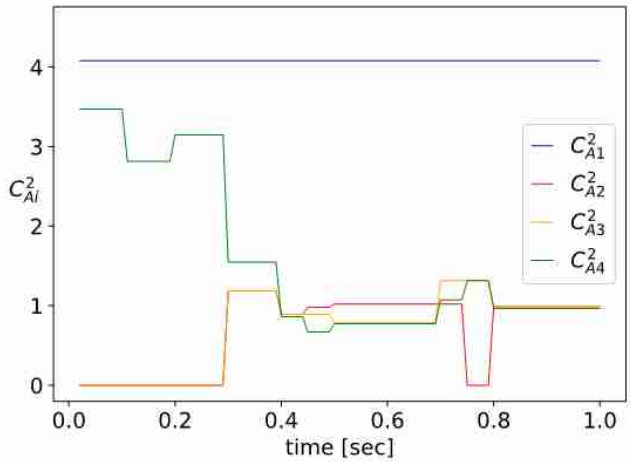

Figure 4.5: Squared coefficients of variation $C_{Ai}^2(t)$ for stations $S1 \ldots S4$.

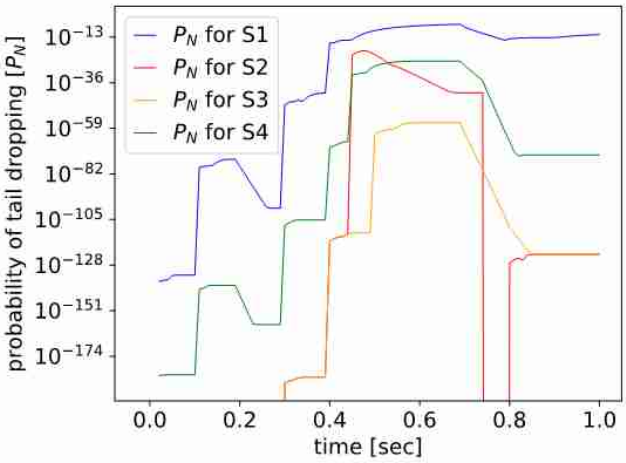

Figure 4.6: Stations $S1$, $S2$, $S3$, $S4$: $p_N(t)$, $C_{A1}^2 = 1.02$.

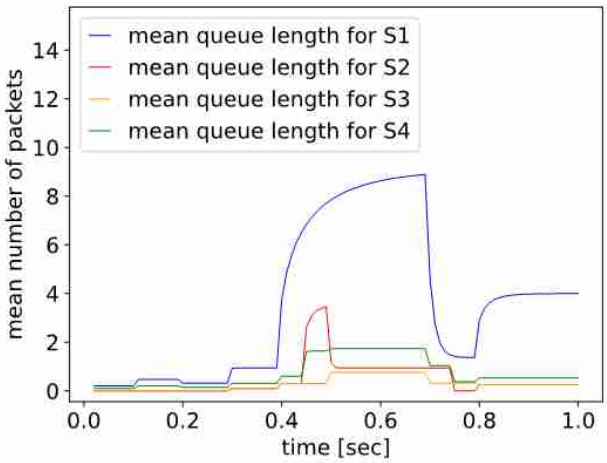

Figure 4.7: Stations $S1$, $S2$, $S3$, $S4$: mean queue, $C_{A1}^2 = 1.02$.

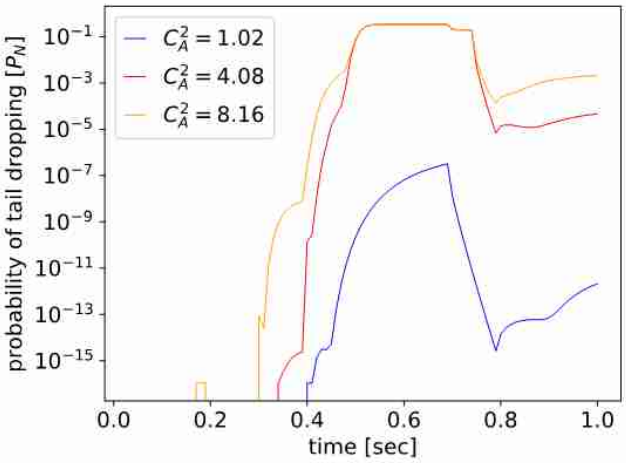

Figure 4.8: Total loss probability for path $S1 - S3 - S4$ for different C_{A1}^2.

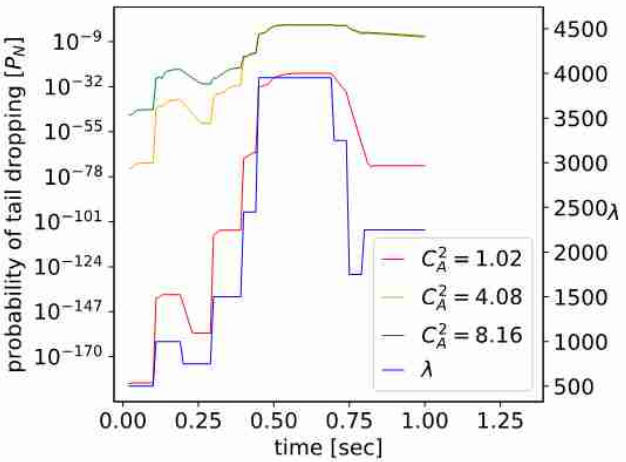

Figure 4.9: Station $S4$: $p_N(t)$ for different C_{A1}^2.

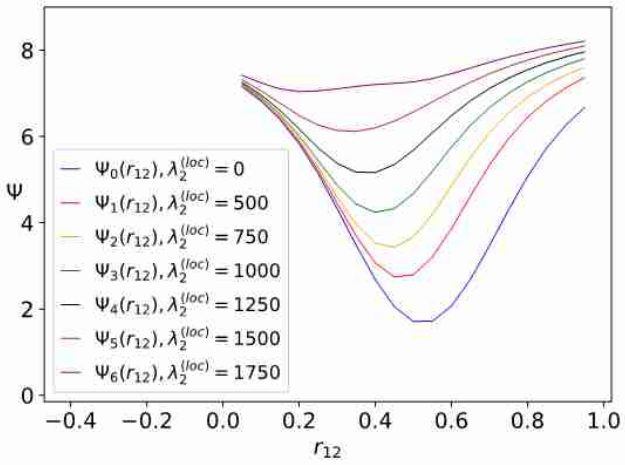

Figure 4.10: Mean backlog Ψ during Δ as a function of routing probabilities r_{12}, $r_{13} = 1 - r_{12}$, each curve corresponds to a different flow coming from Host 4 to S2.

4.4 A simplified route optimisation example

Let us also consider a **simple example of optimization**. Suppose, as previously, that station S1 is forwarding a flow λ_{01} packets to nodes S2 and S3. Station S2 is additionally receiving from Host4 a flow of $\lambda_{02}^{(loc)}$ packets. The controller is changing routing every $\Delta = 100$ msec and needs to determine the routing probabilities for the nearest Δ, knowing the current parameters of flows at the beginning of the interval, as well as the current queue distributions at S1, S2, and S3 representing previous behaviour of the network. The goal is to minimize the mean backlog Ψ at S2 and S3 during Δ:

$$\min_{r_{12}, r_{13}} \left\{ \Psi = \frac{1}{\Delta} \int_0^{\Delta} \left[E[N_2(t)] + E[N_3(t)] \right] dt \right\}.$$

We compute $E[N_2(t)]$, $E[N_3(t)]$ for $t \in \Delta$ and minimize Ψ by the choice of r_{12}, $r_{13} = 1 - r_{12}$; see Figure 4.10.

4.5 Conclusions

The advent of SDN allows the implementation of smart adaptive routing [104], which changes network paths so that new connections may be established and inactive may be removed, as well as to deal with changes in traffic loads and incidents that affect network security. This leads to an interesting paradigm shift in network modelling, which has traditionally addressed "long term" behaviours and computational methods which are appropriate for steady-state analysis. However, when SDN intervenes dynamically to change paths and traffic levels, the network is seldom at a steady state, and optimization must take transients into account.

Therefore in this chapter we have used diffusion approximation modelling for the performance evaluation of a network of SDN switches, that considers both steady-state and transient analysis. We have shown

how changes in routing or forwarding decisions by the SDN controller can influence performance parameters such as delay, queue size, and packet loss probability in the transient state. Our results indicate that this method is computationally operational and can provide useful quantitative results for models with realistic parameter values.

Our analysis captured the interactions among the main parameters of the network, and numerical examples display the dependence of the queue lengths, queueing delays and their dynamics as a function of the changing flow intensity and variance of interarrival times. Our approach also confirms that transient periods play a significant role in the performance of SDN networks, and that they will be useful to analyze much larger networks in future work.

While the performance evaluations performed in this chapter are purely numerical, and based on diffusion approximations models that have been widely validated by simulations [48, 60, 61], in future work we intend to use network emulation tools such as Mininet with real traffic, as well as experiments on a SDN test-bed, to study the influence of time dependent forwarding decisions on the main performance metrics of large SDN networks.

Chapter 5

Modelling of the Energy Depletion Process and Battery Depletion Attacks for Battery-Powered Internet of Things (IoT) Devices

The complexity of battery-powered autonomous devices such as Internet of Things (IoT) Sensor Nodes or Unmanned Aerial Vehicles (UAV) and the necessity they ensure an acceptable quality of service, reliability, and security, have significantly increased their energy demand. These devices are often powered by small batteries with limited energy content. These devices are vulnerable to battery depletion attacks designed to completely deplete the energy stored in the battery and eventually shut down the device. Thus, battery and energy consumption models are required when designing these systems to ensure that they operate within a reasonable time before requiring battery replacement.

In this chapter, we apply a diffusion or Brownian motion process to model the energy depletion process of a battery of an IoT device. We use the model to obtain the probability density function, mean, variance, and probability of the lifetime of an IoT device. Also, we study the influence of the active power consumption, sleep time, battery capacity on the probability density function, mean, and probability of the lifetime of an IoT device. We use numerical examples to study the influence of battery depletion attacks on the distribution of the lifetime of an IoT device. We also introduce in our model an energy threshold after which the battery of the device should be replaced to ensure that the battery is not completely drained before it is replaced. A portion of the material presented in this chapter was published in [58, 53].

5.1 Energy consumption models for IoT devices

An IoT device consists of the sensing (data acquisition unit), the actuator unit, processing, and storage unit, the communication module, the security module, the power supply unit, and the energy storage system. The sensors capture the desired physical data from the environment translate it into digital information, which may be partially processed by the IoT device or transmitted to fog computing servers for lightweight analysis or to a cloud computing data centre for advanced analysis. The analysis results could be sent back

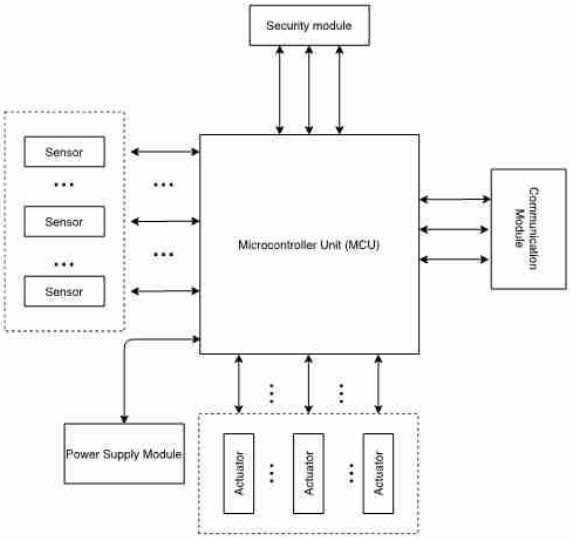

Figure 5.1: The architecture of an IoT device.

to the IoT devices attached to actuators to control cyber-physical systems. Actuators receive digital signals translate them into physical actions to drive or manipulate cyber-physical systems. A simplified architectural model of an IoT deployment is shown in figure 1.1.

5.1.1 Power consumption of an IoT device

A simplified abstract architecture of an IoT device is shown in figure 5.1. The total power consumption of an IoT device is the sum of the power consumption of various IoT components, including the sensing units, the actuator unit, the microcontroller units, the communication unit, the communication unit, and the security unit. The average total power consumed by an IoT device is

$$P_D = R_{ACT}.(P_{SAU} + P_{MCU} + P_{Comm} + P_{SEC}) + R_{SLEEP}.P_{SLEEP} \tag{5.1}$$

where P_{SAU} is the average power consumed by the sensing and actuation units, P_{MCU} is the average power consumed by the microcontroller unit, P_{Comm} is the average power consumed by the communication module, P_{SEC} is the average power consumed by the security module, P_{SLEEP} is the average power consumption of the consumption of the IoT device in the sleep mode. Also, R_{ACT} is the fraction of time that an IoT device spends in the active mode, R_{SLEEP} is the fraction of time the device spend in the sleep mode, $R_{ACT} + R_{SLEEP} = 1$.

The authors in [203] modelled the energy consumption of an IoT node. The authors modelled the energy consumed during the sensing, processing, and communication processes in an IoT device. A significant proportion of the energy is used for the transmission and reception of IoT packets, and it depends on the IoT packet size, channel capacity, and the environmental factors that influence signal propagation through wireless transmission media. The authors in [117] proposed an analytical framework for modeling the energy

consumption of an IoT device for a cellular IoT network (e.g., NB-IoT) and determined the lower bound for the energy consumption of the IoT device.

The most important requirements to consider when designing and planning IoT devices and networks include cost, lifetime [174], and reliability. The lifetime of an IoT device depends on the capacity of the battery used to power the IoT device and on the energy drawn from the battery to power the IoT device. The capacity of the battery increases with cost, and some IoT applications require the deployment of tens, hundreds, or even thousands of battery-powered IoT devices, making cost a very important constraint to consider when designing and deploying IoT devices and networks. The lifetime of the device is the time required to completely deplete the energy stored in the battery and shut down the device. It could be increased by either increasing the battery's capacity or by reducing the power consumed by the IoT device. The energy consumption of the IoT devices is kept at a minimum level by using an energy-efficient microcontroller, using a low-power communication protocol that keeps the device in the sleep mode (energy saving state) most of the time, and by using lightweight energy-efficient security mechanisms. The authors in [179] conducted a comprehensive measurement study of the energy consumption of NB-IoT devices to determine the factors that influence energy consumption and battery energy depletion. The authors found out that NB-IoT's energy consumption largely depends on the communication model, signal quality, use of energy-saving enhancements, and packet size.

5.1.2 Modelling of the expected lifetime of an IoT device

The lifetime of an IoT device can be estimated using empirical methods or using mathematical modelling. Empirical approaches require time and resources to set the testbed for experiments. The authors in [154] presented the first attempt to empirically estimate the lifetime of a battery-powered NB-IoT device using power consumption measurements. The authors in [229] proposed a modelling and experimental framework for the estimation of the lifetime of battery-powered NB-IoT and LTE-M devices using energy consumption profiles for these devices. The expected lifetime of an IoT device is [229].

$$L = \frac{B.SF_{bat}}{P_D} \tag{5.2}$$

SF_{bat} is the battery safty factor which account for self-discharging and B is the energy rating of the battery (in Wh). The average power P_D is required to power all the components of an IoT device could be estimated emperically or using theoretical power profile models for IoT. Since the IoT device can either be in the active mode (data acquisition, processing, security, and communication) or in the sleep mode (energy saving state with the radio transceiver turned off), the expected lifetime of the IoT device is

$$L = \frac{B.SF_{bat}}{R_{ACT}.P_{ACT} + R_{SLEEP}.P_{SLEEP}} \tag{5.3}$$

where $P_{ACT} = P_{SAU} + P_{MCU} + P_{Comm} + P_{SEC}$ is the average power consumed by the IoT in the active mode. By reducing the fraction of time spent by the device in the active mode or by increasing the fraction of time spent by the device in the sleep mode, the device lifetime can be prolonged. If the fraction of time that an IoT device spends in the active mode is $R_{ACT} = \frac{T_{ACT}}{T_D}$ and the fraction of time spend in the sleep mode is $R_{SLEEP} = 1 - R_{ACT} = \frac{T_{SLEEP}}{T_D}$, then equation (5.4) becomes:

$$L = \frac{T_D.B.SF_{bat}}{T_{ACT}.P_{ACT} + T_{SLEEP}.P_{SLEEP}} \tag{5.4}$$

Where, T_{ACT} is the total time spent by the IoT device in the active mode, T_{SLEEP} is the total time spent in the active mode by the device, and $T_D = T_{ACT} + T_{SLEEP}$. If the battery safty factor, $SF_{bat} = 1$, then the expected lifetime of the IoT device given in equations (5.2–5.4) becomes

$$L = \frac{T_D.B}{T_{ACT}.P_{ACT} + T_{SLEEP}.P_{SLEEP}} \tag{5.5}$$

which is the well-known formula for expected lifetime of an IoT device e.g., see [154, 179].
We know that the average total power delivered to the IoT device from the battery, $P_D = P_{ACT} + P_{SLEEP}$ can be expresses in terms of the average total current drawn from the battery, I_D and the battery output voltage, v_o as $P_D = I.v_o$. Also, the energy rating, B (in Wh) of the battery can also be expressed in terms of the charge rating, Q (in Ah) of the battery and the battery output voltage, v_o as $B = Q.v_o$. Therefore, by dividing the numerator and denominator of equation (5.2–5.5) by the output voltage v_o, the lifetime of the IoT device in terms of the charge rating and the total average current delivered to the IoT from the battery is

$$L = \begin{cases} \dfrac{T_D.Q.SF_{bat}}{T_{ACT}.I_{ACT} + T_{SLEEP}.I_{SLEEP}} & \text{for} \quad SF_{bat} > 1 \ , \\ \dfrac{T_D.Q}{T_{ACT}.I_{ACT} + T_{SLEEP}.I_{SLEEP}} & \text{for} \quad SF_{bat} = 1 \end{cases} \tag{5.6}$$

Where $I_D = R_{ACT}.I_{ACT} + R_{SLEEP}I_{SLEEP}$, I_{ACT} is the average current delivered to the IoT device from the battery in the active mode, and I_{SLEEP} is the average current delivered to the IoT device from the battery in the sleep mode. One of the misconceptions when choosing batteries for IoT devices is the assumption that batteries with a higher charge rating (in mAh) will guarantee a long lifetime for the IoT device. For alkaline batteries, which are the most cost-effective and commonly available batteries used to power electronics devices, the output voltage v_o degrades very quickly, shortening the time required to drain the battery completely or the lifetime of the IoT device [259]. The authors in [9] highlighted the difference between the battery capacity, Q in Ah, and the battery capacity, B in Wh. The battery capacity Q is the total amount of electricity generated due to electrochemical reactions in the battery, while B is the total energy that the battery can deliver during the discharge process. The relationship between the battery capacity, Q, and the discharge current may not be linear as shown in equation (5.6), but is exponential according to Peukert law [262]. The relationship between battery capacity in Ah Q and the average discharge current I_D, and the battery discharge time (which is equivalent to the lifetime of the IoT device, L) can be deduced from the Peukert law [262, 9] as

$$\begin{aligned} L &= Q.I_D^{-k} \\ &= \frac{Q}{[R_{ACT}.I_{ACT} + R_{SLEEP}I_{SLEEP}]^k} \end{aligned} \tag{5.7}$$

where k is Peukert constant, and its value lies between 1.1 and 1.3. Therefore, it is preferable estimate the expected lifetime of the IoT device using the energy rating (in Wh) (e.g., (5.2–5.5)), rather than the charge rating as in equation (5.6).

Therefore to estimate the expected lifetime of an IoT device is required to estimate the average power consumption of the device (the rate at which energy is drawn from the battery), and the battery rating or the initial amount of energy in the battery (initially the battery should be charged to full capacity). Using measurements to benchmark the energy consumption of IoT devices before they are used to estimate the lifetime of an IoT device is important. Any hardware and software implementation changes or energy depletion attacks that alter the supposed sleep time of the device may significantly reduce the expected lifetime

of the IoT device determined during IoT network design and deployment.
On the other hand, mathematical modelling provides a faster technique of understanding the relationship between the battery parameters, the power consumption model of the IoT device, and the lifetime of the IoT device. Since it is possible that the energy drawn from the battery could vary stochastically, stochastics models such as Markovian models, fluid flow models, and diffusion approximation models have been used to estimate the model of the battery of sensor devices and to estimate their lifetime.

5.1.3 An overview of stochastic modelling of the battery of computer systems

A useful but seldom used approach in the analysis and optimization of energy, and more broadly for the joint optimizaton of energy and quality of service in computer systems and networks, named "energy packets" was introduced by Gelenbe in 2011 and 2012 [87, 88, 89]. It conveniently represents energy in discrete units, where an energy packet is the minimum amount of energy required to transmit a single data packet or process a single job. This approach was initial applied to the optimization of power flow in multiple node computer networks [100] and joint work and energy in computer systems [90], and then to study the effects of energy leakage in battery powered devices [95]. The model was applied to the study of sensor nodes [91], and also to battery attacks in [73, 72].

when the energy is quantised, Markovian stochastic models can be used to model the energy storage and consumption process. In this case, the state probabilities at time t represent the amount of energy stored present in the battery at time t. The authors in [125] developed a mathematical framework for modelling the charging and discharging of the battery of a nanosensor device. The authors represented the dynamic changes in the battery's energy content using a Markovian process and then computed the state probabilities of the amount of energy present in the battery (the energy state of the battery). One of the limitations of Markovian models is the assumption that the rate at which energy is drawn from the battery is exponentially distributed, which is not a realistic assumption of the IoT energy consumption patterns.

Since energy is a continuous quantity, the changes in the amount of energy in the battery could be considered to be analogous to the changes of a fluid in a reservoir, and hence modelled using fluid flow models. The authors in [83] proposed an analytical model of a battery based on the fluid flow queueing model. The authors modelled the battery as a charge or energy reservoir where the charge gets accumulated or depleted over time. By considering that the charge available in the battery at time t is analogous to the fluid available in a reservoir, the authors used fluid flow analytical methods to determine the cumulative distribution function and the mean of the time required for the battery to be completely discharged. The authors in [123] proposed a fluid queue model for the representation of the dynamic changes in the energy content of a battery and then used it to determine the time required to completely depletes the the energy of the battery. The authors in [243] proposed a Markov fluid queue model for the battery of an energy harvesting IoT device. The authors used their model to compute the probability that the battery's energy level hits zero for the first time within a given finite time horizon. Fluid flow models capture the mean changes in the amount of energy present in the battery but not the variance.

Recently, diffusion models have been proposed as a stochastic model for the battery of an IoT device. The advantage of using diffusion models the time evolution of the energy stored in the battery traditional queueing theoretic models and fluid flow models is that it takes into account fluctuations in the amount of

energy harvested from the environment and the fluctuations in the amount of energy drawn from the battery (energy consumed). The authors in [5] proposed a diffusion process to model the process of energy supply from energy sources, storage, and consumption for a battery of a sensor node. The authors derived some performance metrics such as the average time until the node is shut down when the energy stored in the battery is completely depleted, for a given workload and energy harvesting characteristics, battery capacity. The authors in [4] proposed a pure diffusion model to represent the time evolution of the discharging and charging process of the battery of a wireless sensor node. The authors derived some performance metrics such as the average amount of energy present in the battery at time t and the failure rate of the wireless sensor node when the battery is completely discharged. They also presented the steady-state solution of the model that they proposed. It is sometimes important to obtain the transient distribution of the amount of energy stored in the battery at time t and the distribution of the discharging time, which the authors in [5, 4] did not consider. The authors in [33] applied diffusion approximation to analyse the transient evolution of the charging and discharging process of the battery that is supplied by renewable energy and then used to supply network nodes in a wireless mesh network.

5.2 Energy depletion attacks in IoT networks

Energy depletion attacks are attacks that are designed to exhaust the energy stored in the battery of an IoT device. Reliable security mechanisms are complex and require reasonable computing resources, memory, and energy. The limited resources (e.g., memory, processing power, bandwidth, and battery) in IoT devices make it challenging to implement reliable security mechanisms in devices and IoT networks. Also, some IoT device manufacturers do not implement security futures in their devices to keep the cost low (to be competitive in the market) and to speed up their manufacturing process. However, much effort has been made to implement lightweight security mechanisms that optimise the limited IoT resources while still providing the required security for IoT devices.

Energy depletion attacks are designed to increase the energy consumption of IoT devices, which rapidly depletes the energy stored in the battery and eventually shuts down the IoT device. The authors in [112] presented types of battery deletion attacks which include: service request power attacks and benign power attacks. In the service request power attack, the attacker continuously sends service requests to the target device, which keeps the IoT device awake for longer periods and rapidly depletes the battery energy. In a benign power attack, an attacker forces a compromised IoT device to execute energy-demanding tasks which rapidly depletes the battery energy continuously. The authors also proposed a network-based intrusion detection and prevention technique to detect and prevent battery depletion attacks. Therefore, energy depletion attacks are designed to increase the fraction of time that an IoT device spends in the active mode or reduce the fraction of time that the device spends in the sleep mode.

The secure communication process is the most energy-demanding process in an IoT device. To reduce the energy consumption of IoT devices, low power communication protocols that keep the IoT packet size as small as possible and keep the device in sleep mode for as long as possible are used. Some energy depletion attacks are designed to reduce the sleeping time of the IoT device. Adjusting some of the network parameters, such as the duty cycle, data rates, and the packet size can significantly reduce rapidly depletes the battery of an IoT device and reduce its lifetime [187, 180]. Also, by inducing the device to transmit

useless packets repeatedly, the device rapidly depletes the energy of the battery [200]. A malicious attacker could prevent an IoT device from entering sleep mode by manipulating its contention window size [34], sending massive amounts of packets to create more collisions to the ongoing transmissions [12], and by overwhelming the IoT device by flooding it with packets without payload.

Energy depletion attacks could reduce the lifetime of IoT devices from years to days [204], could also result in the shutting down of an entire IoT network. The authors in [227] studied the impact of battery training attacks such as "hello" flooding, stretch attacks, and versioning on the energy consumption of IoT devices. The authors found out that versioning is the most severe as it draws a lot of energy from the battery, followed by packet flooding and "hello" attacks. The authors in [120] analysed the impact of battery drain or energy depletion attacks on IoT devices. The authors designed and conducted DoS service attacks such as "hello" flooding and version number modification to demonstrate the impact of these attacks on the energy consumption of the IoT devices and rendered some of them unreachable. A similar demonstration was shown in [200], where the authors configured some malicious nodes to intentionally generate and send large amounts of packets to legitimate nodes to excessively consuming the energy resources of the nodes found along the forwarding path.

Unmanned Arial Vehicles (UAVs) are increasingly being adopted for commercial applications [244] such as agriculture, environmental management, supply chains, law enforcement, surveillance, photography [138, 139, 140], and were recently used for deliveries during the COVID-19 Pandemic [75] and to enforce the restrictions designed to slow the spread of the COVID-19 virus. A UAV could be considered an IoT system, especially when connected to the internet (e.g., used as an access point in some IoT deployment requiring a temporary sensor network for a few hours). Like traditional wireless sensor devices, UAVs are powered by batteries, making it difficult to implement sophisticated, reliable security mechanisms. AS a result, the security mechanisms implemented in UAVs are relatively weak and could be easily compromised. One such possible attack is the energy depletion attack designed to take control of the UAV and cause it to perform manoeuvres that consume more energy and, therefore, rapidly drain the drone's battery. Although most UAVs have battery monitoring and management systems that ensure that the drone does not crash due to energy outages by initiating a return-to-home (RTH) mechanism. The RTH mechanism ensures that the drone returns home with a reasonable amount of energy to ensure a safe landing. However, battery depletion attacks could still cause the drone to crash while still executing the RTH procedure, which could be catastrophic and could result in a lawsuit. The authors in [63] presented a framework for the simulation and assessment of battery depletion attacks on UAVs in crisis management systems.

It is important to have techniques to detect energy depletion attacks and mitigate their impact. The majority of attack detection systems used to detect energy depletion attacks in IoT networks are based on monitoring traffic characteristics and QoS metrics. The authors in [120] proposed an Intrusion Detection System (IDS) that detects the presence of energy depletion attacks in IoT networks by monitoring by monitoring packet characteristics and QoS metrics such as packet sending rate, packet interval, and the Receive Signal Strength (RSS). They also demonstrated the use of firewalls to detect traffic coming from intrusions. It should be noted that not all energy depletion attacks degrade the quality of service. Some energy depletion attacks may trigger an increase in the quality of service while gradually increasing the rate at which energy is drawn from the battery until the battery becomes empty and the device(s) are shut down [215]. The authors in [199] proposed a lightweight anomaly detection model against energy depletion attacks on IoT networks.

The model proposed by the authors is based on the analysis of statistical distance metrics to differentiate between the normal and abnormal energy consumption in IoT devices.

Some attempts have been made to model the impact of energy depletion attacks on the performance and safety of battery-powered IoT devices. The authors in [215, 214] modelled the impact of energy depletion attacks on the performance of IoT devices. They investigated the impact of increasing the energy consumption due to energy depletion attacks on the system survivability metric for an IoT device under an energy depletion attack. They considered the survivability metric the Mean Time To Failure (MTTF). The authors discussed the impact of energy depletion attacks which do not degrade the QoS or may improve the QoS while gradually draining the battery of the IoT device. The model proposed by the authors is based on the pure death Markovian process, which assumes that there is no continuous supply of energy into the battery, but the energy consumption process is exponentially distributed. The proposed model is limited by the Markov assumption as realistic energy consumption distribution for IoT devices deviates from the Markovian distribution.

5.3 Analysis of Ghost Energy Depletion Attack on an IoT Network

A ghost energy depletion attack (GEDA) is one in which an adversary masquerades as a trusted device and compels other IoT devices within the network to perform unnecessary computational and communication operations to quickly deplete the energy stored in the battery of the victim devices and eventually shut down the devices. There are two main forms of ghost energy depletion attacks which are the high computational load on device GEDA and the MAC misbehaviour GEDA. In the high computational load GEDA, the adversary overwhelms its victims with bogus messages to quickly drain the energy stored in their batteries. Even though it is easier to detect these kinds of attacks with attack detectors, a ghost attacker may cleverly conduct such an attack by sending the messages at different times or by sending them at different addresses to a subset of victim devices in its range [34]. In a MAC misbehaviour GEDA, a ghost attacker deliberately abuses the MAC protocol (e.g., CSMA/CA protocol) to create collisions on the shared wireless channel to cause other devices within its interference range to consume more energy (quickly draining their batteries) and to deprive them of accessing the channel. To analyse GEDAs, it is essential to take note of the factors that influence such attacks which include:

1. The energy consumption of the various hardware components of the IoT device (e.g., microcontrollers, radio transceivers, sensors, actuators, and other electronic components). Energy-demanding microcontrollers and radio transceivers will consume more energy than energy-efficient ones and will drain the energy stored in the battery of the IoT device quickly during a battery depletion attack.

2. The energy capacity of the IoT battery. For a given IoT device, the lifetime of an IoT device depends largely on the energy capacity of its battery. With a high-capacity battery, the lifetime of the device could be longer. A device with a small battery capacity will easily be shut down by a ghost energy depletion attack.

3. Frequency of data collection (sensing), actuation (where necessary), processing, and communication (reception and transmission of information). The more frequent, the device sensing, processing, actuation, processing, and communication operations, the higher the energy consumption of the device.

A ghost attacker could compel victim IoT devices to perform such operations more frequently than during normal operations.

4. The MAC protocol in the link layer. A ghost attacker can abuse the MAC protocol in the link layer to create collisions, thus, increasing the energy consumption of the devices sharing the channel with it. Using a collision-free protocol at the link layer could reduce this kind of attack.

5. The cryptographic algorithm is implemented on the IoT device to encrypt and decrypt information. The energy required to encrypt or decrypt a packet depends on the number of microcontroller clock cycles required to execute the algorithm (encryption or decryption) and on the average current drawn by each cycle. Therefore, with information about the number of cycles required to execute the algorithm, the current drawn in each cycle, the clock frequency of the microcontroller, and the operating voltage of the microcontroller, the energy required to execute an encryption or decryption algorithm on an IoT device can be estimated. The more sophisticated or computationally intensive the cryptographic algorithm, the more quickly it can be leveraged by an attacker to drain the energy of an IoT device.

6. The packet sizes. The longer the packet size, the more energy is required to transmit the packet and the longer the time required to transmit the packet. A ghost attacker could decide to be created longer packets that take too long to transmit, causing other IoT devices sharing the channel with it to experience more collisions.

5.3.1 Analysis of high computational load ghost energy depletion attack

Consider an IoT device, i in an IoT network with N nodes. Suppose that the device is working in a duty-cycling mode with a duty cycle of $D = \frac{\tau}{T}$, where is the duration of the active period and T is the length of the cycle. Within the active period, the device can receive and decrypt a packet or encrypt and transmit a packet. If the device completes the reception or transmission of packets and the active period is not yet finished, the radio and the microcontrollers could be switched to a low-power mode. If no packet arrives or there is no packet to transmit, the radio is turned off and the microcontrollers are switched to a deep sleep mode (where it consumes a very small amount of power). After a period T_s, the device wakes up again to either receive or transmit packets.

Suppose that a ghost attacker crafts bogus packets and send them to the victim device to force it to spend energy to receive and perform security checks (e.g., access control, message integrity checks, and decryption). The access control mechanism is based on the principle that after receiving a packet, the device compares its source address with a list of valid addresses. If there is a match, the packet is accessed; otherwise, it is rejected. If the ghost attacker can masquerade as a legitimate device, its packets might be accepted by the victim device but after performing a message integrity check or decrypting the message, it will realise that it will fail. Although the security checks eventually failed and the packet from the ghost attacker dropped, the device must have spent a significant amount of energy performing some computation.

Suppose that a ghost attacker sends bogus packets to the victim device at a mean rate of κ and the mean arrival rate of packets to the victim device from both attack and legitimate sources is $\gamma = \kappa + \nu$, where ν is the mean arrival rate of normal packets from legitimate sources. If within a given active period, the

device receives N_r packets (both attack and normal ones) and performs security checks for these packets, then the energy consumed by the microcontroller during the process of receiving the packet and executing the security algorithms to perform the security checks is [34]:

$$E_{comp}^{rx} = N_r(T_{dec}P_{MCU}^{a} + T_{rx}P_{MCU}^{i}) \tag{5.8}$$

Where T_{dec} is the time required to perform the security checks (access control, decryption, and integrity verification) for a given packet, which depends on the security mechanism that is implemented, T_{rx} is the time required to receive a packet (which depends on the size or length of the packet), P_{MCU}^{a} is the power drawn by a microcontroller unit (MCU) when it is in the active mode, and the power drawn by the MCU when it is in the idle mode. The energy required to execute the algorithms required to perform the security checks (including decryption and MIC verification) after receiving the packet can be given by

$$E_{sec} = \frac{N_c I_{sec} V_{sec}}{f} \tag{5.9}$$

where N_c is the number of clock cycles required to perform the security checks, I_{sec} is the average current drawn by each clock cycle, V_{sec} is the operating voltage of the MCU, and f is the clock frequency of the MCU. Therefore, the more attack packets that are successfully received by the victim device, the higher the amount of energy wasted executing the security check algorithms (wasting MCU clock cycles to perform security computations). The energy consumed by the radio module in receiving both the attack and normal packets within a given active period is

$$E_{rx} = \begin{cases} N_r(T_{dec} + T_{rx})P_{rx} & \text{for} \quad N_r(T_{dec} + T_{rx}) \geqslant \tau \ , \\ \tau P_{rx} & \text{for} \quad otherwise \end{cases} \tag{5.10}$$

where P_{rx} is the power required to receive a single packet.

Suppose that an attacker compromises an IoT device, and then reconfigures it to perform more sensing (measurement) operations more frequently than required. The packets that belong to the extra measurement can be considered attack packets because the device spends energy to sense, encrypt and transmit the packets. Suppose that the victim device perform N_t number of transmission (including the transmission of packets from necessary and unneccessary measurements), then the energy consumed by the MCU in performing cryptographic operations (including encryption of the packets) and transmitting the packets is given by

$$E_{comp}^{tx} = N_t(T_{enc}P_{MCU}^{a} + T_{tx}P_{MCU}^{i}) \tag{5.11}$$

Where T_{enc} is the time required to encrypt a packet and T_{tx} is the time required to transmit a packet. The energy required to perform the cryptographic operations (including encryption) before transmitting the packet is similar to equation (5.9). The energy consumed by the radio module in the transmission of both attack and normal packets within an active period, assuming that there are no collisions are

$$E_{tx} = \begin{cases} N_t(\eta P_t + P_o)(T_{enc} + T_{tx}) & \text{for} \quad N_t(T_{enc} + T_{tx}) \geqslant \tau \ , \\ (\eta P_t + P_o)\tau & \text{for} \quad otherwise \end{cases} \tag{5.12}$$

Where η is the conversion factor of the power amplifier from electric power to RF power, P_o is the electronic power consumption overhead.

5.3.2 Analysis of MAC misbehaviour ghost energy depletion attack

Compromised IoT devices could be exploited to create and generate attack packets and to cause collisions in the shared channel. An increase in the number of collisions in the channel will lead to an increase in energy consumption of the IoT devices in the network. The mean total effective traffic intensity in the channel is

$$\lambda = N\lambda_0 \frac{1}{1 - P_c} + \lambda_A \tag{5.13}$$

where P_c is the probability that a device experiences a collision when it tries to transmit a packet and λ_0 is the mean arrival rate of packets at the output transmission queue of an IoT device. The first term is equation (5.13) is the effective normal traffic intensity created in the channel by IoT devices that are behaving normally and the last term λ_A is the additional mean traffic intensity created by compromised IoT devices (we also refer to it as the attack traffic intensify).

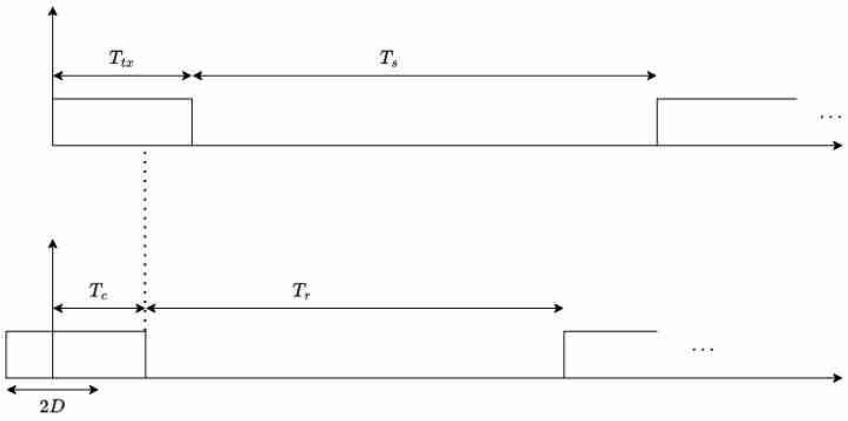

Figure 5.2: Channel access behavioral model of the IoT device.

All the IoT devices are listening to the channel to sense when the channel is free so that they can transmit their packets. Whenever the channel is free, an IoT device can transmit its packet normally within the timeframe of T_{tx}, as shown in figure 5.2. If a device transmits all its packets and its buffer is empty, it switches to a low-power mode until the next that it will wake up to either receive messages from the IoT gateway or take measurements and transmit them to the gateway. The more time a device spends in the low-power mode, the longer its lifetime. However, if a device detects a collision, it will try to access them again after a backoff time or retransmission time T_r, which is proportional to $\frac{1}{\lambda}$. The probability that there is a collision in the channel is

$$P_c = 1 - e^{-2\lambda D} \tag{5.14}$$

where D is the propagation delay of the wireless communication channel between the IoT device and the IoT gateway. By substituting P_c in equation (5.13) and simplifying we have

$$e^{2\lambda D} = \frac{\lambda - \lambda_A}{N\lambda_0} = \frac{\lambda}{N\lambda_0} - \frac{\lambda_A}{N\lambda_0} \tag{5.15}$$

shown in figure 5.3

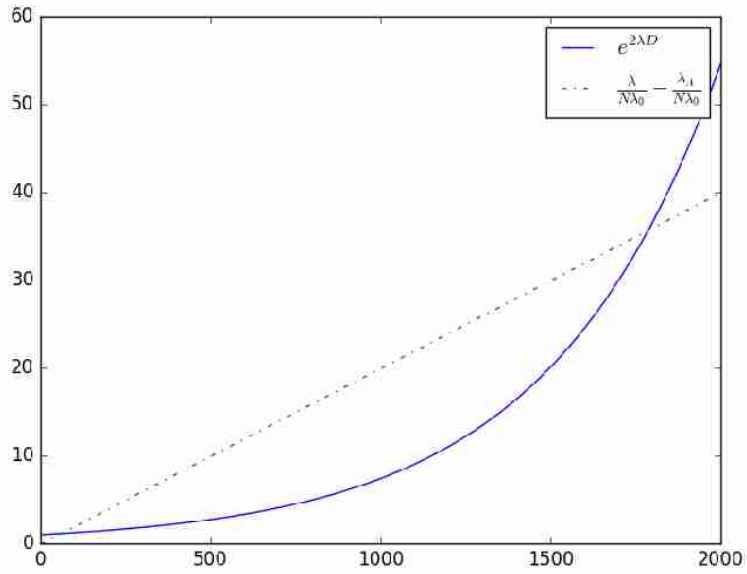

Figure 5.3: $\lambda_A = 5$, $\lambda_0 = 1$, $D = 0.001$, $N = 50$, $\lambda \in [0\ \ 2000]$

The mean number of transmission attempts including successful transmissions is

$$N_t = \sum_{n=1}^{\infty} nP_c^{n-1}(1 - P_c) = \frac{1}{1 - P_c} \tag{5.16}$$

and the mean number of collisions in the channel is

$$N_c = \frac{1}{1 - P_c} - 1 = \frac{P_c}{1 - P_c} \tag{5.17}$$

Therefore, the service time at the output queue of the IoT device is

$$\begin{aligned} \frac{1}{\mu} &= \frac{P_c}{1 - P_c}(T_c + T_r) + T_{tx} \\ &= [e^{2\lambda D} - 1](T_c + T_r) + T_{tx} \end{aligned} \tag{5.18}$$

The mean time required to transmit an IoT packet of size m is

$$T_{tx} = \frac{m}{C} \tag{5.19}$$

Where C is the Shanon capacity of the IoT wireless channel given by

$$C = Wlog_2(1 + SNR) \tag{5.20}$$

where W are the channel bandwidth, $SNR = \frac{P_r}{P_I + P_N}$ is the signal-to-noise ratio, and P_r, P_I, P_N are the received power, the interference power, and the noise power respectively. The power level of the signal transmitted by the IoT device is attenuated by the channel and the power that is received by the IoT wireless access point is

$$P_r = PL.P_t \tag{5.21}$$

where, P_t is the transmit power and the path loss PL is given by

$$PL = L_0 h d^{-a} \tag{5.22}$$

where d is the distance between the IoT device and the wireless access or gateway, h is a random variable that represents the fading in the channel, a is the path-loss exponent, and L_0 is a constant determined by the antenna gain, radio frequency, and the propagation environment. More advanced radio network planning models could be used to determine the path loss to account for possible environmental obstacles which cause radio signal degradation or fading. The retransmission delay is assumed to be exponentially distributed with parameter λ, and the mean retransmission delay T_r is, therefore, proportional to $\frac{1}{\lambda}$. Also, we assume that the collision time is uniformly distributed between 0 and D, and the mean collision time T_c is, therefore, $\frac{D}{2}$. Thus, equation (5.19) becomes

$$\frac{1}{\mu} = [e^{2\lambda D} - 1](\frac{1}{D} + \frac{1}{\lambda}) + \frac{m}{Wlog_2(1 + \frac{L_0 h d^{-a} P_t}{P_I + P_N})} \tag{5.23}$$

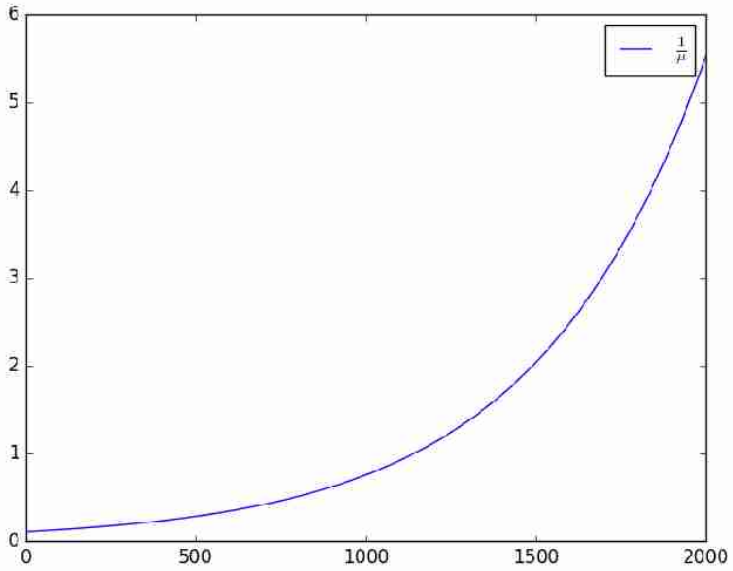

Figure 5.4: $D = 0.001$, $T_c = 0.001$, $T_r = 0.1$, $T_{tx} = 0.01$, $\lambda \in [0 \;\; 2000]$

From the Shanon information and communication theoretic limit to estimate the amount of energy required to transmit an IoT message is [117]

$$\epsilon = (\eta P_t + P_o) t_{tx} \tag{5.24}$$

which can also be expressed in terms of the channel and other transmission parameters as

$$\epsilon = \left(\frac{\eta}{PL} P_r + P_o\right) \frac{m}{Wlog_2(1 + SNR)} \tag{5.25}$$

For most IoT device majority of the communication is in the uplink (from the IoT device to the access point), especially for IoT devices whose function is to take measurements (sensing) and send them to the fog

or cloud servers. Since most of the communication is in the uplink, an attacker can easily conduct energy depletion attacks by causing the IoT device to increase its transmit power, increasing the time required to transmit a packet (by increasing packet sizes m or creating interference in the wireless channel) or by creating collisions in the shared wireless channel.

It should be noted that by deciding to generate or create packets with larger sizes, the ghost attacker increases the transmission times, thereby increasing the likelihood of collisions in the channels. Also, by increasing its traffic (by increasing its measurement frequency), the ghost attacker triggers an increase in λ which increase the likelihood of collisions in the channel and compels other devices sharing the channel with it to consume more energy. Hence, quickly draining their batteries.

Increasing the number of collisions in the shared wireless channel will increase the mean active time of the transmitter. The relationship between the mean active time of the transmitter and the mean total effective traffic intensity in the channel, λ is shown in figure 5.4. The mean total effective traffic intensity in the channel, λ can be increased by increasing the attack traffic. The main objective of the attacker is to maximise the active period, that is, to increase the time that the microcontroller unit and the radio unit spend performing computation and reception or transmission operations respectively. By maximising the active period of the MCU and radio unit, more energy is drained from the battery, and quickly depletes its energy content.

5.4 Modelling the energy depletion process for Battery of IoT devices

In this section, we present stochastic models that are useful for the analysis of the energy depletion process of the battery of an IoT device. We present Markovian model of the battery of an IoT device developed and used to analysed battery depletion attacks in [214, 215]. We propose a similar diffusion-based model of the battery of an IoT device. The diffusion model is also compared with the Markovian model.

5.4.1 Markovian model of the battery for IoT devices

Suppose that initially, the battery is fully charged to its full capacity B. Also, suppose that the energy stored in the battery is quantised, and that fixed-sized energy units are drawn from the battery to power the IoT devices. It is assumed that the energy consumption process is exponentially distributed. Let $N(t), t \geqslant 0$ be a random process that represents the number of energy units present in the battery at time t, and the probability that there are n energy units in the battery be $P\{N(t) = n\} = P_n(t)$, where, P_n is the state probability of having n energy units in the battery. The time evolution of the discharging process of the battery can be described by a pure death Markovian process as

$$\begin{aligned} \frac{dP_B(t)}{dt} &= -P_B(t)P_D \\ \frac{P_n(t)}{dt} &= -P_{n-1}P_D + P_{n+1}P_D \quad 0 < n < B \\ \frac{dP_0(t)}{dt} &= P_0(t)P_D \end{aligned} \tag{5.26}$$

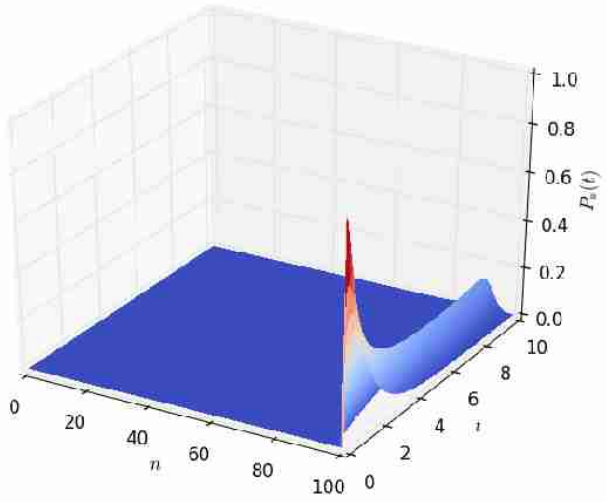

Figure 5.5: The distribution $P_n(t)$, of the amount of energy n in Wh present in the battery at time t during the blackout period, for $t \in (0 \quad 10]$, $n \in [0 \quad 10]$, $B = 100$, and $P_D = 1$.

The solution of the set of equations in (5.27) gives the state probability of the amount of energy present in the battery at time t, and it is given by [214, 215]

$$P_n(t) = \frac{(P_D t)^{B-n}}{(B-n)!} e^{-P_D t}, \quad 0 < n \geqslant B \tag{5.27}$$

Using the normalisation, $\sum_{n=1}^{B} P_n(t) + P_0(t) = 1$, the probability that the battery is empty at time t (service outage probability) during the backout period is

$$P_0(t) = 1 - \sum_{n=1}^{B} \frac{(P_D t)^{B-n}}{(B-n)!} e^{-P_D t} \tag{5.28}$$

Figures 5.8 and 5.5 shows the density of the number of energy units n present in the battery at time t. It starts with a sharp spike and then gradually decreases to zero. It is because, initially (at time $t = 0$), we start with a fully charged battery to its capacity B. The battery's energy content gradually decreases with time and eventually reaches zero when all the energy stored in the battery is completely depleted. A similar probability density can be obtained by modelling the battery using a diffusion process, as shown in this study.

The time required to completely deplete the energy stored in the battery is the first passage time of the Markov process from the state $n = B$ at time $t = 0$ to the state $n = 0$ at time t; when the energy stored in the battery is completely depleted. It is the lifetime of the IoT device. Let the random variable T represent the time required to completely deplete the energy stored in the battery, where the first passage time process is $T = inf\{t > 0 : N(t) = 0\}$. The distribution of the first passage time from a defined point $n = B$ to $n = 0$ is

$$\gamma_{B,0}(t) = \left[P_D e^{-P_D t}\right]^{*(B-1)} \tag{5.29}$$

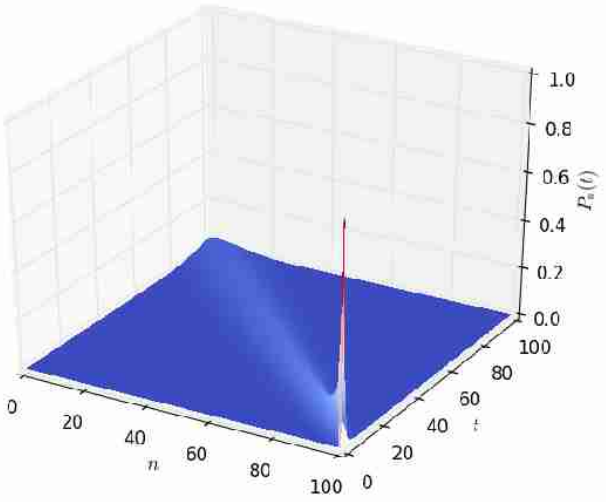

Figure 5.6: The distribution $P_n(t)$, of the amount of energy n in Wh present in the battery at time t during the blackout period, for $t \in (0 \quad 100]$, $n \in [0 \quad 100]$, $B = 100$, and $P_D = 1$.

where $*(B-1)$ is (B-1)-fold concolution of a function with itself. The mean of the distribution (5.29) gives the mean of the time required completely deplates the amount of energy present in the battery and is given by

$$E[T] = \frac{B}{P_D} = \frac{B}{R_{ACT}.P_{ACT} + R_{SLEEP}.P_{SLEEP}} \tag{5.30}$$

Despite the assumption that the energy consumption process is exponentially distributed, the mean lifetime of the IoT device given in 5.30 is the same as the well-known expression for the lifetime of an IoT device given in equation 5.2, for $SF_{bat} = 1$. The same expression can be derived using a diffusion approximation-based model. The diffusion approximation modeling approach removes the assumption that the energy consumption process should be exponentially distributed. It is also unnecessary to discretise the battery's energy as the diffusion process is a continuous stochastic process.

5.4.2 Diffusion approximation model of the battery for IoT devices

Suppose that initially, the battery is fully charged to its full capacity B. Suppose that the cumulative amount of energy drawn from the battery to power the IoT device up to time t is $E_D(t)$, then the amount of energy present in the battery at time t is

$$E(t) = B - E_D(t) \tag{5.31}$$

The change in the amount of energy in the battery between time t and $t + \Delta$ is

$$E(t+\Delta) - E(t) = \{E_D(t+\Delta) - E_D(t)\} \tag{5.32}$$

If we assume that as energy is drawn from the battery, the changes in the energy content of the battery $\Delta E(t) = E(t+\Delta) - E(t)$ are normally distributed, then we can approximate the discharging process of the battery by a diffusion or Brownian motion process $X(t)$ whose changes $\Delta X(t) = X(t+\Delta) - X(t)$ are normally distributed with mean $\beta\Delta t$ and variance $\alpha\Delta t$ where:

$$\begin{aligned}\beta &= \lim_{\Delta t\to 0}\frac{E[X(t+\Delta t)-X(t)]}{\Delta t}\\ \alpha &= \lim_{\Delta t\to 0}\frac{Var[X(t+\Delta t)-X(t)]}{\Delta t}.\end{aligned}$$

We represent the discharging process of the battery by a diffusion process that starts at $X(0) = B$ (battery is fully charged at the beginning) and ends at $X(t) = 0$ (when the battery is fully discharged). The dynamic changes in the energy content of the battery (during the discharging process) can then be described by the second order partial differential equation of a diffusion process, e.g. [44],

$$\frac{\partial\psi(x,t;B)}{\partial t} = \frac{\alpha}{2}\frac{\partial^2\psi(x,t;B)}{\partial x^2} - \beta\frac{\partial\psi(x,t;B)}{\partial x}. \tag{5.33}$$

subject to the conditions

$$\begin{aligned}\psi(B,0;B) &= \delta(B) \\ \psi(0,t;B) &= 0 \quad for \quad t>0\end{aligned} \tag{5.34}$$

where $\psi(x,t,B)$ is the Probability Density Function (PDF) that we have x amount of energy in the battery at time t, given that the discharging process started with $x = B$ amount of energy in the battery. The boundary condition, $\psi(x,0;B) = \delta(B)$ in (5.35) is the initial condition that the battery is fully charged at the beginning, that is, with a probability of 1, the amount of energy present in the battery at time $t = 0$ is B (the probability density in infinite for $t = 0$ and $x = B$). The boundary condition $\psi(0,t;B) = 0 \quad for \quad t>0$ is the final condition that at time time t, the energy stored in the battery is completely depleted. The solution of (5.33) with the conditions in (5.35) gives the PDF of the diffusion process that starts at $x = B$ and ends at $x = 0$ is

$$\psi(x,t;B) = \frac{e^{\frac{\beta}{\alpha}(x-B)-\frac{\beta^2}{2\alpha}t}}{\sqrt{2\pi\alpha t}}\left[e^{-\frac{(x-B)^2}{2\alpha t}} - e^{-\frac{(x+B)^2}{2\alpha t}}\right]. \tag{5.35}$$

The PDF given in equation (5.35) satisfies the partial difference equation in (5.33) and the conditions in (5.35, that is,

$$\begin{aligned}\psi(B,0;B) &= \lim_{t\to 0}\psi(B,t;B)\\ &= \delta(B)\end{aligned}$$

and

$$\begin{aligned}\psi(x,t;B) &= \lim_{t\to\infty}\psi(x,t;B)\\ &= 0 \quad for \quad t>0 \quad and \quad x\geqslant 0\end{aligned}$$

, which implies that the distribution $\psi(x,t;B)$ converges to 0 within a finite time horizon $t > 0$, for any value of $x \geqslant 0$.

The lifetime of the IoT device is the time required to completely deplete the energy stored in the battery and is the first passage time from $x = B$ (when the battery is fully charged) to $x = 0$ (when the energy stored in the battery is completely depleted). Let the random variable T represent the lifetime of the IoT device, where the process is represented by the random process $T = inf\{t > 0 : X(t) = 0\}$ is the first passage time of the diffusion process from $x = B$ to $x = 0$. The PDF of the first passage time of the diffusion process from $x = B$ to $x = 0$ is

$$\begin{aligned} \gamma_{B,0}(t) &= \lim_{x\to 0}[\frac{\alpha}{2}\frac{\partial\psi(x,t;B)}{\partial x} - \beta\psi(x,t;B] \\ &= \frac{B}{\sqrt{2\pi\alpha t^3}}e^{-\frac{(B+\beta t)^2}{2\alpha t}} \end{aligned} \tag{5.36}$$

The mean lifetime of the IoT device (or the mean time to failure) is

$$\begin{aligned} \mu_T &= \int_0^\infty t\gamma_{B,0}(t)dt \\ &= \frac{B}{\beta} \end{aligned} \tag{5.37}$$

and the variance of the lifetime of the IoT device is

$$\begin{aligned} \sigma_T^2 &= \int_0^\infty t^2\gamma_{B,0}(t)dt - \mu_T^2 \\ &= -\frac{B\alpha}{\beta^3} \end{aligned} \tag{5.38}$$

If we consider a battery in which energy is stored in it at a mean rate of P_S and energy is drawn from it at a mean rate P_D, then the energy in the battery changes at a mean rate $\beta = P_S - P_D$ [243]. The variance of these changes is $\alpha = C_A^2 P_S + C_B^2 P_D$, where C_A^2 and C_B^2 are the coefficients of variation of the process supplying energy to the battery and the process of drawing energy from the battery, respectively. However, in this paper, we assumed that energy is not supplied to the battery of the IoT device; that is, we assume that the device depends only on the energy stored in the battery during the deployment of the IoT device (i.e., $P_S = 0$ and $C_A^2 = 0$). Therefore, the parameters of the diffusion process can be defined as $\beta = -(R_{ACT}.P_{ACT} + R_{SLEEP}.P_{SLEEP})$ and $\alpha = C_B^2(R_{ACT}.P_{ACT} + R_{SLEEP}.P_{SLEEP})$ respectively. The diffusion parameter $\beta = -(R_{ACT}.P_{ACT} + R_{SLEEP}.P_{SLEEP})$ and $\alpha = C_B^2(R_{ACT}.P_{ACT} + R_{SLEEP}.P_{SLEEP})$. Therefore, the expected lifetime of the IoT device or the Mean-Time-To-Failure (MTTF) becomes:

$$\mu_T = \frac{B}{R_{ACT}.P_{ACT} + R_{SLEEP}.P_{SLEEP}} \tag{5.39}$$

which is the well-known results for the expected lifetime of an IoT device. Also, the variance of the lifetime of the IoT device becomes:

$$\sigma_T^2 = \frac{B\alpha}{[R_{ACT}.P_{ACT} + R_{SLEEP}.P_{SLEEP}]^3} \tag{5.40}$$

Figure 5.4.2 shows the comparison of the probability density of the lifetime of the IoT device $\gamma_{B,0}(t)$ for $B = 100$ Wh and $P_D = 0.2W$. For the diffusion model, we use $C_B^2 = 1$ to ensure that the energy consumption process is exponentially distributed to compare the Markovian and the diffusion approximation models. It can be observed that the probability density function of the lifetime of the IoT device obtained using the proposed diffusion model is the same as that obtained using the pure death Markovian model used in [214, 215] to model the battery of an IoT device.

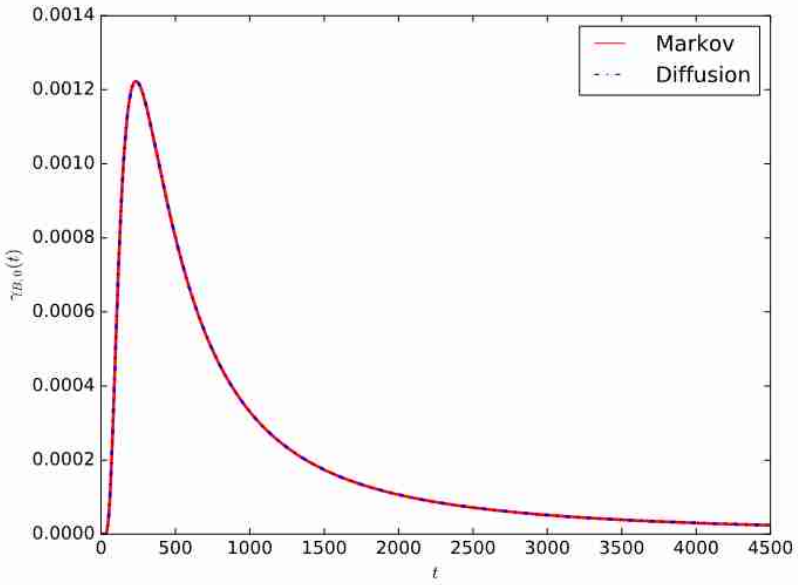

Figure 5.7: Comparing the probability density of the lifetime of the IoT device $\gamma_{B,0}(t)$ for $B = 100$ Wh and $P_D = 0.2W$.

The probability that the energy stored in the battery is completely depleted after time ξ is

$$\begin{aligned} \Gamma_{B,0}(t) &= \int_0^t \gamma_{B,0}(\xi)d\xi \\ &= \frac{1}{2}\left(e^{-\frac{2B\beta}{\alpha}} erfc\left[\frac{B-\beta t}{\sqrt{2\alpha t}}\right] + erfc\left[\frac{B+\beta t}{\sqrt{2\alpha t}}\right]\right) \end{aligned} \tag{5.41}$$

where

$$\text{erfc}(t) = 1 - \text{erf}(t), \quad \text{and} \quad \text{erf}(t) = \frac{2}{\sqrt{\Pi}}\int_0^t e^{-\xi^2} d\xi.$$

Let us suppose that during the deployment of the IoT device, it is desired to predict the time after which the energy level of the battery should have decreased to a predefined threshold so that the operator could consider changing the battery or charging it. Suppose that this threshold is $x = \eta B$, where $\eta \in (0 \quad 1)$, then the time t, after which the energy level of the battery decreased to the defined threshold is the first passage time from $x = B$ at time $t = 0$ to $x = \eta B$ at time t, and it's distribution is

$$\gamma_{B,\eta B}(t) = \frac{B(1-\eta)}{\sqrt{2\pi\alpha t^3}} e^{-\frac{(B(1-\eta)+\beta t)^2}{2\alpha t}} \tag{5.42}$$

with mean

$$\begin{aligned} \mu_t &= \int_0^\infty t\gamma_{B,\eta B}(t)dt \\ &= \frac{B(1-\eta)}{\beta} \end{aligned} \tag{5.43}$$

and variance

$$\begin{aligned} \sigma_T^2 &= \int_0^\infty t^2 \gamma_{B,\eta B}(t)dt - \mu_t^2 \\ &= -\frac{B(1-\eta)\alpha}{\beta^3} \end{aligned} \tag{5.44}$$

The probability that after time t, the energy level of the battery should have decreased to reach the defined threshold is

$$\begin{aligned} \Gamma_{B,\eta B}(t) &= \int_0^t \gamma_{B,\eta B}(\xi)d\xi \\ &= \frac{1}{2}\left(e^{-\frac{2B(1-\eta)\beta}{\alpha}} erfc\left[\frac{B(1-\eta)-\beta t}{\sqrt{2\alpha t}}\right] + erfc\left[\frac{B(1-\eta)+\beta t}{\sqrt{2\alpha t}}\right]\right) \end{aligned} \tag{5.45}$$

5.5 Numerical examples

This section presents some numerical examples to study the dynamics of the energy depletion process in a small-sized battery used to power IoT devices. We present the influence of design parameters that can be selected during the design and deployment of the IoT devices on the time it takes to drain all the energy stored in the battery completely. This time gives the lifetime of the IoT device from the time the device is deployed to when the energy stored in its battery is completely drained and the device is shut down.

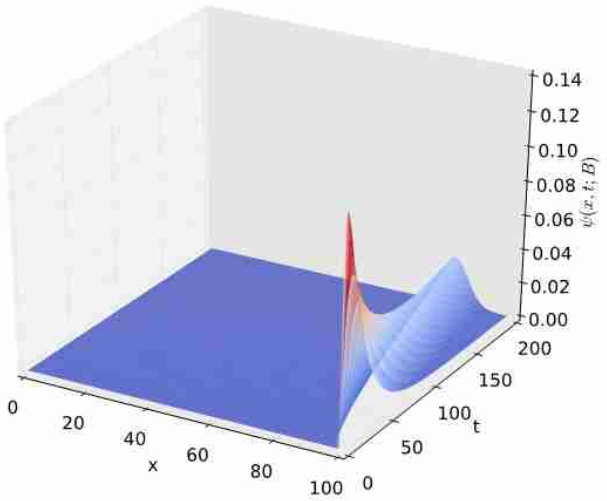

Figure 5.8: The distribution $\psi(x,t;B)$, of the amount of energy x in Wh present in the battery at time t, for $t \in (0 \quad 200]$ and $x \in [0 \quad 100]$.

Fig. 5.8 shows the probability density of having x amounts of energy in the battery at time t given that the discharging process started with $x = B$ amounts of energy in the battery. After a certain time t, the distribution of the amount of energy present in the decreases to $x = 0$, which happens when the energy

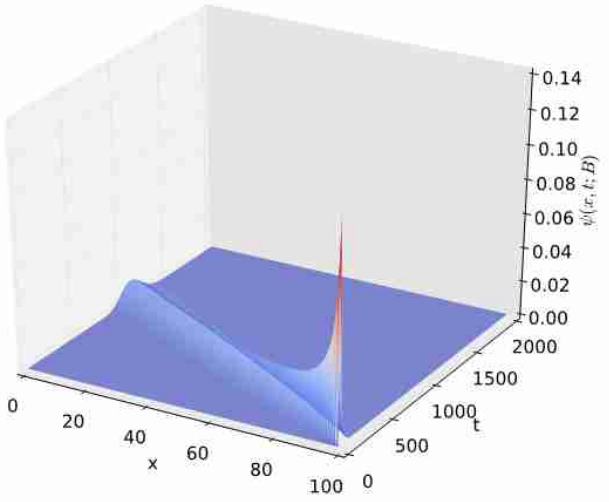

Figure 5.9: The distribution $\psi(x, t; B)$, of the amount of energy x in Wh present in the battery at time t, for $t \in (0 \quad 2000]$ and $x \in [0 \quad 100]$.

stored in the battery is completely depleted. The time axis gives the distribution of time required for the amount of energy present in the battery to decrease from $x = B$ to a give amount x. To clearly observe how the distribution of the amount of in battery varies over time, we increased the time axis as shown in Fig. 5.5. For the results presented in Figures 5.8 and 5.5, $P_A = 0.12471$ W, $P_S = 0.090$ W, and $R_{SLEEP} = 0.95$, and $R_{ACT} = 0.05$, $CB^2 = 1$, and $B = 100$.

Figure 5.5 shows the influence of the average power consumed by the device when it is in the active mode, P_{ACT} on the distribution of the lifetime of the device. It can be observed that a very small increase in the average power consumed by an IoT device in active mode can significantly reduce the lifetime of the IoT device. In battery depletion attacks, an attacker could cause the IoT device to increase its transmission power, increasing the power consumed by the IoT device in the active mode. There are various ways in which the power consumed by the device when it is in the active mode can be increased. For the Plot in Fig. 5.5, $B = 100$, $P_S = 0.090$ W, and $R_{SLEEP} = 0.95$, and $R_{ACT} = 0.05$, $CB^2 = 1$. Figure 5.5 shows the influence of the proportion of sleep time, R_{SLEEP} on the distribution of the lifetime of an IoT device. The distribution is shifted to the right as R_{SLEEP} increases. It is because increasing the sleep time reduces the energy consumed by the IoT device and hence, increases its lifetime. The most popular energy depletion attacks designed to completely drain the energy of the battery of the IoT device are denial of sleep and various types of vampire attacks. They are conducted by manipulating some device or network parameters to reduce the sleep time (and hence, R_{SLEEP}) of the device. Fig. 5.5 shows the influence of the proportion of sleep time, R_{SLEEP} on the probability that the energy stored in the battery is completely depleted before a given time t. It shows that as the proportion of sleep time increases, the higher the probability that the energy stored in the battery will be completely depleted before a defined time t. Figure 5.5 influence of the

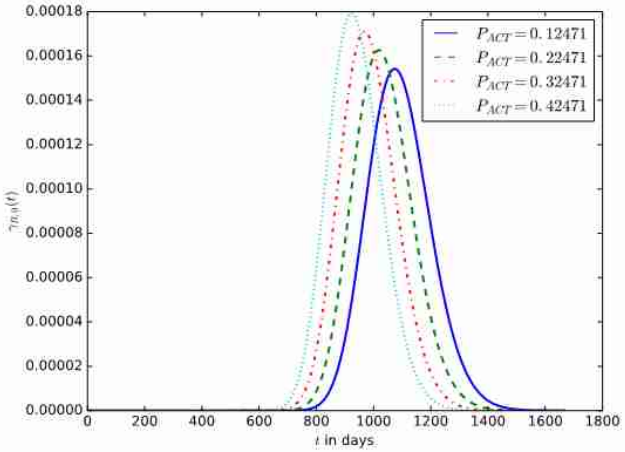

Figure 5.10: The influence of the active mode power, P_{ACT} on the distribution of the lifetime of the IoT device.

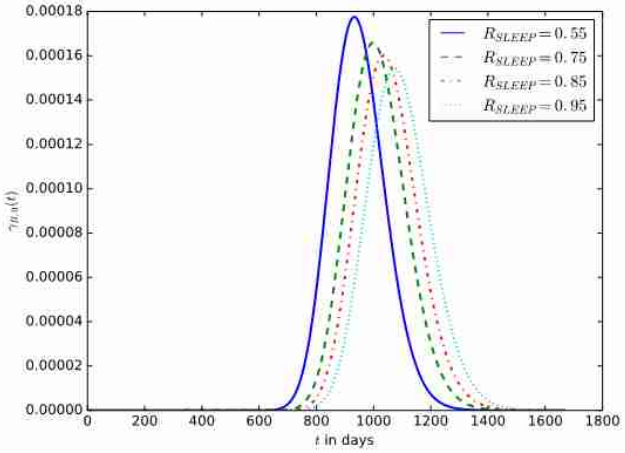

Figure 5.11: The influence of the proportion of sleep time, R_{SLEEP} on the distribution of the lifetime of the IoT device.

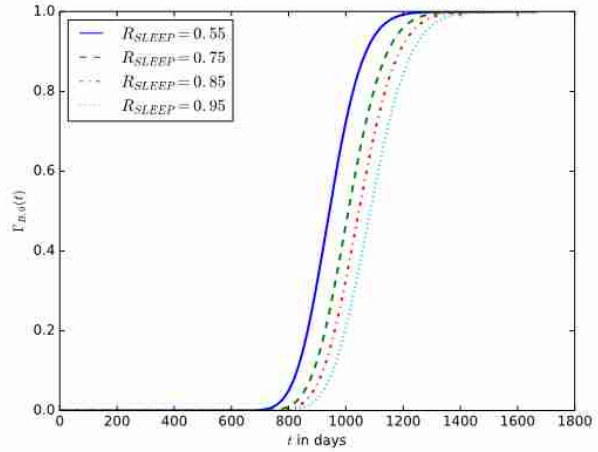

Figure 5.12: The influence of the proportion of sleep time, R_{SLEEP} on the probability that the energy stored in the battery is completely depleted.

Coefficient of variance of the energy consumption C_B^2 on the distribution of the lifetime of the IoT device. It shows that large variations in the energy consumption of the IoT also result in a large variation in the lifetime of the IoT devices and makes it difficult to predict the lifetime of the IoT device or the expected time to change the battery of the device. Usually, the energy consumption of the IoT devices may vary very slightly, but the variations could be significant if the device experiences energy depletion attacks at random times. Some energy depletion attacks that do not degrade the QoS (some improve QoS by increasing the transmit power when the noise or interference level is high while draining more energy from the battery) make them difficult to detect using the traditional QoS-based attack detection mechanisms. In this case, it could be preferable to monitor both the QoS and the energy consumption metrics and use them for attack detection. Figure 5.5 shows the influence of the battery capacity on the probability that before time t, the energy stored in the battery is completely depleted. The IoT devices that require a long life include those used in industries such as oil and gas, agriculture, health care, wildlife conservation, forestry, and water monitoring [186]. The use of batteries with small energy storage capacity results in frequent battery replacements, which increase maintenance costs. However, the choice of battery capacity for an IoT device depends on the battery's cost, size, weight, and energy density. Therefore, based on the power consumption budget of the IoT device, the battery specifications should be selected in such a way as to have a long lifetime. Figure 5.5 shows the relationship between the battery capacity and the mean lifetime of the device.

When deploying an IoT device in an IoT network, it is possible that the probability of the time after the energy stored in the battery should have decreased to a defined percentage ($\eta\%$) of its initial amount. It enables the operator to estimate the time after which the battery of the IoT device should be replaced without waiting until the energy stored in the battery is completely depleted before the battery is replaced. It is to ensure that the device is not shut down due to the complete depletion of the energy stored in the battery. Figure 5.5 shows the influence of the energy threshold percentage $x = \eta$, on the probability density function (PDF), $\gamma_{B,\eta B}(t)$. It is the PDF that after time t, the amount of energy present in the battery is $x = \eta B$, that

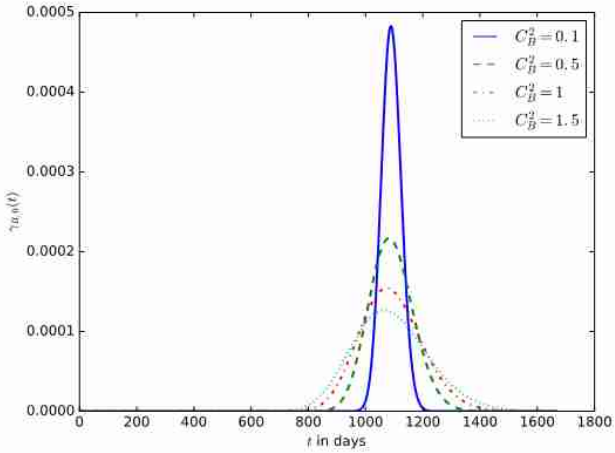

Figure 5.13: The influence of the Coefficient of varience of the energy consumption C_B^2 on the distribution of the lifetime of the IoT device.

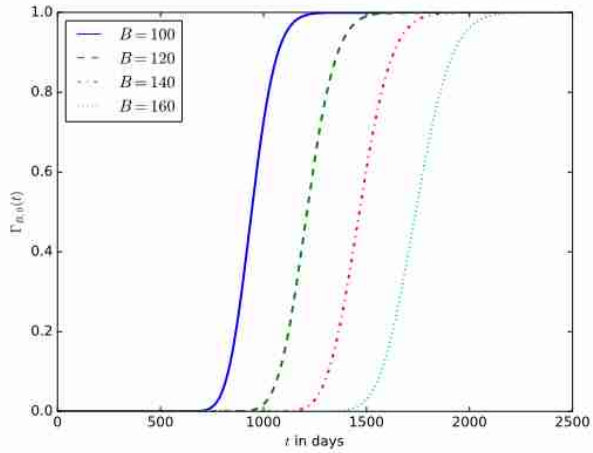

Figure 5.14: The influence of the battery capacity B on the probability that the energy stored in the battery is completely depleted.

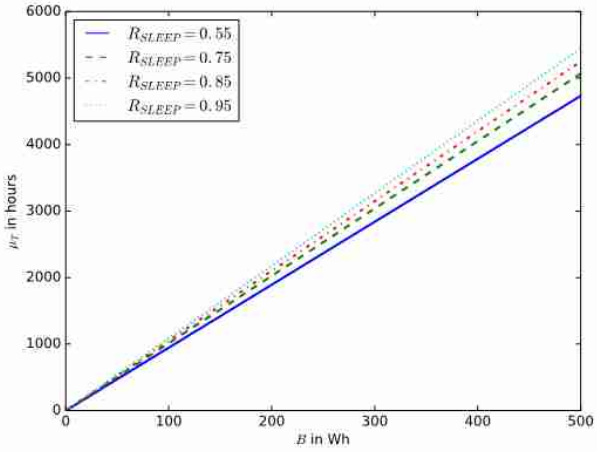

Figure 5.15: Mean device lifetime μ_T versus battery capacity B for various values of sleep mode ratio R_{SLEEP}.

is, the battery must have discharged to $1 - \eta$ percent of its initial amount of energy. Figure 5.5 shows the influence of the energy threshold percentage $x = \eta$, probability $\Gamma_{B,\eta B}(t)$ that after time t, the amount of energy present in the battery is $x = \eta B$; that is, the battery must have discharged to $1 - \eta$ percent of it's initial amount of energy.

5.6 Conclusion

During the development and deployment of IoT devices and networks, making reasonable tradeoffs between QoS, security, and the energy consumption is essential to ensure reliability, security, and a longer lifetime of the IoT devices. It is important to prioritize power consumption when designing and deploying IoT devices and networks. Some of the ways to reduce power consumption include the implementation of sleep mode (which could reduce up to 90% of the energy consumption), avoiding excessive push notifications, choosing when and how to transmit information, selecting the most appropriate wireless protocol [186], and implementation of energy depletion attack systems to prevent attacks that are aimed at rapidly draining the battery of the device. To prolong the lifetime of IoT devices and to minimise the impact of battery depletion, energy harvesting has been used where possible to recharge the battery with energy harvested from the environment. Our study was limited to developing the diffusion approximation for a battery without any renewable energy source.

We have applied a diffusion or Brownian motion process to model the energy depletion process of a battery of an IoT device. We used the model to obtain the probability density function, mean, variance, and probability of the lifetime of an IoT device. Also, we studied the influence of the active power consumption, sleep time, battery capacity on the probability density function, mean, and probability of the lifetime of an IoT device. Since battery depletion attacks are always aimed at manipulating the IoT device to increase its energy consumption significantly, the numerical examples enabled us to study the influence of battery

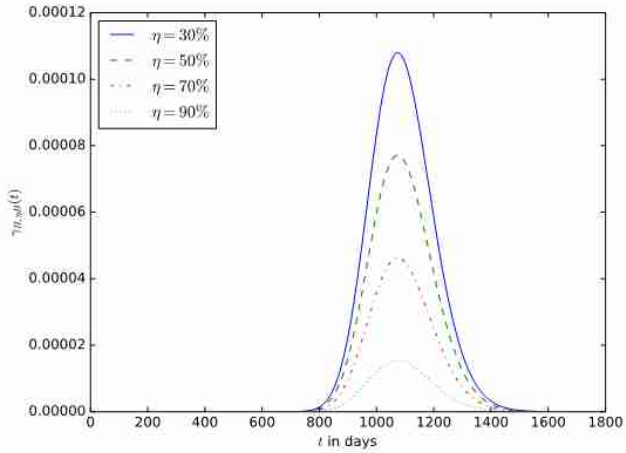

Figure 5.16: The probability density function (PDF), $\gamma_{B,\eta B}(t)$ that after time ξ, the amount of energy present in the battery is $x = \eta B$, that is, the battery must have discharged to $1 - \eta$ percent of it's initial amount of energy.

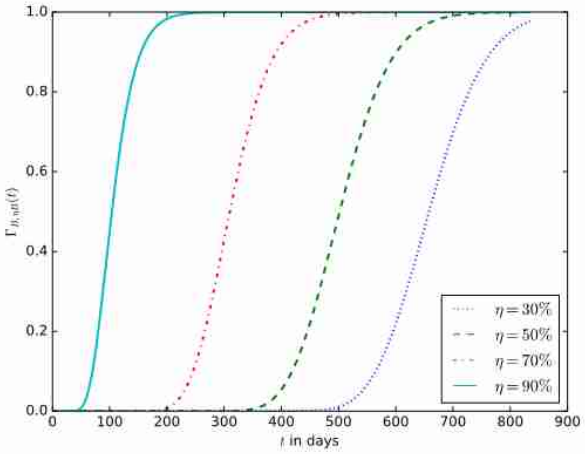

Figure 5.17: The probability $\Gamma_{B,\eta B}(t)$ that after time ξ, the amount of energy present in the battery is $x = \eta B$, that is the battery must have discharged to $1 - \eta$ percent of it's initial amount of energy.

depletion attacks on the distribution of the lifetime of an IoT device. We also introduced in our model the energy threshold after which the battery of the device should be replaced to ensure that the battery is not completely drained before it is replaced. The time after which the battery should be replaced can be obtained from our model. Therefore, the diffusion approximation can be used to conveniently model the energy depletion process of the battery of an IoT device, and with knowledge of the battery capacity, the average power consumption, and the variance of energy consumption (if any), the probability density function, the mean, variance, and probability of the lifetime of an IoT device can be obtained.

Chapter 6

Performance Modelling of the Battery Energy Storage System (BESS) for a Green Mobile Network Base Station (e.g BTS,NodeB,eNodeB or gNodeB) Site

In the previous chapter, we discussed modelling the energy depletion process used to supply computer systems (e.g., IoT devices, drones, mobile phones, etc.). Some of these battery-powered computer systems can be deployed alongside energy harvesting systems. The energy harvesting systems can harvest energy from environmental energy sources such as solar, thermal, wind, and vibrations. The harvested energy can be used to supply the computer systems or stored in a Battery Energy Storage System (BESS). The energy stored in the BESS can then be drawn to power the computer systems.

The use of energy harvesting systems to replenish the energy depleted from the battery of battery-powered computer systems to operate for a long time without human intervention to recharge or replace the battery. In battery-powered computer systems (e.g., IoT devices, drones, etc.), their lifetime (the required to drain the energy stored in the battery) is very important. Deploying energy harvesting systems to continuously recharge the battery significantly extends the lifetime of the computer system, hence improving the QoS.

Also, using renewable energy sources to power computer systems and Information and Communication Technology (ICT) infrastructures is a sustainable way to reduce ICT-related carbon emissions. In off-grid rural environments, renewable energy may be a cheaper solution to power computer systems or ICT infrastructures. Electricity supply in sub-Saharan Africa is subject to frequent outages due to insufficient energy generation and/or poor transmission distribution infrastructure [77]. Also, many rural communities in these regions are not yet connected to the grid. Thus, ICT infrastructures (e.g., base stations, network access points, transmission systems) deployed in off-grid environments can be reliably powered using energy harvesting and storage infrastructures.

There is increasing use of renewable energy to power base station sites to reduce carbon footprint and operational expenditures (OPEX) [248]. With the recent increase in energy prices, the cost of energy has become the dominating operational cost for mobile network operators [197]. Although the base stations of

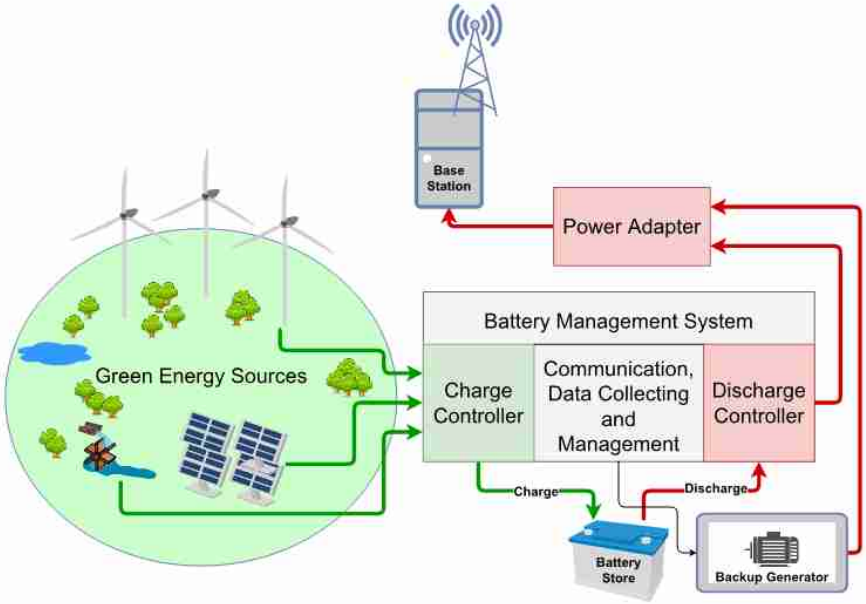

Figure 6.1: The architecture of a green base station site.

next-generation mobile networks (5G/6G mobile networks) are designed to be energy efficient, the dense and large-scale deployment of these base stations will increase the energy demands of mobile networks. Therefore, increasing the energy mix of mobile networks to include renewable energy sources and energy storage systems could reduce the carbon emission and operation of mobile networks.

Because of the transient fluctuations in the supply of energy from renewable energy sources [201] and the fluctuation in the energy demand of some computer systems and ICT infrastructure, BESS is essential to absorb the energy harvesting intermittency and demand fluctuations. The harvested energy is temporarily buffered in the BESS and then drawn to power computer systems or ICT infrastructure. In this chapter, we apply Markovian and diffusion approximation models to analyse the performance of BESS for a green base station site. We compared the results obtained from the two modelling approaches.

In this chapter, we present an architecture of a green base station site. We develop Markovian and diffusion approximation models for the analysis of steady-state and transient-state performance of battery energy storage systems. We apply Markovian and diffusion approximation model to derive the time after which the battery energy storage system is completely discharged or fully charged. By assuming that the energy harvesting and the energy consumption processes are exponentially distributed, we compare the result obtained from the Markovian model to those from diffusion approximation models. A portion of the material presented in this chapter was published in [58, 53].

6.1 The architecture of battery energy storage systems for a base station

The architecture of a green base station site consists of renewable energy sources (e.g., wind or solar), an energy storage system (e.g., battery bank), a battery management system, and the base station system with other hardware infrastructure. Renewable energy sources generate energy that is stored in the battery

energy storage system. The stored energy is drawn and transferred to the base station power system and used to power the base station.

Consider a base station that is located in a remote area without access to an electricity grid, as shown in figure 6. Renewable energy sources such as photovoltaic or wind are installed to generate enough electrical energy to power the base station, the transmission systems, and all the other electrical infrastructure installed on the base station site. Since the energy generated by renewable energy sources may fluctuate depending on the weather conditions, a battery energy storage system is installed between the renewable energy sources and the base station to store the excess energy generated and smoothen the output power delivered to the base station.

The battery management system performs supervisory control and data acquisition. The data about the state of charge and state of health of the BESS, the average energy generation rate, and the average energy consumption rate is collected. The data collected can be used for local supervisory and control and sent to the operation, maintenance, administration (OMA) centre of the mobile operator for further processing and optimization. If the energy stored in the BESS is depleted below a defined energy threshold, the battery management system should switch the base station from the battery to the backup generator to ensure that the base is not shut down when the battery is fully drained. A shut down of a base station results in significant losses for the mobile operator.

Suppose the weather condition or the deployed renewable energy resources is able to generate sufficient amounts of energy that can supply the base station with little or no fluctuations. In that case, the supervisory control should be configured so that the energy infrastructure can supply the base station directly and charge the BESS simultaneously. In this case, when the weather condition is not sufficient for generating enough energy to supply the base station directly, the base station is powered through the BESS. When the energy stored in the BESS decreases below a defined threshold, and the renewable energy sources are not sufficient to power the base station, it is switched from the BESS to a backup generator to avoid service interruption.

6.2 Stochastic model for a battery energy storage systems

The amount of energy generated by the PV systems or Wind turbine is non-deterministic because it depends on variable environmental factors such as intensity of the sunlight, duration of the sunlight, temperature, humidity, and win speed. The energy drawn from the battery to power the base station is non-deterministic because it depends on the service traffic which is random [167]. Therefore, since the energy generation and energy consumption processes are non-deterministic, the dynamic changes in the amount of energy present in the battery at time t can be modelled using stochastic processes e.g., Markovian models and diffusion approximation models.

The metric that is used to represent the available storage capacity of a battery energy storage system is the state-of-charge (SoC)[256]. It provides information about the relative amount of energy remaining in the batter. It can be estimated using the Coulomb counting (Ampere-hour integral) as [237]

$$SoC(t) = \left[1 + \frac{\eta_e \int_0^t I(\xi)d\xi}{Q}\right] * 100 \tag{6.1}$$

where η_e is the battery's coulombic efficiency, I is the battery terminal current, and Q is the rated capacity of the battery (in Ah), and assuming that the battery is full initially at time $t = 0$. The use of state-of-charge

metric is sometimes preferable because it is can be expressed in terms of the battery voltage which is influnced by by environmental factors and battery parameters such as discharging rate, ambebient temperature, and battery aging. It provides information about the residual energy of the battery and is defined as the ratio of the remaining energy to the total energy [252]. It can be expressed as

$$\begin{aligned} SoE(t) &= \left[1 + \frac{\eta_e \int_0^t I(\xi)V(\xi)d\xi}{B}\right] * 100 \\ &= \left[1 + \frac{\eta_e \int_0^t P(\xi)d\xi}{B}\right] * 100 \\ &= \left[1 + \frac{\eta_e E(t)}{B}\right] * 100 \end{aligned} \tag{6.2}$$

Where V is the battery voltage, P is the power, B is the energy rating of the battery (in Wh), and The energy present in the battery at time t. The changes in SoC or SoE indicates changes in the energy content of the battery e.g.,

$$\frac{B}{100 * \eta_e}\left[SoE(t+\Delta) - SoE(t)\right] = E(t+\Delta) - E(t) \tag{6.3}$$

The amount of energy present in the battery at time t is

$$E(t) = E_H(t) - E_D(t) \tag{6.4}$$

where $E_H(t)$ is the cummulative amount of energy harvested and stored in the battery and $E_D(t)$ up to time t is the cummulative amount of energy drawn from the battery to supply the load up to time t. The change in the amount of energy in the queue between time t and $t + \Delta$ is

$$E(t+\Delta) - E(t) = \{E_H(t+\Delta) - E_H(t)\} - \{E_D(t+\Delta) - E_D(t)\} \tag{6.5}$$

For small energy changes ($\Delta - > 0$), the energy present in the battery at any given time t satisfies the following differential equation:

$$\frac{dE(t)}{dt} = \frac{dE_H(t)}{dt} + \frac{dE_D(t)}{dt} \tag{6.6}$$

The differential equation in 6.6 is difficult to solve because the function $E(t)$ is not differentiable at some points [233]. Also, it is difficult to express $E_H(t)$ and $E_D(t)$ as continuous functions, instead what is often available is the data of the power generated by the photovoltaic systems and the data of the energy consumption of the base station. Since the rate at which the energy is generated and stored into the battery and the rate at which energy is drawn from the battery to supply the base station are non-deterministic, then, the SoC, SoE, or the amount of energy present in the battery at any given time is also non-deterministic, and can be modelled as a stochastic process.

6.3 Markovian model of the BESS for a base station site

To apply Markovian models to model the battery energy storage system of a base station site of a mobile network, we suppose that the energy generated by the photovoltaic system is stored into the battery energy storage system in discrete amounts called energy units or energy packets. Also, we discretise the energy drawn from the battery to power the base station into energy units. Since the amount of energy generated and the amount of energy drawn from the battery varies randomly over time, we assume that the arrival

process of energy packets into the battery follows a Poison process with mean rate $\hat{P_H}$ (in W) and that the discretised energy consumption process is exponentially distributed with mean rate $\hat{P_D}$ (in W).

The discretization and these assumptions are used for mathematical convenience to treat the battery energy storage system as an energy buffer that is continuously supplied with random amounts of energy and random amounts of energy is drawn from it to power the basse station. It enables the application of well-known Markovian models developed for the analysis of queueing systems in computer systems and networks, manufacturing, and the service industry. When the battery is charged to full capacity, additional amounts of energy generated is not added to the battery (it is either redirected to another energy storage system or wasted). Therefore, we model the battery as an M/M/1/B Markovian model, which is simple and its results are well-known in the queueing theory literature. The symbol M represents the the energy generation and consumptions processes, B is the capacity of the battery, 1 indicates that we have a single load (base station) that is supplied by the batery.

A Markovian process is a stochastic process that possess the "memoryless" property in which the future of the process depends only only on its present state (regardless of its past states); that is, the knowledge about the future states of the process, depends entirely on the present states. Let $N(t), t \geqslant 0$ be a random process that represents the number of energy units present in the batter at time t, and the probability that there are n energy units in the battery be $P\{N(t) = n\} = P_n(t)$, where, P_n is the state probability of having n energy units in the battery. If the process is at state n (that is n energy units in the battery), and with the arrival of an energy unit the process jumps to the state $n + 1$. The difference-differential equations for the time-dependent evolution of the process is

$$\begin{aligned} \frac{dP_0(t)}{dt} &= -P_0(t)\hat{P_H} + P_1(t)\hat{P_D} \\ \frac{P_n(t)}{dt} &= P_{n-1}(t)\hat{P_H} - P_n(t)(\hat{P_H} + \hat{P_D}) + P_{n+1}(t)\hat{P_D} \quad 0 < n < B \\ \frac{dP_B(t)}{dt} &= P_{B-1}(t)\hat{P_H} - P_B(t)\hat{P_D} \end{aligned} \tag{6.7}$$

Where $P_0(t)$ and $P_B(t)$ is the probability that the battery is empty (that is, the energy stored in the battery is completely depleted) at time t and the probability that the battery is fully charged to its maximum capacity at time t.

6.3.1 Transient-sate analysis

We use the method developed by Sharma and Gupta [218] for the transient analysis of an M/M/1/k queueing model to analyse the transient behaviour of the BESS. Suppose that $P_0(0) = 1$ and $P_n(0) = 0$, for $n \neq 0$ (that the BESS is initially empty), and taking the Laplace transform of the equations in (6.8) we have

$$\begin{aligned} (\hat{P_H} + s)\bar{P}_0(s) &= 1 + \bar{P}_1(s)\hat{P_D} \\ (\hat{P_H} + \hat{P_D} + s)\bar{P}_n(s) &= \bar{P}_{n-1}(s)\hat{P_H} + \bar{P}_{n+1}(s)\hat{P_D} \quad 0 < n < B \\ (\hat{P_D} + s)\bar{P}_B(s) &= \bar{P}_{B-1}(s)\hat{P_H} \end{aligned} \tag{6.8}$$

where,

$$\bar{P}_n(s) = \int_0^\infty e^{-st} P_n(t)dt$$

and,

$$\mathscr{L}\{\frac{P_n(t)}{dt}\} = s\bar{P}_n(s) - P_n(0)$$

Also, suppose that $P_0(0) = 0$ and $P_n(0) = 1$, for $n \neq 0$ (that the BESS is not empty initially), then the Laplace transform of the equations in (6.8) becomes

$$\begin{aligned} (\hat{P_H} + s)\bar{P}_0(s) &= \bar{P}_1(s)\hat{P_D} \\ (\hat{P_H} + \hat{P_D} + s)\bar{P}_n(s) &= 1 + \bar{P}_{n-1}(s)\hat{P_H} + \bar{P}_{n+1}(s)\hat{P_D} \quad 0 < n < B \\ (\hat{P_D} + s)\bar{P}_B(s) &= 1 + \bar{P}_{B-1}(s)\hat{P_H} \end{aligned} \tag{6.9}$$

The set of equations in 6.9 and 6.10 can be solved using any linear solver to obtain the state probabilities. The solution to the equations in (6.9) were developed by the authors in [218]. Therefore, the transient-state probability of having n energy packets in the BESS is given by

$$P_n(s) = \frac{(\chi\omega)^n[\chi^{B-n+1} - \omega^{B-n+1}] - (\chi\omega)^{n+1}[\chi^{B-n} - \omega^{B-n}]}{s[\chi^{B+1} - \omega^{B+1}]} \tag{6.10}$$

where,

$$\chi(s) = \frac{s + \hat{P_H} + \hat{P_D} + \sqrt{(s + \hat{P_H} + \hat{P_D})^2 - 4\hat{P_H}\hat{P_D}}}{2\hat{P_D}},$$

and

$$\omega(s) = \frac{s + \hat{P_H} + \hat{P_D} - \sqrt{(s + \hat{P_H} + \hat{P_D})^2 - 4\hat{P_H}\hat{P_D}}}{2\hat{P_D}},$$

The transient state probability of having $n = 0$ energy units in the BESS is

$$P_0(s) = \frac{[\chi^{B+1} - \omega^{B+1}] - (\chi\omega)[\chi^B - \omega^B]}{s[\chi^{B+1} - \omega^{B+1}]} \tag{6.11}$$

and the transient state probability of having $n = B$ energy units in the BESS is

$$P_B(s) = \frac{(\chi\omega)^B[\chi - \omega]}{s[\chi^{B+1} - \omega^{B+1}]} \tag{6.12}$$

For large capacity of the BESS, which is possible if the BESS is oversized,

$$\lim_{B\to\infty} P_n(s) = \frac{(1-\omega)\varrho^n}{s\chi^n} \tag{6.13}$$

We consider some numerical examples to study the transient state probabilities and the influence of energy supply-demand ratio, ϱ on the transient state probability. We invert the Laplace transform in (6.10) numerically using the Stehfest algorithm [230].

In Figure 6.3.1, it can be seen that the probability that energy stored in the battery will be completely depleted, $P_0(t)$ decreases with time and then attains steady-state. The observed decrease is because, we assumed that the BESS is empty at time $t = 0$ (e.g, $P_0(0) = 1$). It can also be seen that for small values of ϱ, the decrease is very rapid, while for larger values of ϱ the decrease is very gradual. After some time, $P_0(t)$ attains steady-state, and the steady-state value attained decreases with increasing value of ϱ as shown in Figure 6.3.2 in the next subsection below.

Fig. 6.3.1 shows the transient behaviour of the state probabilities of having n energy packets in the BESS. The state probabilities increase for $n > 0$ and then attain a steady state. In this case, it is assumed

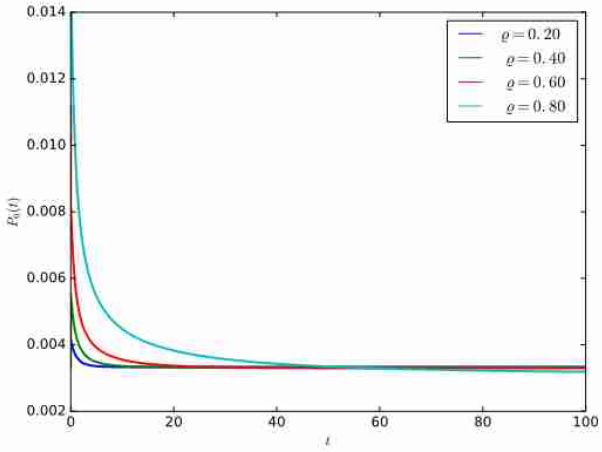

Figure 6.2: The influence of ϱ on the transient-state probability that the energy stored in the BESS is completely depleted.

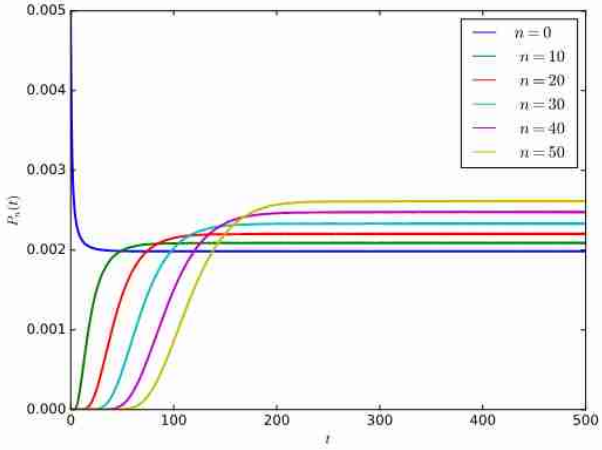

Figure 6.3: The transient-state probabilities for the BESS, for $\varrho = 0.60$.

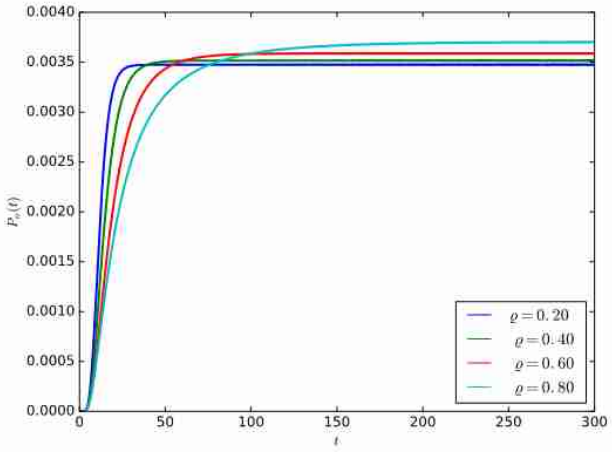

Figure 6.4: The influence of ϱ on the transient-state probability for $n = 10$.

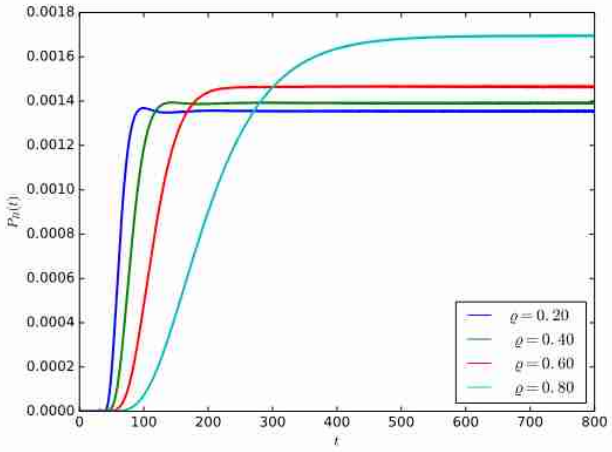

Figure 6.5: The influence of ϱ on the transient-state probability that the energy stored in the BESS is fully charged.

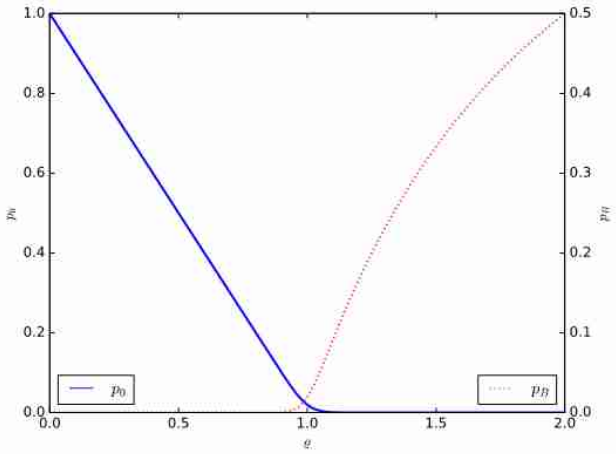

Figure 6.6: The variation of P_0 and P_B with the energy generation to demand ratio, ϱ.

that at $t = 0$, the BESS is empty and $P_n(0) = 0$ for $n > 0$. For small values of n, the increase is rapid, and for larger values of n, the increase is gradual.

Figure 6.3.1 shows the influence of ϱ on the transient-state probability for $n = 10$. As ϱ increases, it takes longer to attain steady-state. A similar observation can be seen in figure 6.3.1 on the influence of ϱ on the transient-state probability, $P_B(t)$ that the energy stored in the BESS is fully charged. When $P_B(t)$ attains steady-state, its steady-state value increases with increasing ϱ as shown in Figure 6.3.2 in the next subsection below.

6.3.2 Steady-state analysis

In steady-state, the equations in (6.8) becomes linear equations, and the steady-state probability distribution of the number of energy units in the battery is given by

$$P_n = \begin{cases} P_0 \varrho^n & \text{for} \quad \varrho \neq 1 \ , \\ \dfrac{1}{B+1} & \text{for} \quad \varrho = 1 \end{cases} \tag{6.14}$$

where $\varrho = \frac{\hat{P_H}}{\hat{P_D}}$ is the energy generation to demand ratio. The probability that the energy stored in the battery is completely depleted is an important performance metrics because when there is no energy in the battery, the base station will be shutdown, resulting in service outage which is undesirable for users and results in financial loses for the operator. The steady-state probability that the energy stored in the battery is completely depleted is given by

$$P_0 = \begin{cases} \dfrac{1-\varrho}{1-\varrho^{B+1}} & \text{for} \quad \varrho \neq 1 \ , \\ \dfrac{1}{B+1} & \text{for} \quad \varrho = 1 \end{cases} \tag{6.15}$$

The probability that the battery is fully charged is important because when the battery is fully charged, additional energy generated could be wasted. The photovoltaic resourse should be provisioned in such as

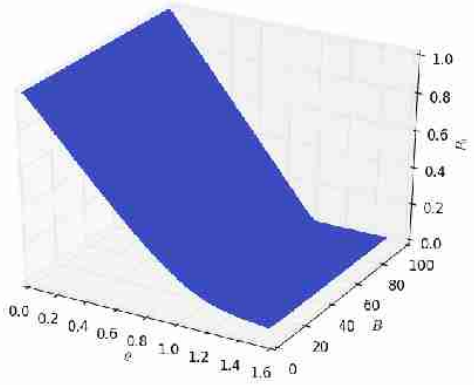

Figure 6.7: The variation of battery depletion probability, P_0 with the energy generation to demand ratio, ϱ and BESS capacity B.

way as to ensure that it generates sufficient amount of energy to ensure that the battery does not become empty. However, the capacity of the battery is limited (due to cost and technological limitations), and it is unneccessary to deploy more sources to generate more energy in which a majority of it is wasted. The steady-state probability that the battery is fully charged is

$$P_B = \begin{cases} P_0 \varrho^B & \text{for} \quad \varrho \neq 1 \ , \\ \dfrac{1}{B+1} & \text{for} \quad \varrho = 1 \end{cases} \tag{6.16}$$

Figure 6.3.2 shows that the probability that the energy stored in the battery is completely depleted, P_0 decreases as the energy generation to demand ratio, ϱ increases. For $\varrho < 1$ ($\hat{P_H} < \hat{P_D}$), the battery is discharging and there is a higher probability that the battery will be completely drained. Also, for $\varrho \geqslant 1$ ($\hat{P_H} \geqslant \hat{P_D}$), the probability that the battery will be completely drained is low but the probability that the battery will become fully charged is higher. Therefore, the resources of the photovoltaic system should be determined in such a way as to ensure that value of ϱ is as high as possible, but reasonable enough as to keep the probability of overcharging the battery within reasonable limits. By selecting desirable or target values of P_0 and ϱ, the capacity of the battery can be determined as

$$B = \begin{cases} \dfrac{log\left[1 - \frac{1}{P_0}(1-\varrho)\right]}{log(\varrho)} - 1 & \text{for} \quad \varrho \neq 1 \ , \\ \dfrac{1-P_0}{P_0} & \text{for} \quad \varrho = 1 \end{cases} \tag{6.17}$$

The capacity of the battery should be a large as possible to ensure that enough energy is stored during the periods where a lot of energy is generated and then used when little or no energy is generated. The capacity should be large enough to ensure that energy generated is not wasted to due to battery overcharging. However, the capacity of the battery that can be selected is limited cost, size, energy density of the material, and the battery technology.

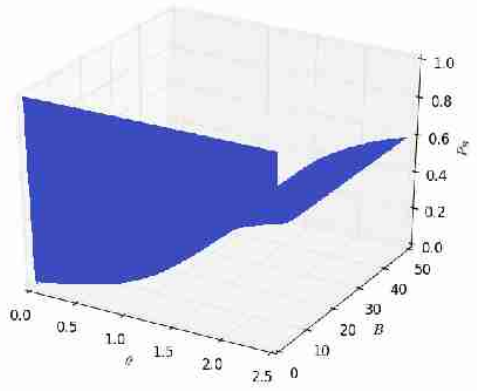

Figure 6.8: The variation of energy wastage probability, P_B with the energy generation to demand ratio, ϱ and BESS capacity B.

Figure 6.3.2 shows the variation of energy depletion probability, P_0 with the energy generation to demand ratio, ϱ and BESS capacity B. Based on the energy demands on the base station site, the renewable energy generation system should be designed in such a way as to ensure that the energy depletion probability, P_0 is as low as possible. Figure 6.3.2 shows the variation of energy energy wastage probability, P_B with the energy generation to demand ratio, ϱ and BESS capacity B. It is desirable that $\varrho > 1$, to reduce the probability of completely draining the battery, but it increases the probability of overcharging the battery P_B, which damages the battery and also waste excess energy. The energy capacity of the BESS decreases over time, and sharply increases the probability of battery overcharging P_B. Therefore, The probability of battery overcharging increases sharply with decreasing battery capacity B and also increases sharply with increasing energy generation to demand ratio, ϱ.

6.4 Markovian models for the analysis of the time required to completely discharge or fully charge the BESS

6.4.1 The time after which the energy stored in the BESS is completely depleted

Suppose that initially, we have i energy packets (for $i \in [1\ B]$) in the battery at time $t = 0$. Also, let the random variable T represents the time after which the battery becomes empty which is the first passage time of the process from $N(0) = i$ to $N(t) = 0$ (when all the energy stored in the battery is completely depleted), that is $T = inf\{t > 0 : N(t) = 0 \mid N(0) = i\}$. This time is a very important parameter because when the energy stored in the battery is deleted, a more expensive backup energy source is activated, otherwise, the base station and all the other transmission equipments on the site will be shutdown resulting in financial loses for the mobile network operator. It is equivalent to the duration or the length of the busy period of an M/M/1/B Markovian process that started from $N(0) = i$ and ends at $N(t) = 0$. Therefore, PDF of the first passage time of a Markovian "birth-death" process in (6.8) from $N(0) = i$ to $N(t) = 0$ (duration of the

busy period), with a reflecting barrier at B can be expressed in the Laplace domain as [236]:

$$\bar{h}_{i,0}(s) = \varrho^{-i}\frac{[\eta(s)]^{B-i}[\eta(s)-1]+[\xi(s)]^{B-i}[\xi(s)-1]}{[\eta(s)]^{B}[\eta(s)-1]+[\xi(s)]^{B}[\xi(s)-1]} \tag{6.18}$$

where,

$$\xi(s) = \frac{s+\hat{P_H}+\hat{P_D}-\sqrt{(s+\hat{P_H}+\hat{P_D})^2-4\hat{P_H}\hat{P_D}}}{2\hat{P_H}},$$

and

$$\eta(s) = \frac{s+\hat{P_H}+\hat{P_D}+\sqrt{(s+\hat{P_H}+\hat{P_D})^2-4\hat{P_H}\hat{P_D}}}{2\hat{P_H}},$$

Its mean is

$$E[T] = \begin{cases} \dfrac{i}{\hat{P_D}-\hat{P_H}} - \varrho^{B-i+1}\dfrac{1-\varrho^i}{\hat{P_D}(1-\varrho)^2} & \text{for } \varrho \neq 1\ , \\ \dfrac{i(2B-i+1)}{2\hat{P_D}} & \text{for } \varrho = 1 \end{cases} \tag{6.19}$$

and variance

$$var(T) = E[T^2] - E[T] \tag{6.20}$$

where,

$$E[T^2] = \frac{i^2}{\hat{P_D}^2(1-\varrho)^2} + \frac{i(1+\varrho+2\varrho^{B-i+1}+2\varrho^{B+1})}{\hat{P_D}^2(1-\varrho)^3} \\ \frac{2\varrho^{B-i+1(1-\varrho^i)[2+2B(1-\varrho)+\varrho^{B+1}]}}{\hat{P_D}^2(1-\varrho)^4}$$

for $\varrho \neq 1$ and

$$E[T^2] = \frac{i[(i+1)(i-1)(i-2)+4B(2B^2+3B+2-i^2)]}{12\hat{P_D}^2}$$

for $\varrho = 1$.

Figure 6.4.1 shows the influence of i on the probability density of the time after which the energy stored in the BESS is completely depleted. It can be seen that as initial amount of energy stored in the BESS i increases, the probability of the time after which the energy stored in it is completely depleted decreases. Figure 6.4.1 shows the influence of ϱ on the probability density of the time after which the energy stored in the BESS is completely depleted. It can be seen that as the probability of the time after which the energy stored in the BESS is completly depleted decreases with increasing ratio of the energy supplied to the BESS to the energy drawn, ϱ. For very large battery capacity,

$$\lim_{B\to\infty} \bar{h}_{i,0}(s) = [\xi(s)]^i \tag{6.21}$$

and

$$E[T] = -\frac{i}{\hat{P_H}-\hat{P_D}} \text{ for } \varrho < 1 \text{ and } B- > \infty$$

Assuming that the BESS is initially empty and that the arrival of one unit of energy starts that charging process, then, the busy period of the BESS is

$$\bar{h}_{1,0}(s) = \varrho^{-1}\frac{[\eta(s)]^{B-1}[\eta(s)-1]+[\xi(s)]^{B-1}[\xi(s)-1]}{[\eta(s)]^{B}[\eta(s)-1]+[\xi(s)]^{B}[\xi(s)-1]} \tag{6.22}$$

Also, assuming that initially, the BESS is fully charged ($i = B$), then, the first passage time from $N(0) = B$ to $N(t) = 0$ is

$$\bar{h}_{B,0}(s) = \varrho^{-B}\frac{[\eta(s)-1]+[\xi(s)-1]}{[\eta(s)]^{B}[\eta(s)-1]+[\xi(s)]^{B}[\xi(s)-1]} \tag{6.23}$$

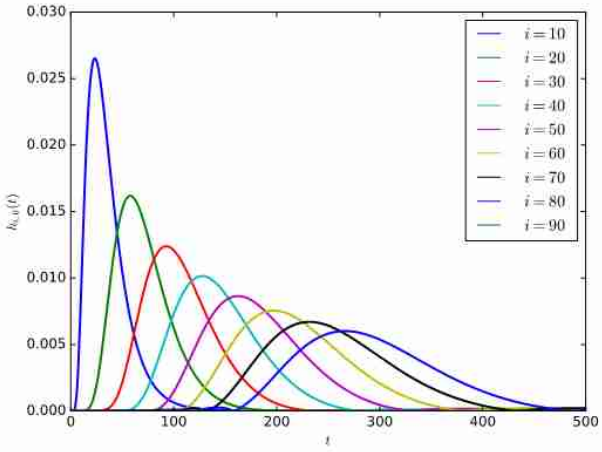

Figure 6.9: The influence of i on the probability density of the time after which the energy stored in the BESS is completely depleted, $h_{i,0}(t)$, for $\varrho = 0.60$.

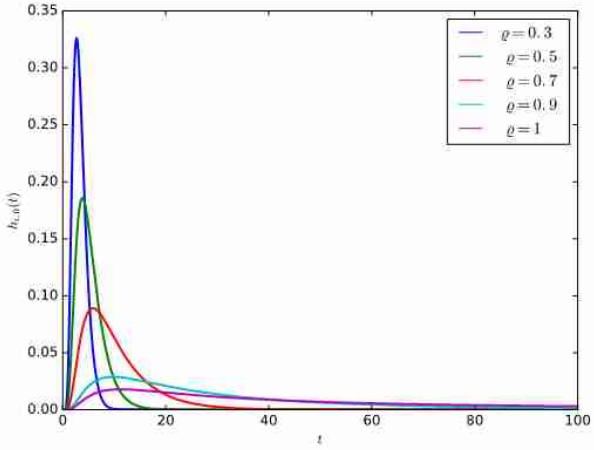

Figure 6.10: The influence of ϱ on the probability density of the time after which the energy stored in the BESS is completely depleted, $h_{i,0}(t)$, for $i = 10$.

6.4.2 The time after which the BESS is fully charged

It is the time required for the energy stored in the battery to increase from its initial amount of $i \in [0\ B-1]$ to the the full capacity of the BESS B. It is important to know the time required for the energy stored in the BESS to increase to its full capacity, B, for a given supply to demand ratio, ϱ. When the BESS is fully charged, if the base station cannot be supplied directed with the harvested energy, then it is will be wasted which is not desirable. It can be obtained by deriving the first passage time of the Markovian process from $N(0) = i$ to $N(t) = B$, with a reflecting barrier at 0. The time required for the energy stored in the BESS to increase from i to B is given in the Laplace domain can be adapted from equation (6.18) as:

$$\bar{h}_{i,B}(s) = \varrho^{-(B-i)} \frac{\{[\eta(s)]^{i+1} - [\xi(s)]^{i+1}\} - \{[\eta(s)]^{i} - [\xi(s)]^{i}\}}{\{[\eta(s)]^{B+1} - [\xi(s)]^{B+1}\} - \{[\eta(s)]^{B} - [\xi(s)]^{B}\}} \tag{6.24}$$

with mean,

$$E[T] = \begin{cases} \dfrac{B-i}{\hat{P_D} - \hat{P_H}} - \dfrac{\varrho^{-i} - \varrho^{-B}}{\hat{P_D}(\varrho - 1)^2} & \text{for } \varrho \neq 1\ , \\ \dfrac{(B-i)(B+i+1)}{2\hat{P_D}} & \text{for } \varrho = 1 \end{cases} \tag{6.25}$$

and variance

$$var(T) = E[T^2] - E[T] \tag{6.26}$$

where,

$$E[T^2] = \frac{(B-i)^2}{\hat{P_D}^2(\varrho-1)^2} + \frac{(B-i)(1+\varrho+2\varrho^{-i}+2\varrho^{-B})}{\hat{P_D}^2(\varrho-1)^3} - \frac{2\varrho^{-i}(1-\varrho-(B-i))[2\varrho+2B(\varrho-1)+\varrho-B]}{\hat{P_D}^2(\varrho-1)^4}$$

for $\varrho \neq 1$ and

$$E[T^2] = \frac{(B-i)\{(B-i+1)(B-i-1)(B-i-2)}{12\hat{P_D}^2} \frac{(B-i)\{4B(B^2+(3+2i)B+2-i^2)\}}{12\hat{P_D}^2}$$

for $\varrho = 1$. For very large battery capacity,

$$\lim_{B\to\infty} \bar{h}_{i,B}(s) = [\xi(s)]^{B-i} \tag{6.27}$$

and

$$E[T] = -\frac{B-i}{\hat{P_H} - \hat{P_D}} \text{ for } \varrho < 1 \text{ and } B- > \infty$$

Figure 6.4.2 shows the influence of the capacity of the BESS B, on the probability density of the time after which the BESS is fully charged. It can be observed that the mean time required to charge from i to its full capacity, decreases with decreasing value of B. When the BESS is full, any additional energy harvested is wasted. Figure 6.4.2 shows the influence of ϱ on the probability density of the time after which the BESS is fully charged. It can be seen that the probability of the time after which the BESS is fully charged increases with increasing ratio of the energy supplied to the BESS to the energy drawn ϱ. The energy harvesting resources should be provisionsed in such a way as to avoid quicktly over charging of the BESS.

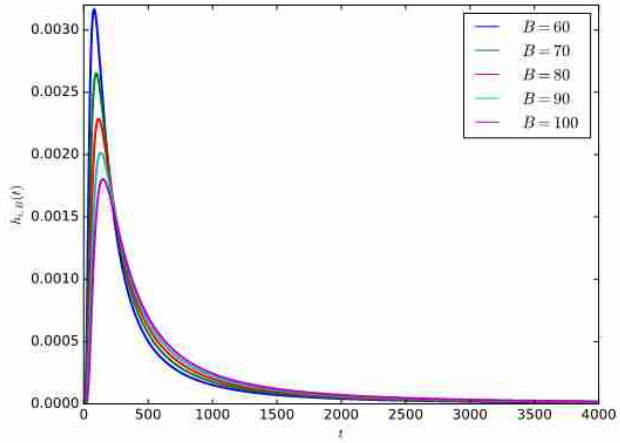

Figure 6.11: The influence of B on the probability density of the time after which the BESS is fully charged, $h_{i,B}(t)$, for $i = 10$, $\varrho = 0.60$.

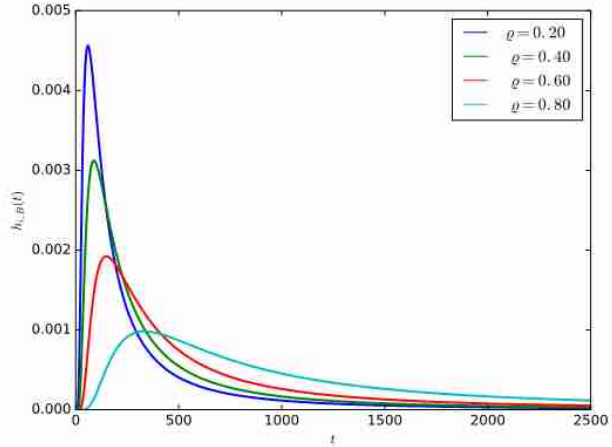

Figure 6.12: The influence of ϱ on the probability density of the time after which the BESS is fully charged, $h_{i,B}(t)$, for $i = 10$, $\varrho = 0.60$.

6.5 Diffusion model of the BESS for a base station site

The assumption that the energy generation process follows the Poison process and that the energy consumption process is exponentially distributed is not exactly true in reality. Other alternative approaches to realistically model the stochastic and continuous nature of energy is the use of fluid flow models [83] and diffusion approximation models [5]. In applying fluid flow models to model the time evolution of the energy stored in the battery, the changes in the amount of energy present in the battery at time t are considered to be analogous to the changes of a fluid in a reservoir. With diffusion approximation, the changes in the energy content of the buffer are modelled using diffusion or Brownian motion process, which is a continuous stochastic process. The advantage of using the diffusion approximation model is that it requires realistic distributions of the energy generation and consumption processes, which gives room for the use of historical data collected over time in the design and dimensioning of base station sites powered by renewable energy sources.

If we assume that the changes in the energy content of the battery $\Delta E(t) = E(t+\Delta) - E(t)$ are normally distributed, then we can approximate the energy changing process of the battery by a diffusion or Brownian motion process $X(t)$ whose changes $\Delta X(t) = X(t+\Delta) - X(t)$ are normally distributed with mean $\beta\Delta t$ and variance $\alpha\Delta t$ where:

$$\begin{aligned} \beta &= \lim_{\Delta t \to 0} \frac{E[X(t+\Delta t) - X(t)]}{\Delta t} \\ \alpha &= \lim_{\Delta t \to 0} \frac{Var[X(t+\Delta t) - X(t)]}{\Delta t}. \end{aligned} \tag{6.28}$$

The dynamic changes in the energy content of the battery (during charging and discharging) can then be described by the second order partial differential equation of a diffusion process [44]

$$\frac{\partial f(x,t;x_0)}{\partial t} = \frac{\alpha}{2}\frac{\partial^2 f(x,t;x_0)}{\partial x^2} - \beta\frac{\partial f(x,t;x_0)}{\partial x}. \tag{6.29}$$

The Diffuson process X(t) on the interval $[0, B]$ represents the amount of energy present in the battery at time t. Its probability density function $f(x,t;x_0)$ gives probability that the amount of the energy present in the battery at time t is x, given that the initial amount of energy in the battery was x_0. The model is equivelent to G/G/1/B queueing model, where the symbol G means that the distribution of the energy generation and and energy consumption are general (not limited to any particular distribution). By restricted the diffusion process, X(t) within the interval $[0, B]$, the diffusion equations describing the energy changes in the battery becomes [84]

$$\begin{aligned} \frac{\partial f(x,t;x_0)}{\partial t} &= \frac{\alpha}{2}\frac{\partial^2 f(x,t;x_0)}{\partial x^2} - \beta\frac{\partial f(x,t;x_0)}{\partial x} + \\ &\quad + \hat{P_H} p_0(t)\delta(x-1) + \hat{P_D} p_B(t)\delta(x-B+1)\ , \\ \frac{dp_0(t)}{dt} &= \lim_{x\to 0}\left[\frac{\alpha}{2}\frac{\partial f(x,t;x_0)}{\partial x} - \beta f(x,t;x_0)\right] - \hat{P_H} p_0(t)\ , \\ \frac{dp_B(t)}{dt} &= \lim_{x\to B}\left[-\frac{\alpha}{2}\frac{\partial f(x,t;x_0)}{\partial x} + \beta f(x,t;x_0)\right] - \hat{P_D} p_B(t)\ , \end{aligned} \tag{6.30}$$

where $\delta(x)$ is the Dirac delta function, $p_0(t)$, $p_B(t)$ are probabilities that the process is at the barriers at $x = 0$ or $x = B$, (battery is empty or fully charged). If the battery is charged at the beginning, then $p_B(0) = 1$.

6.5.1 The transient analysis

The transient solution of (6.30) may be computationaly obtained with the approach of [47], [56] presented in chapter three. In this chapter, we adapt the method to the analysis of the transient state probabilities of having x amount of energy in the BESS at time t. In the first step we consider a diffusion process with two absorbing barriers at $x = 0$ and $x = B$, that started at $t = 0$ from $x = x_0$. Its probability density function $\phi(x, t; x_0)$ has for $t > 0$ the following form [44]

$$\begin{aligned}&\phi(x,t;x_0) = \\ &\frac{1}{\sqrt{2\Pi\alpha t}}\sum_{n=-\infty}^{\infty}\{\exp[\frac{\beta x'_n}{\alpha} - \frac{(x - x_0 - x'_n - \beta t)^2}{2\alpha t}] \\ &-\exp[\frac{\beta x''_n}{\alpha} - \frac{(x - x_0 - x''_n - \beta t)^2}{2\alpha t}]\},\end{aligned} \tag{6.31}$$

where $x'_n = 2nB$, $x''_n = -2x_0 - x'_n$.

The Laplace transform of $\phi(x, t; x_0)$ is

$$\begin{aligned}&\bar{\phi}(x,s;x_0) = \\ &\frac{\exp[\frac{\beta(x-x_0)}{\alpha}]}{A(s)}\sum_{n=-\infty}^{\infty}\{\exp[-\frac{|x - x_0 - x'_n|}{\alpha}A(s)] \\ &-\exp[-\frac{|x - x_0 - x''_n|}{\alpha}A(s)]\},\end{aligned} \tag{6.32}$$

with $A(s) = \sqrt{\beta^2 + 2\alpha s}$.

The probability density function $f(x, t; B)$ of the diffusion process with jumps from the boundaries is composed of a spectrum of functions $\phi(x, t-\tau; 1)$, $\phi(x, t-\tau; B-1)$ representing diffusion processes with absorbing barriers at $x = 0$ and $x = B$, that started with densities $g_1(\tau)$ and $g_{B-1}(\tau)$ at time $\tau < t$ at points $x = 1$ and $x = B - 1$ due to jumps from the barriers:

$$\begin{aligned}f(x,t;B) &= \int_0^t g_1(\tau)\phi(x, t-\tau; 1)d\tau \\ &+ \int_0^t g_{B-1}(\tau)\phi(x, t-\tau; B-1)d\tau \\ &= g_1(t) * \phi(x,t;1) + g_{B-1}(t) * \phi(x,t;B-1)\end{aligned} \tag{6.33}$$

where $*$ denotes convolution, and densities $g_1(t)$, $g_{B-1}(t)$, as well as $p_0(t)$ and $p_B(t)$, are obtained from the probability balance equations at the barriers.

The densities $\gamma_0(t)$, $\gamma_B(t)$ of probability that at time t the process enters a barrier at $x = 0$ or $x = B$ are

$$
\begin{aligned}
\gamma_0(t) &= \int_0^t g_1(\tau)\gamma_{1,0}(t-\tau)d\tau \\
&+ \int_0^t g_{B-1}(\tau)\gamma_{B-1,0}(t-\tau)d\tau \\
&= g_1(t) * \gamma_{1,0}(t) + g_{B-1}(t) * \gamma_{B-1,0}(t), \\
\gamma_B(t) &= p_B(0)\delta(t) + \int_0^t g_1(\tau)\gamma_{1,B}(t-\tau)d\tau \\
&+ \int_0^t g_{B-1}(\tau)\gamma_{B-1,B}(t-\tau)d\tau = \\
&= p_B(0)\delta(t) + g_1(t) * \gamma_{1,B}(t) \\
&+ g_{B-1}(t) * \gamma_{B-1,B}(t),
\end{aligned} \tag{6.34}
$$

where $\gamma_{1,0}(t)$, $\gamma_{1,B}(t)$, $\gamma_{B-1,0}(t)$, $\gamma_{B-1,B}(t)$ are densities of the first passage times between the corresponding points. The density of the first passage time of the diffusion process from the point $x = x_0$ to $x = 0$ is the first passage time distribution for a diffusion process that starts from the point $x = x_0$ and is absorbed at $x = 0$ is

$$
\begin{aligned}
\gamma_{x_0,0}(t) &= \int_{0+}^{\infty} \frac{\partial f(x,t;x_0)}{\partial t} dx \\
&= \int_{0+}^{\infty} [\frac{\alpha}{2}\frac{\partial^2 f(x,t;x_0)}{\partial x^2} - \beta\frac{\partial f(x,t;x_0)}{\partial x}]dx \\
&= \lim_{x\to 0}[\frac{\alpha}{2}\frac{\partial f(x,t;x_0)}{\partial x} - \beta f(x,t;x_0] \\
&= \frac{x_0}{\sqrt{2\Pi\alpha t^3}} e^{-\frac{(x_0-\beta t)^2}{2\alpha t}} ,
\end{aligned} \tag{6.35}
$$

with the Laplace transform

$$
\tilde{\gamma}_{x_0,0}(s) = e^{-x_0\frac{\beta+\sqrt{\beta^2+2\alpha s}}{\alpha}} . \tag{6.36}
$$

Eq. (6.35) presents a probability density function in case of $\beta < 0$, when probability that the process will reach the barrier equals 1, and $\int_0^\infty \gamma_{x_0,0}(t)dt = 1$. Otherwise, for $\beta > 1$, the probability that the process ends at the barrier is $e^{-2\beta x_0/\alpha}$ and the conditional pdf is $\gamma'_{x_0,0}(t) = \gamma_{x_0,0}(t)e^{2\beta x_0/\alpha}$ and $\tilde{\gamma}'_{x_0,0}(s) = \tilde{\gamma}_{x_0,0}(s)e^{2\beta x_0/\alpha}$. The same refers to the case $\beta < 0$ with the initial point x_0 placed left to the absorbing barrier. The first passage time of the diffusion process from the point x_1 to another point x_2, $x_1 < x_2$ is

$$
\gamma_{x_1,x_2}(t) = \frac{x_2 - x_1}{\sqrt{2\Pi\alpha t^3}} e^{-\frac{(x_2-x_1-\beta t)^2}{2\alpha t}} \tag{6.37}
$$

and its corresponsing Laplace transform is

$$
\tilde{\gamma}_{x_1,x_2}(s) = e^{-(x_2-x_1)\frac{\beta+\sqrt{\beta^2+2\alpha s}}{\alpha}} . \tag{6.38}
$$

The densities of jumps in Eq. (6.33) depend on $\gamma_0(t)$, $\gamma_B(t)$ in the following way:

$$
\begin{aligned}
g_1(t) &= \int_0^t \gamma_0(t-\tau)l_0(\tau)d\tau = \gamma_0(t) * l_0(t), \\
g_{B-1}(t) &= \int_0^t \gamma_B(t-\tau)l_B(\tau)d\tau = \gamma_B(t) * l_B(t),
\end{aligned} \tag{6.39}
$$

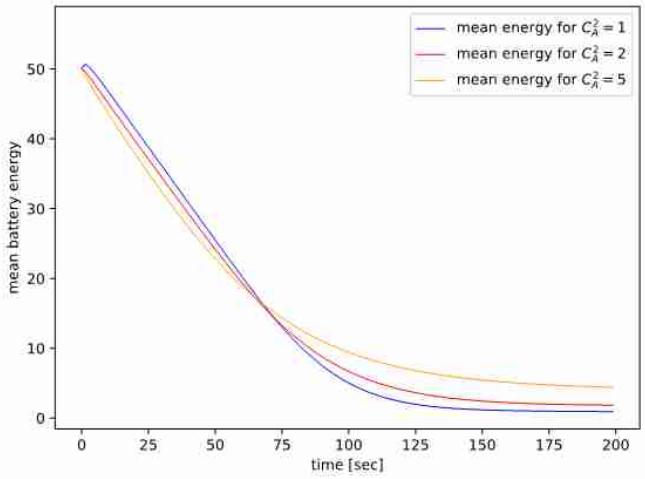

Figure 6.13: The influence of C_A^2, on the transient mean energy content of the BESS, $\lambda = 0.5$.

where $l_0(t)$, $l_B(t)$ are the densities of sojourn times in $x = 0$ and $x = B$ (they have means $1/\lambda$ and $1/\mu$ but are of general type).

With the use of (6.34) and (6.39) we obtain all information needed in the solution (6.33). To simplify the task, we use these equations in Laplace domain where the transform of the convolution of functions becomes a prodact of the transforms of these functions and at the end we seek numerically the original of the obtained transform of $f(x, t; B)$.

Probabilities that the proces is at barriers are

$$\begin{aligned} p_0(t) &= \int_0^t [\gamma_0(\tau) - g_1(\tau)]d\tau \,, \\ p_B(t) &= \int_0^t [\gamma_B(\tau) - g_{B-1}(\tau)]d\tau \,. \end{aligned} \tag{6.40}$$

Figure 6.5.1 shows the influence of C_A^2 on the transient mean energy content of the BESS. It was assumed that at time $t = 0$, the BESS is not empty. Since $\varrho < 1$, the energy content of the BESS decreases with with time and then attains steady state. For small values of C_A^2, the crease is rapid, and as C_A^2 increases, the decrease becomes slower.

Figures 6.5.1 shows the influence $\varrho = \frac{\lambda}{\mu}$, on the transient mean energy content of the BESS. For small values of ϱ, the crease is rapid, and as ϱ increases, the decrease becomes slower

The probaility that a BESS that started with x_0 amount of energy will never become empty is β is

$$1 - e^{\frac{-2\beta x_0}{\alpha}} \tag{6.41}$$

6.5.2 The steady analysis

Suppose that the battery is supplied by energy energy generated from a solar energy system, which converts solar energy directly into electrical energy, then the battery energy storage system is supplied by the photovoltaic output power. The output power of the photovoltaic system is given by:

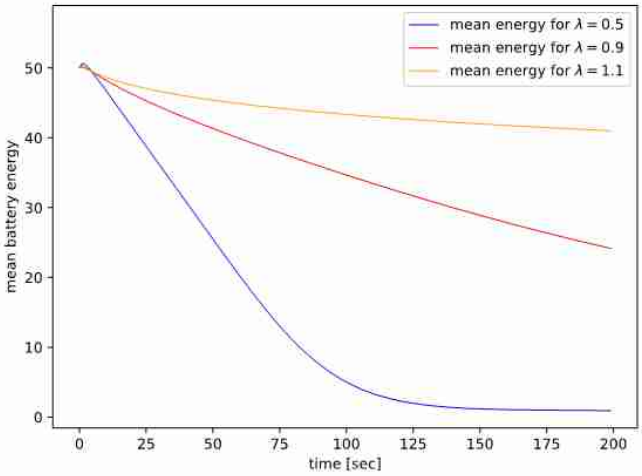

Figure 6.14: The influence of $\varrho = \frac{\lambda}{\mu}$, on the transient mean energy content of the BESS, $C_A^2 = 1$ $\mu = 1$.

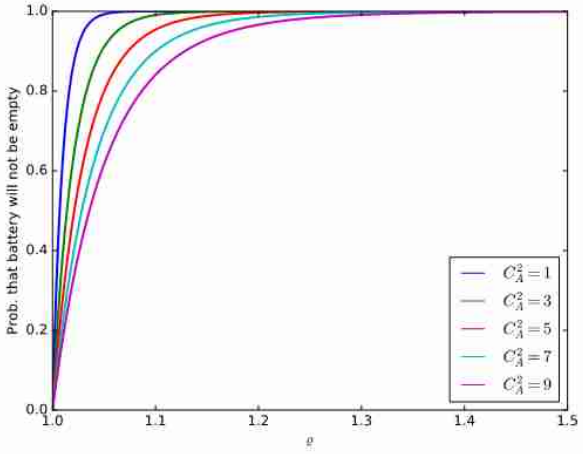

Figure 6.15: The influence of C_A^2 on the probabbility that a BESS that started with x_0 amount of energy will never become empty, for $\varrho = 0.8$, $\mu = 1$.

In steady-state, the partial differential equations in (6.30) become ordinary differential equations and the state probabilities for the amount of energy present in the battery are independent of time, e.g., $\lim_{t\to\infty} p_0(t) = p_0$, $\lim_{t\to\infty} p_B(t) = p_B$, $\lim_{t\to\infty} f(x,t;x_0) = f(x)$. The steady-state distribution of the amount of energy present in the buffer is

$$f(x) = \begin{cases} \dfrac{\hat{P_H} p_0}{-\beta}(1 - e^{zx}) & \text{for} \quad 0 < x \leqslant 1 \ , \\ \dfrac{\hat{P_H} p_0}{-\beta}(e^{-z} - 1)e^{zx} & \text{for} \quad 1 \leqslant x \leqslant B - 1 \ , \\ \dfrac{\hat{P_D} p_B}{-\beta}(e^{z(x-B)} - 1) & \text{for} \quad B - 1 \leqslant x < B \ , \end{cases} \tag{6.42}$$

where $z = \frac{2\beta}{\alpha}$, and p_0 is the probability that the battery is empty (that the energy stored in the battery is completely depleted), and p_B is the probability that the battery is fully charged. p_0 and p_B are obtained through normalization, $p_0 + f(x) + p_B = 1$, as:

$$p_0 = \begin{cases} \{1 + \varrho e^{z(B-1)} + \dfrac{\varrho}{1-\varrho}[1 - e^{z(B-1)}]\}^{-1} & \text{for } \varrho \neq 1, \\ \dfrac{1}{2[1 + (B-1)\hat{P_H}/\alpha]} & \text{for } \varrho = 1 \end{cases} \tag{6.43}$$

and

$$p_B = \begin{cases} \varrho p_0 e^{z(B-1)} & \text{for } \varrho \neq 1, \\ \dfrac{1}{2[1 + (B-1)\hat{P_H}/\alpha]} & \text{for } \varrho = 1 \end{cases} \tag{6.44}$$

For a given probability of completely depleting the amount of energy stored in the BESS, p_0 and the energy supply to load ration ϱ, the capacity of the BESS can be selected using the diffusion model as

$$B = \begin{cases} log\left(\dfrac{1}{\varrho^2}\left[1 - \dfrac{1-\varrho}{p_0}\right]\right) + 1 & \text{for} \quad \varrho \neq 1 \ , \\ \dfrac{\alpha}{2\hat{P_H}}\left[\dfrac{1}{p_0} + 2\left(\dfrac{\hat{P_H}}{\alpha} - 1\right)\right] & \text{for} \quad \varrho = 1 \end{cases} \tag{6.45}$$

Figure 6.5.2 shows the influence of ϱ on the steady state probability density of the amount of energy present in the BESS. For small values of $\varrho < 1$ the probability of having larger amounts of energy stored in the BESS is smaller and as ϱ increase the probability of having larger amounts of energy stored in the BESS in creases. For larger values of $\varrho > 1$, the probability of having smaller amount of energy stored in the BESS is smaller while the probability of having larger amounts of energy stored in the BESS increases.

The probability of having $x = 0$ amount of energy in the BESS, p_0 and the probability of having $x = B$ amount of energy in the BESS, p_B are also computed using the diffusion approximation model. The variation of p_0 and p_B with ϱ shown in 6.5.2 computed using the diffusion approximation model, and it is similar to the results obtained using the Markovian model shown in figure 6.3.2. A comparison of p_0 and p_B from the Markovian model and diffusion approximation is shown in figure 6.5.2.

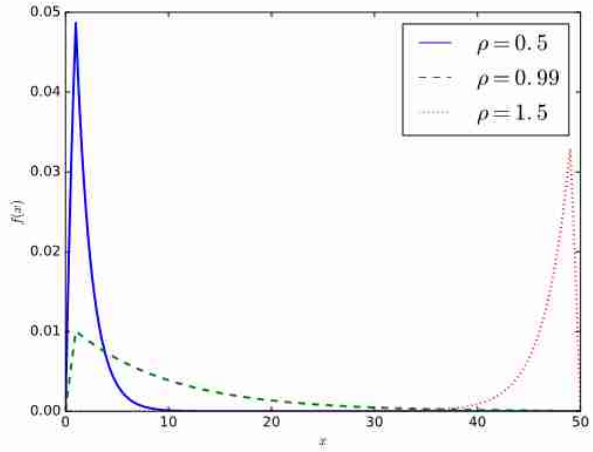

Figure 6.16: The influence of the energy generation to demand ratio, ϱ on the steady-state distribution of the amount of energy present in the battery, for $0 < x < B$.

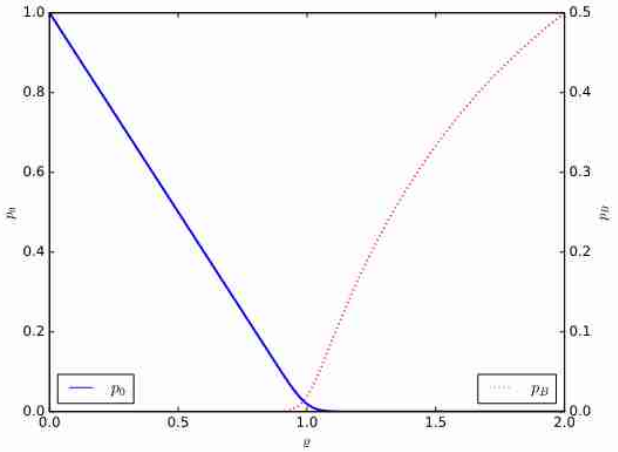

Figure 6.17: The variation of P_0 and P_B with the energy generation to demand ratio, ϱ: diffusion model.

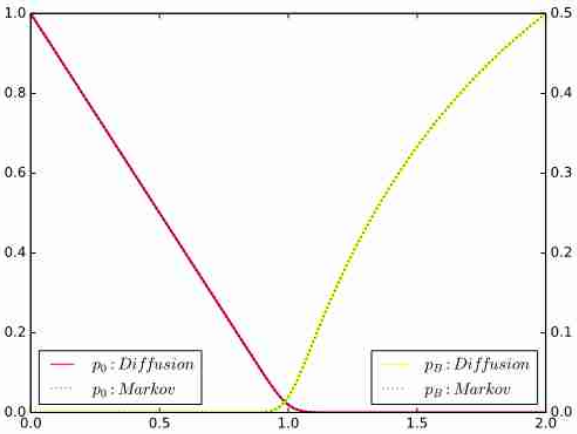

Figure 6.18: The variation of P_0 and P_B with the energy generation to demand ratio, ϱ: comparison of diffusion and Markov models model.

6.6 Diffusion models for the analysis of the time required to completely discharge or fully charge the BESS

6.6.1 The time after which the energy stored in the BESS is completely depleted

We use diffusion approximation to derive the time after which the energy stored in the BESS is completely depleted and then compare the results obtained using diffusion approximation with those obtained using Markovian model. Suppose that the random variable T represent the first passage time of the diffusion process from $X(0) = i$ (for $i > 0$) to $X(t) = 0$; that is, $T = inf\{t > 0 \ : X(t) = 0|X(0) = i\}$. The time after which the energy stored in the BESS is completely depleted, $h_{i,0}(t)$ is a combinations of the first passage time of a G/G/1/B diffusion process from $x = i$ to $x = 0$ or the duration of the busy period of a G/G/1/B diffusion process. The process may start at $x = i$ and moves directly to $x = 0$ or the process starts at $x = i$ and moves to $x = B$, stays there for a time that is exponentially distributed with mean μ, and then jumps to $x = B - 1$. From $x = B - 1$, the process can move to $x = 0$ and is absorbed or it can jump back to $x = B$, stays there for a time that is exponentially distributed and then jumps back to $x = B$.

To determine the probabilities of the possible sequence of movement of the diffusion process (e.g., whether a diffusion process that started at $x = i$ will move from $x = i$ to $x = 0$ or it will move to $x = B$ and if the process is at $x = B - 1$, the probaility that it will either move to $x = 0$ or to $x = B$). Let us consider a diffusion process that started at $x = x_0$ and is absorbed at $x = 0$, then the probability that the

process does not reach a certain x_n, for $x_n > x_0$, is given as

$$\begin{aligned} H(x_0, x_n) &= \int_0^\infty g(t, x_n; x_0)dt \\ &= \lim_{s\to 0}[\bar{\gamma}_0(s) - \bar{\gamma}_{x_0,x_n}(s)\bar{\gamma}_{x_n,0}(s)] \\ &= 1 - \lim_{s\to 0}\bar{\gamma}_{x_0,x_n}(s) \\ &= \frac{1-\exp[\frac{2\beta}{\alpha}(x_n - x_0)]}{1-\exp[\frac{2\beta}{\alpha}x_n]}. \end{aligned} \tag{6.46}$$

where

To determine the probabilities of all these sequences, assume at first a diffusion process having an absorbing barrier at $x = 0$, started at $x = x_0$ and compute the probability that it does not go beyond a certain x_n before ending at the origin. The pdf that the process is ended at t may be written as

$$\gamma_0(t) = g(t, x_n; x_0) + \int_0^t \gamma_{x_0,x_n}(\tau)\gamma_{x_n,0}(t-\tau)d\tau$$

where $g(t, x_n; x_0)$ is the probability density that the process will finish its motion at time t without reaching the point $x_n > x_0$, $\gamma_{x_0,x_n}(\tau)$ is the probability that the process that started at $x = 0$ will reach the point x_n for the first time at within the time τ, for $\tau < t$, and $\gamma_{x_n,0}(t-\tau)$ is the probability density that the process will pass from x_n to $x = 0$ during the time, $t - \tau$. The density that the process is ended at the barrier at $x = 0$, at time t is given as

$$\gamma_0(t) = g(t, x_n; x_0) + \int_0^t \gamma_{x_0,x_n}(\tau)\gamma_{x_n,0}(t-\tau)d\tau$$

It should be noted that for a function $f(x)$, and its Laplace transform $\bar{f}(s)$ we have $\bar{f}(0) = \int_0^\infty f(x)dx$, and if $f(x)$ is a probability density function defined for $x \geqslant 0$, $\bar{f}(0) = 1$. The duration of the busy period of a G/G/1/B diffusion process that starts at $X(0) = i$ and is absorbed at $X(t) = 0$ which represent the time after which the energy stored in the BESS is completely depleted is given by

$$\begin{aligned} h_{i,0}(t) &= H(i,B)\gamma'_{i,0}(t) + [1-H(i,B)]\{H(B-1,B)\gamma'_{i,B}(t) * l_B(t) * \gamma'_{B-1,0}(t) \\ &+ [1-H(B-1,B)]H(B-1,B)\gamma'_{i,B}(t) * l_B(t)^{2*}\gamma'_{B-1,B}(t) * \gamma'_{B-1,0}(t) \\ &+ [1-H(B-1,B)]^2 H(B-1,B)\gamma'_{i,B}(t) * l_B(t)^{3*}\gamma'_{B-1,B}(t)^{2*} * \gamma'_{B-1,0}(t) + \ldots\} \end{aligned} \tag{6.47}$$

with the Laplace transform

$$\begin{aligned} \bar{h}_{i,0}(s) &= H(i,B)\gamma'_{i,0}(t) + [1-H(i,B)]\{H(B-1,B)\gamma'_{i,B}(t) * l_B(t) * \gamma'_{B-1,0}(t) + \\ &+ [1-H(B-1,B)]H(B-1,B)\gamma'_{i,B}(t) * l_B(t)^{2*}\gamma'_{B-1,B}(t) * \gamma'_{B-1,0}(t) + \\ &+ [1-H(B-1,B)]^2 H(B-1,B)\gamma'_{i,B}(t) * l_B(t)^{3*}\gamma'_{B-1,B}(t)^{2*} * \gamma'_{B-1,0}(t) + \ldots\} \\ &= H(i,B)\bar{\gamma}'_{i,0}(s) + [1-H(i,B)]\bar{\gamma}'_{i,B}(s)\bar{\gamma}'_{B-1,0}(s)\frac{H(B-1,B)\bar{l}_B(s)}{1-[1-H(B-1,B)]\bar{\gamma}'_{B-1,B}(s)\bar{l}_B(s)} \end{aligned} \tag{6.48}$$

where ^{i*} denotes i-fold convolution, with its Laplace transform $\bar{h}_{i,0}(s)$ (we denote the Laplace transform of any function $f(t)$ by $\bar{f}(s)$)

Figure 6.6.1 shows the comparison of the Markovian and Diffusion models of the probability density of the time required to completely deplete the energy stored in the BESS, $h_{i,0}(t)$. The comparison is done for

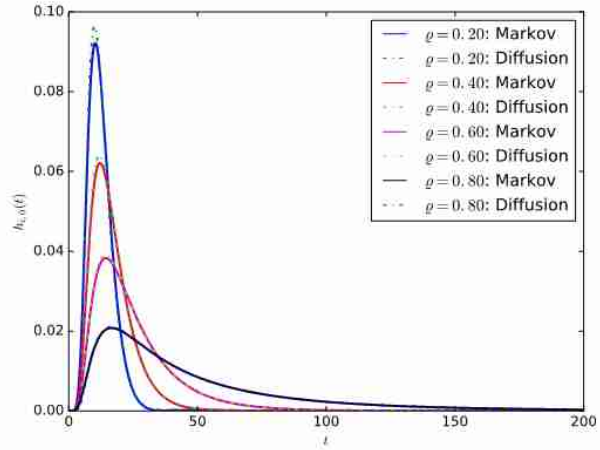

Figure 6.19: Comparison of the Markovian and Diffusion models of the probability density, $h_{i,0}(t)$, for $i = 10$, $\mu = 1$, $C_A^2 = C_B^2 = 1$, $B = 100$.

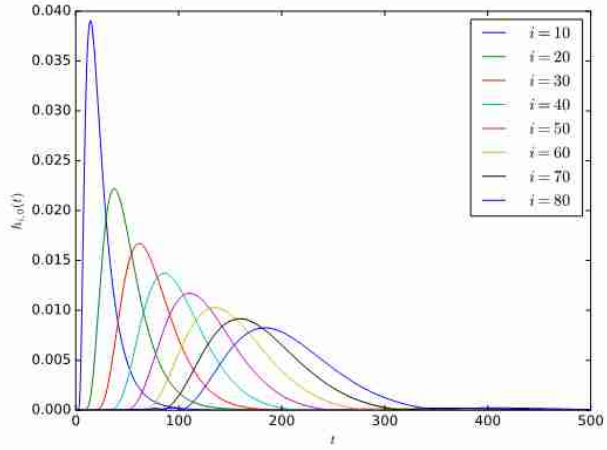

Figure 6.20: The influence of i on the probability density, $h_{i,0}(t)$, for $\mu = 0.6$, $\mu = 1$, $C_A^2 = C_B^2 = 1$, $B = 100$..

different values of ϱ and it can be seen that the densities from the two models are exactly the same. In order to compare these two models, we assume that $C_A^2 = C_B^2 = 1$. Figure 6.6.1 shows the influence of i on the density of the time required to completely deplete the energy stored in the BESS. The pattern is exactly the same as that obtained using Markovian models as shown in Figure 6.4.1. That is, for larger i, the average time required to completely deplete the energy stored in the BESS increases.

For very large battery capacity, $\lim_{B\to\infty} H(i,B) = 1$ and

$$\lim_{B\to\infty} \bar{h}_{i,0}(s) = \gamma'_{i,0}(s) \tag{6.49}$$

The average duration of the busy period $E[T]$, for very large B, is the same as that obtained using Markovian model in the previous section and is given by

$$E[T] = -\frac{i}{\hat{P_H} - \hat{P_D}} \text{ for } \varrho < 1 \text{ and } B- > \infty$$

6.6.2 The time after which the BESS is fully charged

We use diffusion approximation to determine the time required to charge the BESS to its full energy capacity. Suppose that at time $t = 0$, we have have i amount of energy in the BESS, then, the time required to charge the BESS to its full capacity is the first passage time of the G/G/1/B diffusion process from the point $x = i$ to $x = B$. The process starts at $x = i$, at time $t = 0$, and moves to $x = B$ and is absorbed with probability $H, 0$, or the process moves to $x = 0$, stays there for a time exponentially distributed with mean rate λ, and then jumps to $x = 1$. At $x = 1$, process can move to $X = B$, with probability $1 - H(1,i)$ and is absorbed or it moves to $x = 0$ with probability $H(1,i)$, stays there for a time that is exponentially distributed with mean rate λ, and the movements of the process are repeated until the process is absorbed at $x = B$. The time required for the process to move from $x = i$ to $x = B$, taking into consideration all the movements is

$$\begin{aligned} h_{i,B}(t) &= H(i,0)\gamma'_{i,B}(t) + [1 - H(i,0)]\{H(1,i)\gamma'_{i,0}(t) * l_0(t) * \gamma'_{1,B}(t) \\ &+ [1 - H(i,0)]H(1,i)\gamma'_{i,0}(t) * l_0(t)^{2*}\gamma'_{1,0}(t) * \gamma'_{1,B}(t) \\ &+ [1 - H(1,i)]^2 H(1,i)\gamma'_{i,0}(t) * l_0(t)^{3*}\gamma'_{1,0}(t)^{2*} * \gamma'_{1,B}(t) + \ldots\} \end{aligned} \tag{6.50}$$

with the Laplace transform

$$\begin{aligned} \bar{h}_{i,B}(s) &= H(i,0)\gamma'_{i,B}(t) + [1 - H(i,0)]\{H(1,i)\gamma'_{i,0}(t) * l_0(t) * \gamma'_{1,B}(t) + \\ &+ [1 - H(1,i)]H(1,i)\gamma'_{i,0}(t) * l_0(t)^{2*}\gamma'_{1,0}(t) * \gamma'_{1,B}(t) + \\ &+ [1 - H(1,i)]^2 H(1,i)\gamma'_{i,0}(t) * l_0(t)^{3*}\gamma'_{1,0}(t)^{2*} * \gamma'_{1,B}(t) + \ldots\} \\ &= H(i,0)\bar{\gamma}'_{i,B}(s) + [1 - H(i,0)]\bar{\gamma}'_{i,0}(s)\bar{\gamma}'_{1,B}(s)\frac{H(1,i)\bar{l}_0(s)}{1 - [1 - H(1,i)]\bar{\gamma}'_{1,0}(s)\bar{l}_0(s)} \end{aligned} \tag{6.51}$$

Figure 6.6.2 shows the comparison of the Markovian and Diffusion models of the probability density of the time required to charge the BESS to its full capacity, $h_{i,B}(t)$. It can be observed that the probability densities obtained using both models at exactly the same assuming that the energy supply and energy consumption processes are exponentially distributed (e.g., $C_A^2 = C_B^2 = 1$). Figure 6.6.2 shows the influence of

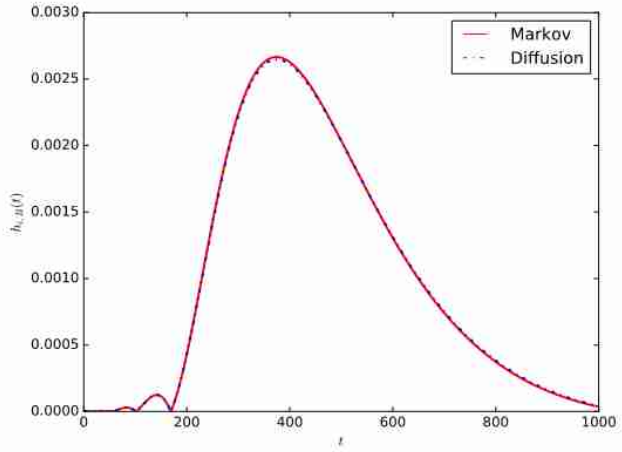

Figure 6.21: Comparison of the Markovian and Diffusion models of the probability desnisty, $h_{i,B}(t)$, for $i = 10$, $\varrho = 0.80$, $\mu = 1$, $C_A^2 = C_B^2 = 1$, $B = 100$.

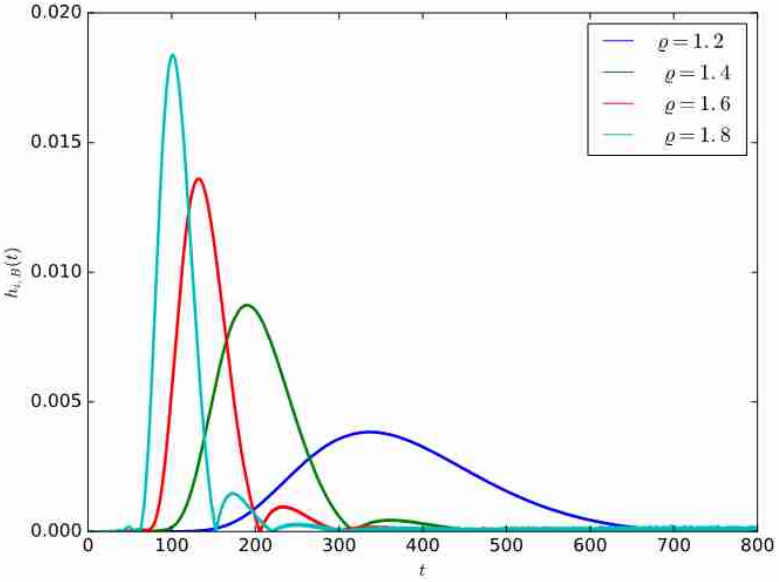

Figure 6.22: The influence of ϱ on the probability density, $h_{i,B}(t)$, for $i = 10$, $\mu = 1$, $C_A^2 = C_B^2 = 1$, $B = 100$.

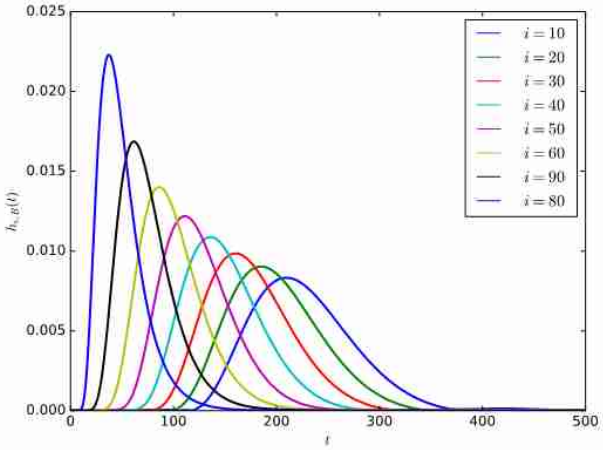

Figure 6.23: The influence of i on the probability density, $h_{i,B}(t)$, for $\varrho = 0.6$, $\mu = 1$, $C_A^2 = C_B^2 = 1$, $B = 100$.

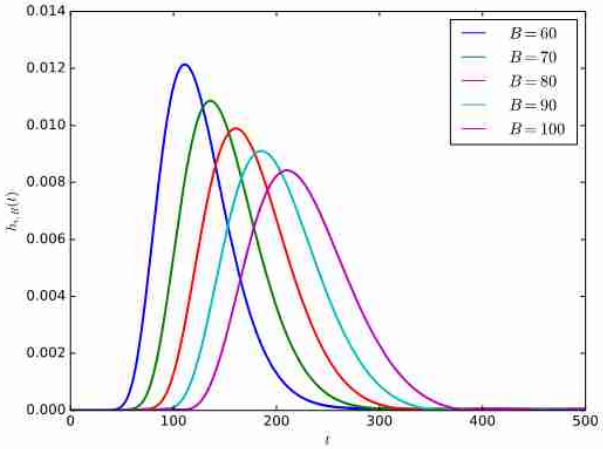

Figure 6.24: The influence of B on the probability density, $h_{i,B}(t)$, for $i = 10$, $\varrho = 0.6$, $\mu = 1$, $C_A^2 = C_B^2 = 1$.

ϱ on the probability density of the time required to charge the BESS to its full capacity, $h_{i,B}(t)$. It can be observed that as ϱ increases, density of the time required to charge the BESS to its full capacity increases.

Figure 6.6.2 shows the influence of i on the probability density of the time required to charge the BESS to its full capacity, $h_{i,B}(t)$. It can be seen that as i increases, the probability density of the time required to charge the BESS to its full capacity decreases. That is, for larger i, the average time required to charg the BESS to its full capacity decreases. Figure 6.6.2 shows the influence of B on the probability density of time required to charge the BESS to its full capacity, $h_{i,B}(t)$. It can be observed that for a fixed i, the mean time required to charge the BESS from i to its full capacity, B, decreases with decreasing value of B.

For very large capacity of the BESS,

$$\lim_{B\to\infty} \bar{h}_{i,B}(s) = \gamma'_{i,B}(s) \tag{6.52}$$

The average duration of the busy period $E[T]$, for very large B, is the same as that obtained using Markovian model in the previous section and is given by

$$E[T] = -\frac{B-i}{\hat{P_H} - \hat{P_D}} \text{ for } \varrho < 1 \text{ and } B- > \infty$$

6.7 Conclusion

We have analysed the performance of BESS that is continuously charged by renewable energy sources and discharged to supply base station. We have presented an architecture of a green base station site. We developed Markovian and diffusion approximation models for the analysis of steady-state and transient-state performance of battery energy storage systems. We applied Markovian and diffusion approximation model to derive the time after which the battery energy storage system is completely discharged or fully charged. By assuming that the energy harvesting and the energy consumption processes are exponentially distributed, we compared the result obtained from the Markovian model to those from diffusion approximation models.

Chapter 7

Conclusion

When studying, designing, or deploying computer systems and networks, researchers, systems architects, and designers are often required to understand the relationship between the design parameters and the performance metrics of the systems. That is, they seek to understand how the design parameters influence the performance of the various parts of the computer system or network and the performance of the computer system or network as a whole. The most popular methods used for planning, dimensioning, optimization, and performance evaluation of computer systems and networks are experimental test beds, real computer systems (or networks), simulations, or mathematical modelling.

Mathethical modelling together with computer simulations is the most commonly used method in the analysis and performance evaluations of computer systems and networks, although they have their limitations. One of the limitations of mathematical modelling and computer simulations is the assumptions that are often made which are usually different from reality. For example, Markovian models are often used for mathematical modelling and simulation of computer systems and networks due to their simplicity but the assumption that the jobs or packets arrive into queues in computer systems and network equipment following a Poison process and that the service times are exponentially distributed is sometimes far from reality. Thus, mathematical and simulation models are valid provided the assumptions are true which are sometimes far from reality.

7.1 Design and performance modelling of the packet aggregation algorithms and their applications

One of the problems addressed in this dissertation is the design and performance modelling of the packet aggregation algorithms and their applications in access (IoT and mobile networks), Core, and data center networks. A detailed review of packet aggregation algorithms implemented in IoT access networks, IoT over SDN networks, IoT over mobile radio networks, Cloud Radio Access networks, IP over all-optical networks, and cloud computing servers (to transfer data between driver domains and the virtual machine) was performed. The existing mathematical models for the design, analysis and evaluation of packet algorithms were based on the assumption that the interarrival of packets into the aggregation buffers follows a Poison process and that the packet sizes are exponentially distributed. These assumptions are not true in reality as revealed by the traffic datasets (of the interarrival times and packet sizes) that we downloaded from the

repository of the Center for Applied Internet Data Analysis (CAIDA) based at the University of California's San Diego Supercomputer Center. We also looked at the dataset of the arrival times of IoT traffic and it showed that the interarrival times of IoT packets are far from being Poison distributed. Thus, Poison-based packet aggregation performance evaluation models are far from reality and are likely to be less accurate.

We proposed a Brownian motion or diffusion-based performance evaluation model for the evaluation of packet aggregation algorithms. The novelty of our proposed models is that they can take real (measured) traffic parameters such as the mean arrival rate of packets, the mean of the packet sizes, the variance of the interarrival times of packets into the aggregation buffer, and the variance of the packet sizes as inputs and then the distribution of the performance metrics like the sizes of the aggregated packets and the interdeparture times (or the delays experienced by the packets in the aggregation buffer). We validated our proposed mathematical models with discrete event simulations. Despite the benefits of packet aggregation, they introduce the problem of delays which degrade the Quality of service (which is a problem for real-time traffic, especially in Industry 4.0 applications). We modelled the trade-off between throughput efficiency, energy consumption, and delays. As a continuation of this study, we intend to conduct empirical studies and mathematical modelling to evaluate the impact of packet aggregation on quality of service parameters like delays, packet dropping, and jitter in IoT traffic over IP networks (implementing priority-based mechanisms to sort and aggregate only non-real-time traffic).

7.2 Performance modelling of a Software Defined Networking (SDN) Switches and Network

Another problem that was adressed as part of the dissertation is the modelling of the performance of a Software Defined Networking (SDN) switch. Software defined networking (SDN) is a dynamic, adaptable, and manageable paradigm that facilitates innovations and rapid prototyping and deployment of flexible routing mechanisms in computer networks. Unlike traditional networking which involves manual configurations of distributed proprietary network devices, a cumbersome and error-prone process that can underutilize network resources, SDN offers a programmable architecture where routing decisions are moved to centralized controllers. The SDN data plane switches are simple forwarding devices that forward the data traffic depending on the controller's flow forwarding rules.

At the time of our study, most of the models for the design, analysis, and evaluation of the performance of SDN switches and networks were based on the assumption that the arrival of packets into an SDN switch follows a Poison process and that the packet processing times of an SDN switch are exponentially distributed. It was shown that these assumptions are inaccurate, therefore, better performance evaluation models are required. We proposed diffusion approximation models for the evaluation of the performance of an SDN switch. One of our key contribution was the modelling of the flow matching process or the process of searching the flow tables to find the appropriate flow rules required to process the packet (both in the case of hardware and software switches). We then used the mean packet arrival rate and the variance of the interarrival times from the CAIDA datasets together with the mean packet processing rate and the variance of the processing times (from our proposed packet processing model in an SDN switch) as an input to the diffusion models to estimate the performance metrics (delays and tail dropping of packets when the buffer is full).

In existing studies, it is assumed that the network is always in a steady state but in SDN networks where the controller may frequently change the routing decisions (like those developed during the SerIoT project for SDN core networks designed to be resilient, secure, ensure acceptable QoS, and to optimise the energy consumption), the network may frequently enter into transient states. The broader use of Software Defined Network (SDN) controllers to create periodic changes in the network's topology sometimes lead to changes in traffic intensities at the various switches. Thus, the SDN network constantly enters into transient states and the transient behaviour of network components, particularly data switches, is becoming of great interest. We extended the model for an SDN switch to a network model of SDN switches and investigated the influence of constantly changing traffic intensities resulting from changes logical topology of the network as the controller updates the routing tables of the switches with new flow rules.

The modelling of the performance of the controller was out of the scope of this study, although it could be a performance bottleneck. Packets whose flow rules are not found on the SDN switch have to ensure delays due to packet processing at the SDN switch and waiting for the controller to determine the flow rule for the packet and update the switches. A single controller serves multiple switches and could be a performance bottleneck. The performance modelling of an SDN switch was out of the scope of this dissertation. Future studies could model the processing mechanism of the controller and then apply the diffusion approximation models discussed in this study to evaluate the performance of an SDN controller.

7.3 Performance modelling of battery energy storage systems

We adapted Markovian and diffusion approximation models that we applied to the modelling of QoS in a computer network to the modelling of the energy depletion process in battery energy storage systems. We proposed a Brownian motion or diffusion-based model for the energy depletion process in an energy storage system without the presence of energy harvesting sources. We applied our model to study the problem of energy depletion attacks in IoT networks. We modelled ghost energy depletion attacks (GEDA) in IoT networks. The two main ghost energy depletion attacks in IoT networks that were modelled are the high computational load on device GEDA and the MAC misbehaviour GEDA. With high computational load GEDA, the adversary overwhelms its victims with bogus messages to quickly drain the energy stored in their batteries. While for a MAC misbehaviour GEDA, the ghost attacker deliberately abuses the MAC protocol (e.g., CSMA/CA protocol) to create collisions on the shared wireless channel, thus, forcing other devices within its interference range to consume more energy (quickly draining their batteries) and to deprive them of accessing the channel.

We used the diffusion-based model for the energy depletion process of a battery energy storage system to estimate the life of an IoT device that is powered by a battery (without any harvesting source). We then used the data on the energy consumption of an IoT device as the input to the diffusion model and estimate the impact of active power and the sleep time of the IoT device on its lifetime. It was shown that an attack model that is aimed at manipulating the sleep time (increasing the active power of the IoT) or manipulating transmit power of the device will quickly drain energy stored in the battery of the device and reduces its lifetime significantly. The sleep time of the device could be manipulated through high computational attacks (sending bogus messages and forcing the device to perform unnecessary energy-demanding security computations), creating collisions in the channel, and reconfiguring the device to generate more data for transmission than

necessary (increasing the sensing or measuring frequency).

We extended the model for the modelling of a battery energy storage system (in the absence of an energy harvesting source) to incorporate the presence of an energy harvesting source. We investigated the impact of the ratio of the mean energy harvesting rate to the energy consumption rate on the time required to completely drain the energy stored in the battery and on the time required to fully charge the battery. It was shown that as this ratio increases the time required to completely drain the battery increases while the time required to fully charge the battery decreases. Thus, when sizing the energy harvesting system it is important to ensure that this ratio is not far less to increase the time required to drain the battery (lifetime of the device) and reduce the time required to charge the battery to its full capacity.

We compared the results obtained using diffusion approximation to those obtained using the Markovian model. The comparisons showed an almost perfect match. The advantage of using diffusion models for the time evolution of the energy stored in the battery traditional queueing theoretic models and fluid flow models is that it takes into account fluctuations in the amount of energy harvested from the environment and the fluctuations in the amount of energy drawn from the battery (energy consumed). Another advantage of using diffusion approximation to other methods is that energy can be treated as a continuous quantity. The changes in the amount of energy in the battery could be considered to be analogous to the changes of a fluid in a reservoir (modelled using fluid flow differential equations). The changes in the amount of energy in the battery can also be modelled as a Brownian motion or diffusion process which is a continuous process and is suitable for modelling energy changes in the battery because energy is a continuous quantity. With data about the mean and variance of the energy harvesting or energy consumption process, the diffusion-based model of the battery can conveniently predict the lifetime of a computer system device (IoT device or drone).

The limitation of performance evaluation modelling is that the systems are often studied in isolation. It should be noted that improving the performance of a part of a computer system or network, does not guarantee the improvement of the performance of the computer system or network as a whole. That is, the performance of computer systems and networks is often influenced by a combination of multiple factors and by the performance of the various subsystems that constitute the computer system or networks. Thus, improving the performance of a computer system or network does not guarantee that performance of the computer system or network as a whole will be improved (some unintended problems may be created). Performance evaluation studies that treat the system as a whole and investigate the influence of queueing on the performance of the computer system and network as a whole may be more useful.

7.4 Future research direction

As a continuation of this study, we intend to apply diffusion approximation to study the problem of reneging and correlated reneging (impatience) in the performance evaluation of queues in computer systems. One of the significant challenges in cloud computing is the impact of impatient users or request (task) reneging. When a request has been compromised, missed its execution deadline, or depends on other rejected ones, it must be removed from the queue without being processed. The reneging or removal of tasks from queues may trigger the reneging or removal of other tasks that depend on them. We refer to this reneging of requests from the load balancing or computing queues as correlated reneging. Existing studies on reneging tasks from cloud computing servers did not take into account the possibility of correlated reneging (or

dropping of tasks without processing them). All of the studies that modelled the reneging or dropping of tasks from the cloud computing queues did not consider the possibility that the reneging of tasks could be correlated.

In [150, 151, 149], we developed queueing models for the evaluation of cloud computing queues with correlated reneging (sometimes with resubmission of reneged tasks) and used them to propose a framework for evaluating cloud computing infrastructures. Existing models on the modelling of reneging in cloud computing queues are based on the assumption that the arrival rates of tasks into the cloud computing buffers follow a Poison a Poisson process and that the processing times are exponentially distributed. It is also assumed that the reneging times are exponentially distributed. We intend to use diffusion approximation to study the problem of task reneging in cloud computing queues.

We adopted the queueing model with correlated reneging to model the problem of patient balking and reneging (impatience) in the health care management system in [147]. In a recent paper that was recently accepted in the International Journal of Services, Economics, and Management, we introduced a mechanism of customer retention (through incentives that motivate the customer to wait long and don't leave the queue) and adaptively decrease or increase the service time (by increasing or reducing the processing or service resources) depending on the queue size. We intend to conduct similar studies using diffusion approximation given its advantages in evaluating queues with real data. Diffusion approximation offers an effective modelling methodology for the time evolution of the waiting time or probability of rejection (for limited capacity queues).

Bibliography

[1] Prism – probabilistic model checker.

[2] Center for applied internet data analysis (CAIDA) based at the university of california's san diego supercomputer center february 2016 data sets, equinix-chicargo link (https://data.caida.org/datasets/passive-2016/equinix-chicago/20160218-130000.utc/), 2016.

[3] O. Abdelrahman and E. Gelenbe. A diffusion model for energy harvesting sensor nodes. In *Proc. of 24th IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 154–158. IEEE, 2016.

[4] O. H. Abdelrahman. A markov-modulated diffusion model for energy harvesting sensor nodes. *Probability in the Engineering and Informational Sciences*, 31:505–515, 2017.

[5] H. V. Abeywickrama, B. A. Jayawickrama, Y. He, and E. Dutkiewicz. Comprehensive energy consumption model for unmanned aerial vehicles, based on empirical studies of battery performance. *IEEE Access*, 6:58383–58394, 2018.

[6] B. Agrawal and T. Sherwood. Modeling tcam power for next generation network devices. In *Proceedings of the 2006 IEEE International Symposium on Performance Analysis of Systems and Software*, pages 120–129. IEEE, 2006.

[7] B. Agrawal and T. Sherwood. Ternary cam power and delay model: Extensions and uses. *IEEE Transactions on Very Large Scale Integration (VLSI)*, 16(5):554–564, 2008.

[8] A. Aktaş and YağmurKirçiçek. *Solar Hybrid Systems and Energy Storage Systems*. Elsevier, 2021.

[9] A. S. Akyurek and T. S. Rosing. Optimal packet aggregation scheduling in wireless networks. *IEEE Transactions on Mobile Computing*, 17(12):2835–2852, 2018.

[10] M. Al-Hubaishi, C. Ceken, and A. Al-Shaikhli. A novel energy-aware routing mechanism for sdn-enabled wsan. *IEEE Transactions on Green Communications and Networking*, pages 1–17, 2018.

[11] A. Ali, G. A. Shah, and J. Arshad. Energy efficient techniques for m2m communication: A survey. *Journal of Network and Computer Applications*, 68:42–55, 2016.

[12] M. H. Alsharif, S. Kim, and N. Kuruoglu. Energy harvesting techniques for wireless sensor networks/radio-frequency identification: A review. *Symmetry*, 11(865):1–24, 2019.

[13] A.M.K.Tarabia. Transient analysis of m/m/1/n queue - an alternative approach. *Tamkang Journal of Science and Engineering*, 3(4):263–266, 2000.

[14] O. P. Angwech, A. S. Alfa, and B. Maharaj. Managing the harvested energy in wireless sensor networks: A priority geo/geo/1/k approach with threshold. *Energy Reports*, 8:2448–2461, 2022.

[15] J. Ansell, W. K. G. Seah, B. Ng, and S. Marshall. Making queueing theory more palatable to SDN/openflow-based network practitioners. In *Proceeding of the 2016 IEEE/IFIP Network Operations and Management Symposium*, pages 1119–1124, Istanbul, Turkey, 2016. IEEE.

[16] S. Arabi, E. Sabir, H. Elbiaze, and M. Sadik. Data gathering and energy transfer dilemma in uav-assisted flying access network for iot. *Sensors*, 18(1519):1–20, 2018.

[17] T. Atmaca, A. Kamli, G. S. Kuaban, and T. Czachórski. Performance evaluation of the packet aggregation mechanism of an N-GREEN metro network node. In C. M. et al., editor, *MASCOTS 2020: Modelling, Analysis, and Simulation of Computer and Telecommunication Systems, Lecture Notes in Computer Science*, volume 12527, pages 62–78, Milan, Italy, 2021. Springer, Cham.

[18] S. Azodolmolky, R. Nejabati, M. Pazouki, P. Wieder, R. Yahyapour, and D. Simeonidou. An analytical model for software defined networking: A network calculus-based approach. In *Proceeding of the IEEE Global Communications Conference*, pages 1397–1402, Atlanta, USA, 2013. IEEE.

[19] S. Azodolmolky, P. Wieder, and R. Yahyapour. Performance evaluation of a scalable software-defined networking deployment. In *Proceeding of the 2013 Second European Workshop on Software Defined Networks*, pages 68–74, Berlin, Germany, 2013. IEEE.

[20] D. Baek, Y. Chen, and M. Poncino. Battery-aware energy model of drone delivery tasks. In *Proceedings of the International Symposium on Low Power Electronics and Design*, pages 1–6. ACM, 2018.

[21] J. Bauer and N. Aschenbruck. Measuring and adapting MQTT in cellular networks for collaborative smart farming. In *Proceedings of the 2017 IEEE 42nd Conference on Local Computer Networks (LCN)*, pages 294–302, Singapore, 2017. IEEE.

[22] S. Bhandari, S. K. Sharma, and X. Wang. Latency minimization in wireless iot using prioritized channel access and data aggregation. In *Proceedings of the GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, pages 1–6, Singapore, 2017. IEEE.

[23] N. Bisnik and A. A. Abouzeid. Queuing network models for delay analysis of multihop wireless ad hoc networks. *Ad Hoc Networks 7*, 7(1):79–97, 2009.

[24] V. Bonneau and B. Copigneaux. Industry 4.0 in agriculture: Focus on iot aspects, 2017.

[25] V. Bonnin, E. Bénard, J.-M. Moschetta, and C. Toomer. Energy-harvesting mechanisms for uav flight by dynamic soaring. *International Journal of Micro Air Vehicles*, 7(3):212–230, 2015.

[26] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, ennifer Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker. P4: Programming protocol-independent packet processors. *ACM SIGCOMM Computer Communication Review*, 44(3):88–95, 2014.

[27] M. Bourguiba, K. Haddadou, I. E. Korbi, and G. Pujolle. Improving network I/O virtualization for cloud computing. *IEEE Transactions on Parallel and Distributed Systems*, 25(3):673–681, 2014.

[28] M. Bourguiba, K. Haddadou, and G. Pujolle. Packet aggregation based network i/o virtualization for cloud computing. *Computer Communications*, 35:309–319, 2012.

[29] Z. Bozakov and A. Rizk. Aming sdn controllers in heterogeneous hardware environments. In *Proceeding of the 2013 Second European Workshop on Software Defined Networks*, pages 50–55, Berlin, Germany, 2013. IEEE.

[30] P. J. Burke. The output of a queuing system. *Operations Research*, 4(6):699–704, 1956.

[31] R. Buyya et al. A manifesto for future generation cloud computing: Research directions for the next decade. *ACM Computing Surveys (CSUR)*, 51(5):1–38, 2019.

[32] L. X. Cai, Y. Liu, T. H. Luan, X. S. Shen, J. W. Mark, and H. V. Poor. Sustainability analysis and resource management for wireless mesh networks with renewable energy supplies. *IEEE Journal On Selected Areas in Communications*, 32(2):345–355, 2014.

[33] X. Cao, D. M. Shila, Y. Cheng, Z. Yang, Y. Zhou, and J. Chen. Ghost-in-zigbee: Energy depletion attack on zigbee-based wireless networks. *Internet of Things Journal*, 3(5):816–829, 2016.

[34] B. Carsten, Castellani, P. Angelo, and S. Zach. Coap: An application protocol for billions of tiny internet nodes. *IEEE Internet Computing*, 16(2):62–67, 2012.

[35] D. C. Champernowne. An elementary method of solution of the queueing problem with a single server and constant parameters. *J.R. Statist. Soc*, 3(4):263–266, 2000.

[36] W. C. Chan, T. C. Lu, and R. J. Chen. Pollaczek-khinechin formula for the m/g/1 queue in discrete time with vacations. *IEE Proc. Comput. Digit. Tech.*, 44(4):222–226, 1997.

[37] Y. Chen, D. Baek, A. Bocca, A. Macii, E. Macii, and M. Poncino. A case for a battery-aware model of drone energy consumption. In *Proceedings of the 2018 IEEE International Telecommunications Energy Conference (INTELEC)*, pages 1–8. IEEE, 2018.

[38] Y. Cheng and M. Huang. Based on the internet of things environment data aggregation security requirements and key technology research. In *Proceedings of the 2019 International Conference on Information Technology and Computer Application (ITCA)*, pages 19–24, Guangzhou, China, 2019. IEEE.

[39] S. Chiamsiri and S. Moore. Accuracy comparisons between two diffusion approximations for MX/G/l queues - instantaneous return vs. reflecting boundary, 1977.

[40] J. Choi, H. L. Vu, C. W. Cameron, M. Zukerman, and M. Kang. The effect of burst assembly on performance of optical burst switched networks. In H.-K. Kahng and S. Goto, editors, *Lecture Notes in Computer Science*, pages 729–739. Springer, 2004.

[41] R. K. Chung, L. Santucci, M. Koriyama, E. Lee, H. Lee, S. Chung, M. Sato, J. Jung, S. Zhu, P. Hoontrakul, A. J. Chin, and Y. Monoe. Decentralized energy system, 2012.

[42] S. Conti, G. Faraci, R. Nicolosi, S. A. Rizzo, and G. Schembra. Battery management in a green fog-computing node: a reinforcement-learning approach. *IEEE Access*, 5:21126–21138, 2017.

[43] R. P. Cox and H. D. Miller. *The Theory of Stochastic Processes*. Chapman and Hall, London, UK, 1965.

[44] C. E. Cramer and E. Gelenbe. Video quality and traffic qos in learning-based subsampled and receiver-interpolated video sequences. *IEEE Journal on Selected Areas in Communications*, 18(2):150–167, 2000.

[45] E. D. Cuypere, K. D. Turck, and D. Fiems. A queueing model of an energy harvesting sensor node with data buffering. *Telecommun Syst*, 67:281–295, 2018.

[46] T. Czachórski. A method to solve diffusion equation with instantaneous return processes acting as boundary conditions. *Bulletin of Polish Academy of Sciences, Technical Sciences*, 41(4), 1993.

[47] T. Czachórski. Queuing models for performance evaluation of computer networks-transient state analysis. In M. R. V. Mityushev, editor, *Analytic Methods in Interdisciplinary Applications*, volume 116, pages 51–80. Springer, 2014.

[48] T. Czachórski, A. Domańska, J. Domańska, and A. Rataj. A study of ip router queues with the use of markov models. In *Proceeding of the 2016 International Conference on Computer Networks (CN2016)*, pages 294–305. Springer, 2016.

[49] T. Czachórski, E.Gelenbe, K. Suila, and D.Marek. Time dependent diffusion model for security driven software defined networks. In *Proc. of the Second International Workshop on Stochastic Modeling and Applied Research of Technology (SMARTY 2020)*, pages 38–56, 2020.

[50] T. Czachórski, E.Gelenbe, K. Suila, and D.Marek. Transient behaviour of a network router. In N. Herencsar and F. Benedetto, editors, *Proc. of 43th International Conference on Telecommunications and Signal Processing (TSP), Milan, Italy*, pages 246–252. IEEE, 2020.

[51] T. Czachórski, J.-M. Fournau, S. Jędruś, and F. Pekergin. Transient state analysis in cellular networks: the use of diffusion approximation. In *Proceeding of the Fourth International Workshop on Queueing Networks with Finite Capacity QNETs 2000*, Ilkley, West Yorksire, UK, 2000.

[52] T. Czachórski, E. Gelenbe, and G. S. Kuaban. Modelling energy changes in the energy harvesting battery of an iot device. In *Proceedings of the Modelling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, Nice, France, 2022. IEEE.

[53] T. Czachórski, E. Gelenbe, G. S. Kuaban, and D. Marek. Time dependent diffusion model for security driven software defined networks. In *Proceedings of the Second International Workshop on Stochastic Modeling and Applied Research of Technology (SMARTY 2020)*, volume 2792, pages 38–56, 2020.

[54] T. Czachórski, E. Gelenbe, G. S. Kuaban, and D. Marek. Transient behaviour of a network router. In *Proceedings of the 43th International Conference on Telecommunications and Signal Processing (TSP)*, pages 246–252, Milan, Italy, 2020. IEEE.

[55] T. Czachorski, E. Gelenbe, G. S. Kuaban, and D. Marek. Transient behaviour of a network router. In *Proceedings of the 43th International Conference on Telecommunications and Signal Processing*, Milano, Italy, 2020. IEEE.

[56] T. Czachórski, E. Gelenbe, G. S. Kuaban, and D. Marek. Time-dependent performance of a multi-hop software defined network. *Applied Sciences*, 11(2469):1–21, 2021.

[57] T. Czachórski, E. Gelenbe, G. S. Kuaban, and D. Marek. Energy optimization for an unmanned aerial vehicle (e.g drone) during its mission. In *Security in Computer and Information Sciences. EuroCybersec 2021. Communications in Computer and Information Science*, volume 1596, pages 61–75. Springer, 2022.

[58] T. Czachórski, G. S. Kuaban, and T. Nycz. Multichannel diffusion approximation models for the evaluation of multichannel communication networks. In K. D. e. Vishnevskiy V., Samouylov K., editor, *Distributed Computer and Communication Networks. DCCN 2019. Lecture Notes in Computer Science*, volume 11965, pages 43–57, Moscow, Russia, 2019. Springer.

[59] T. Czachórski, M. Nycz, and T. Nycz. Modelling transient states in queueing models of computer networks: A few practical issues. In V. V. et al., editor, *Distributed Computer and Communication Networks*, volume 279, pages 58–72, Moscow, Russia, 2013. Springer.

[60] T. Czachórski and F. Pekergin. Diffusion approximation as a modelling tool. In D. D. Kouvatsos, editor, *Network Performance Engineering*, volume 5233, pages 447–476. Springer, 2011.

[61] J. Deng and Mark.Davis. An adaptive packet aggregation algorithm for wireless networks. In *Proceedings of the 2013 International Conference on Wireless Communications and Signal Processing*, pages 1–6, Hangzhou, China, 2013. IEEE.

[62] V. Desnitsky and I. Kotenko. Simulation and assessment of battery depletion attacks on unmanned aerial vehicles for crisis management infrastructures. *Simulation Modelling Practice and Theory*, 107:102244, 2021.

[63] S. Dobson et al. A survey of autonomic communications. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 1(2):223–259, 2006.

[64] J. Domańska, I. Kotuliak, T. Atmaca, and T. Czachórski. Optical packet filling. In *Proceedings of the 10th Polish Teletraffic Symposium PSRT2003*, Krakow, Poland, 2003.

[65] A. Duda. Diffusion approximations for time-dependent queueing systems. *IEEE Journal on Selected Areas in Communications*, 4:905–918, 1986.

[66] F. Dürr and T. Kohler. *Comparing the Forwarding Latency of OpenFlow Hardware and Software Switches*. University of Stuttgart, Faculty of Computer Science, Electrical Engineering, and Information Technology, Technical Report Computer Science No. 2014/04, english, 2014.

[67] D. L. C. Dutra, M. Bagaa, T. Taleb, and K. Samdanis. Ensuring end-to-end qos based on multi-paths routing using sdn technology. In *Proceeding of the GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, pages 1–6, Singapore, 2017. IEEE.

[68] H. Elahi, K. Munir, M. Eugeni, and P. Gaudenzi. Energy harvesting towards self-powered iot devices. *Energy*, 13(5528):1–31, 2020.

[69] Enerdata. Open vswitch flow table analysis.

[70] Enerdata. Between 10 and 202030?, 2018.

[71] G. Erol and K. Y. Murat. Battery attacks on sensors: Wireless nodes with battery attacks. In *EuroCybersec 2018: International symposium on computer and information sciences, cybersecurity workshop*. Springer Cham, 2018.

[72] G. Erol and K. Y. Murat. Energy life-time of wireless nodes with network attacks and mitigation. In *2018 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1–6. IEEE, 2018.

[73] M. A. Ertürk, M. A. Aydın, and M. T. Büyükakkaslar. A survey on LoRaWAN architecture, protocol and technologies. *Future Internet*, 11(216):1–34, 2019.

[74] J. Euchi. Do drones have a realistic place in a pandemic fight for delivering medical supplies in healthcare systems problems? (in press). *Chinese Journal of Aeronautics*, 2020.

[75] A. Fahmin, Y.-C. Lai, S. Hossain, Y.-D. Lin, and D. Saha. Performance modeling of sdn with nfv under or aside the controller. In *Proceeding of the 5th International Conference on Future Internet of Things and Cloud Workshops*, pages 211–216, Prague, 2017. IEEE.

[76] D. Farquharson, P. Jaramillo, and C. Samaras. Sustainability implications of electricity outages in sub-saharan africa. *Nature Sustainability*, 1:589–597, 2018.

[77] F. Francois and E. Gelenbe. Towards a cognitive routing engine for software defined networks. In *Proceeding of the 2016 IEEE International Conference on Communications*, pages 1–6, Kuala Lumpur, 2016. IEEE.

[78] F. Francois and E. Gelenbe. Towards a cognitive routing engine for software defined networks. In *Proceeding of the 2016 IEEE International Conference on Communications (ICC)*, pages 1–6, Kuala Lumpur, 2016. IEEE.

[79] P. Frohlich, E. Gelenbe, J. Fiolka, J. Checinski, M. Nowak, and Z. Filus. Smart SDN management of fog services to optimize QoS and energy. *Sensors*, 21(3105):1–18, 2021.

[80] N. Garg and R. Garg. Energy harvesting in iot devices: A survey. In *Proceedings of the International Conference on Intelligent Sustainable Systems*, pages 131–135. IEEE, 2017.

[81] E. Garsva, N. Paulauskas, G. Grazulevicius, and L. Gulbinovic. Packet inter-arrival time distribution in academic computer network. *Elektronika IR Elektrotechnika*, 20(20):87–90, 2014.

[82] A. Gautam and S. Dharmaraja. An analytical model driven by fluid queue for battery life time of a user equipment in LTE-A networks. *Physical Communication*, 30:213–219, 2018.

[83] E. Gelenbe. On approximate computer system models. *Journal of the ACM (JACM)*, 22(2):261–269, 1975.

[84] E. Gelenbe. On approximate computer systems models. *Journal of the ACM*, 22(2):261–269, 1975.

[85] E. Gelenbe. Probabilistic models of computer systems part ii: Diffusion approximations, waiting times and batch arrivals. *Acta Informatica*, 12:285–303, 1979.

[86] E. Gelenbe. Energy packet networks: Ict based energy allocation and storage. In *International Conference on Green Communications and Networking*, pages 186–195. Springer, 2011.

[87] E. Gelenbe. Energy packet networks: adaptive energy management for the cloud. In *CloudCP'12: Proceedings of the 2nd International Workshop on Cloud Computing Platforms*, pages 1–5. ACM, https://doi.org/10.1145/2168697.2168698, 2012.

[88] E. Gelenbe. Energy packet networks: smart electricity storage to meet surges in demand. In *SimuTools*, pages 1–7, 2012.

[89] E. Gelenbe. Adaptive management of energy packets. In *COMPSAC Workshops*, pages 1–6. IEEE Computer Society, 2014.

[90] E. Gelenbe. A sensor node with energy harvesting. *ACM SIGMETRICS Performance Evaluation Review*, 42(2):37–39, 2014.

[91] E. Gelenbe and O. H. Abdelrahman. An energy packet network model for mobile networks with energy harvesting. *Nonlinear Theory and Its Applications*, 9(3):322–322, 2018.

[92] E. Gelenbe, J. Domanska, P. Frohlich, M. Nowak, and S. Nowak. Self-aware networks that optimize security, qos and energy. *Proceedings of the IEEE*, 108(7):1150–1167, 2020.

[93] E. Gelenbe and R. Iasnogorodski. A queue with server of walking type (autonomous service). In *Annales de l'IHP Probabilités et statistiques*, volume 16, pages 63–73, 1980.

[94] E. Gelenbe and Y. M. Kadioglu. Energy loss through standby and leakage in energy harvesting wireles sensors. In *CAMAD*, pages 231–236. IEEE, 2015.

[95] E. Gelenbe and Y. M. Kadioglu. Energy life-time of wireless nodes with network attacks and mitigation. In *Proceedings of the 2018 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1–6. IEEE, 2018.

[96] E. Gelenbe, R. Lent, and A. Nunez. Self-aware networks and qos. *Proceedings of the IEEE*, 92(9):1478–1489, 2004.

[97] E. Gelenbe, P. Liu, and J. Lainé. Genetic algorithms for route discovery. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 36(6):1247–1254, 2006.

[98] E. Gelenbe, X. Mang, and R. O. Onvural. Diffusion based statistical call admission control in ATM. *Performance Evaluation*, 27(12):411–436, 1996.

[99] E. Gelenbe and C. Morfopoulou. A framework for energy-aware routing in packet networks. *Comput. J.*, 54(6):850–859, 2011.

[100] E. Gelenbe and G. Pujolle. The behaviour of a single-queue in a general queueing network. *Acta Inf.*, 7:123–136, 1976.

[101] E. Gelenbe and K. Sevcik. Analysis of update synchronization for multiple copy data-bases. In *3rd Berkeley Workshop on Distributed Data and Computer Networks*, pages 69–90, 1978.

[102] E. Gelenbe and S. Silvestri. Reducing power consumption in wired networks. In *2009 24th International Symposium on Computer and Information Sciences*, pages 292–297. IEEE, 2009.

[103] E. Gelenbe et al. Self-aware networks that optimize security, qos and energy. *Proceedings of the IEEE, accepted for publication*, 108(7), 2020.

[104] A. Ghiasian. Impact of TCAM size on power efficiency in a network of openflow switches. *IET Networks*, 9(6):367–371, 2020.

[105] P. W. Glynn. *Diffusion Approximations*, volume 2. Elsevier, North-Holland, 1990.

[106] D. Gopi, S. Cheng, and R. Huck. Comparative analysis of sdn and conventional networks using routing protocols. In *Proceeding of the 2017 International Conference on Computer, Information and Telecommunication Systems (CITS)*, pages 108–112, Dalian, 2017. IEEE.

[107] Y. Goto, B. Ng, W. K.G.Seah, and Y. Takahashi. Queueing analysis of software defined network with realistic openflow-based switch model. *Computer Networks*, 164, 2019.

[108] A. Gowda, J. A. Hernández, D. L. López, and L. Kazovsky. Delay analysis of mixed fronthaul and backhaul traffic under strict priority queueing discipline in a 5G packet transport network. *Trans Emerging Tel Tech*, pages 1–9, 2017.

[109] S. Gŕtz. *Battery energy storage for intermittent renewable electricity production*. Umea University, New York, 2015.

[110] J. W. Guck, A. V. Bemten, M. Reisslein, and W. Kellerer. Unicast qos routing algorithms for sdn: A comprehensive survey and performance evaluation. *IEEE Communications Surveys and Tutorials*, 20(1):388–415, 2018.

[111] Z. Guo, I. G. Harris, Y. Jiang, and L. feng Tsaur. An efficient approach to prevent battery exhaustion attack on ble-based mesh networks. In *Proceeding of the 2017 International Conference on Computing, Networking and Communications (ICNC)*, pages 1–5, Orlando, Florida, 2017. IEEE.

[112] N. Ha and N. Kim. Efficient flow table management scheme in sdn-based cloud computing networks. *Journal of Information Processing Systems*, 14(1):228–238, 2018.

[113] J. A. Hernández, J. Aracil, V. López, and J. L. de Vergara. On the analysis of burst-assembly delay in OBS networks and applications in delay-based service differentiation. *Photon Network Commununi-cation*, 14:49–62, 2007.

[114] K. Hinton, F. Jalali, and A. Matin. Energy consumption modelling of optical networks. *Photon Netw Commun*, 30:4–16, 2015.

[115] M. H. Homaei, E. Salwana, and S. Shamshirband. An enhanced distributed data aggregation method in the internet of things. *Sensors*, 19(3173):1–26, 2019.

[116] B. A. Homssi, A. Al-Hourani, S. Chandrasekharan, K. M. Gomez, and S. Kandeepan. On the bound of energy consumption in cellular iot networks. *IEEE Transactions on Green Communications and Networking*, 4(2):355–364, 2019.

[117] J. Hu and A. Lanzon. An innovative tri-rotor drone and associated distributed aerial drone swarm control. *Robotics and Autonomous Systems*, 103:162–174, 2018.

[118] O. C. Ibe. *Markov Processes for Stochastic Modeling*. Elsevier, 2013.

[119] P. P. Ioulianou, V. G. Vassilakis, and M. D. Logothetis. Battery drain denial-of-service attacks and defenses in the internet of things. *Journal of Telecommunications and Information Technology*, 2:37–45, 2019.

[120] R. H. Jhaveri, R. Tan, and S. V. Ramani. Real-time qos-aware routing scheme in sdn-based robotic cyber-physical systems. In *IEEE 5th International Conference on Mechatronics System and Robots (ICMSR)*. IEEE, May 2019.

[121] W. Jiawei, Q. Xiuquan, and N. Guoshun. Dynamic and adaptive multi-path routing algorithm based on software-defined network. *International Journal of Distributed Sensor Network*, 14(10):1–10, 2018.

[122] G. L. Jones, P. G. Harrison, U. Harder, and T. Field. Fluid queue models of battery life. In *Proceeding of the 2011 IEEE 19th Annual International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems*, pages 278–285. IEEE, 2011.

[123] S. K. Jones, R. K. Cavin, and D. A. Johnston. An efficient computational procedure for the evaluation of the $m/m/1$ transient state occupancy probabilities. *IEEE Trans. on Comm.*, 28(12):2019–2020, 1980.

[124] J. M. Jornet and I. F. Akyildiz. Joint energy harvesting and communication analysis for perpetual wireless nanosensor networks in the terahertz band. *IEEE Transactions on Nanotechnology*, 11(3):570–580, 2012.

[125] A. Kamli. *Analysis and Optimisation of a new futuristic optical network architecture*. PhD thesis, Ecole doctorale n=580 Sciences et technologies de l'information et de communication (STIC), Université Paris-Saclay, France, 2020.

[126] A. Kamli, T. Atmaca, C. Lepers, A. Rataj, and D. Amar. Performance improvement of colored optical packet switching thanks to time slot sharing. In *Proceedings of the 14th Advanced International Conference on Telecommunications*, pages 12–31, Barcelona, Spain, 2018. IARIA.

[127] M. Karakus and A. Durresi. A scalable inter-as qos routing architecture in software defined network (SDN). In *IEEE 29th International Conference on Advanced Information Networking and Applications*, pages 148–154, 2015.

[128] M. Karakus and A. Durresi. A survey: Control plane scalability issues and approaches in software-defined networking (sdn). *Computer Networks*, 112:279–293, 2017.

[129] A. Khan, S. N. K. Marwat, S. Ahmed, and Y. Mehmood. Packet aggregation in mobile networks for IoT traffic. In *Proceedings of the 2019 IEEE 6th International Conference on Engineering Technologies and Applied Sciences (ICETAS)*, pages 1–4, Kuala Lumpur, Malaysia, 2019. IEEE.

[130] A. Khan, S. N. K. Marwat, S. Ahmed, and Y. Mehmood. Packet aggregation in mobile networks for IoT traffic. In *Proceedings of the 2019 IEEE 6th International Conference on Engineering Technologies and Applied Sciences (ICETAS)*, pages 1–4, Kuala Lumpur, Malaysia, 2019. IEEE.

[131] T. Kimura. Diffusion approximation for an m/g/m queue. *Operations Research*, 31(2):304–321, 1983.

[132] H. Kobayashi. Application of the diffusion approximation to queueing networks, part 1: Equilibrium queue distributions. *J.ACM*, 21(2):316–328, 1974.

[133] H. Kobayashi. Application of the diffusion approximation to queueing networks, part 1i: Noneequilibrium distributions and applications to computer modeling. *J.ACM*, 21(3):459–469, 1974.

[134] H. Kobayashi. *Modeling and Analysis: An Introduction to System Performance Evaluation Methodology*. Addison-Wesley, Reading, Massachusetts, 1978.

[135] A. Koike, T. Ohba, and R. Ishibashi. IoT network architecture using packet aggregation and disaggregation. In *Proceedings of the 2016 5th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI)*, pages 1140–1145, Kumamoto, Japan, 2016. IEEE.

[136] V. Konakov and E. Mammen. Accuracy of Diffusion Approximations for High Frequency Markov Data. working paper or preprint, 2006.

[137] C. Koparan, A. B. Koc, C. V. Privette, and C. B. Sawyer. In situ water quality measurements using an unmanned aerial vehicle (uav) system. *Water*, 10(3):264, March 2018.

[138] C. Koparan, A. B. Koc, C. V. Privette, and C. B. Sawyer. Autonomous in situ measurements of noncontaminant water quality indicators and sample collection with a uav. *Water*, 11(3):604, March 2019.

[139] C. Koparan, A. B. Koc, C. V. Privette, and C. B. Sawyer. Adaptive water sampling device for aerial robots. *Drones*, 4(1):5, 2020.

[140] T. C. T. Kotiah. Approximate transient analysis of some queueing systems. *Operations Research*, 26(2):334–346, 1998.

[141] W. Kraemer and M. Langenbach-Belz. Approximate formulae for the delay in the queueing system gi/g/1. In *Proceedings of the 8th International Telegraphic Congress, volume*, volume 2(3), page 235/1 235/8, Melbourne, Australia, 1976.

[142] G. S. Kuaban, E. Anyam, T. Czachórski, and A. Rataj. Performance of a buffer between electronic and all-optical networks, diffusion approximation model. In C. T. et al., editor, *ISCIS 2018: Computer and Information Sciences, Communications in Computer and Information Science*, volume 935, pages 68–75, Poznań, Poland, 2018. Springer, Cham.

[143] G. S. Kuaban, T. Atmaca, T. Czachórski, and P. Czekalski. Performance analysis of packet aggregation mechanisms and their applications in access (e.g., IoT, 4G/5G), core, and data centre networks. *Sensors*, 21(3898):1–33, 2021.

[144] G. S. Kuaban, T. Czachórski, and A. Rataj. A queueing model of the edge node in IP over all-optical networks. In P. G. et al., editor, *Proceedings of the International Conference on Computer Network (CN 2018)*, volume 860 of *CCIS*, pages 258–271, Gliwice, Poland, 2018. Springer.

[145] G. S. Kuaban, T. Czachórski, and A. Rataj. A queueing model of the edge node in IP over all-optical networks. In G. P. et al., editor, *Computer Networks. CN 2018. Communications in Computer and Information Science*, volume 860, pages 258–271, Gliwice, Poland, 2018. Springer, Cham.

[146] G. S. Kuaban, B. S. Soodan, R. Kumar, and P. Czekalski. Performance evaluations of a cloud computing physical machine with task reneging and task resubmission (feedback). In P. G. et al., editor, *Proceedings of the International Conference on Computer Networks (CN2020)*, pages 185–198, Gdansk, Poland, 2020. Springer.

[147] G. S. Kuaban, B. S. Soodan, R. Kumar, and P. Czekalski. Analysis of the performance of a cloud computing processing queue with correlated reneging of tasks and resubmission. In *Proceedings of the 2021 International Conference on Electrical, Computer and Energy Technologies (ICECET)*, pages 1–8. IEEE, 2021.

[148] G. S. Kuaban, B. S. Soodan, R. Kumar, and P. Czekalski. Analysis of the performance of a cloud computing processing queue with correlated reneging of tasks and resubmission. In *Proceedings of the 2021 International Conference on Electrical, Computer and Energy Technologies (ICECET)*, pages 1–8. IEEE, 2022.

[149] G. S. Kuaban, B. S. Soodan, R. Kumar, and P. Czekalski. A queueing-theoretic analysis of the performance of a cloud computing infrastructure: Accounting for task reneging or dropping. In *Proceedings of the 2022 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*, pages 1–7. IEEE, 2022.

[150] R. Kumar, B. S. Soodan, G. S. Kuaban, P. Czekalski, and S. Sharma. Performance analysis of a cloud computing system using queuing model with correlated task reneging. *Journal of Physics:Conference Series, presented in 5th International Scientific Conference on Information, Control, and Communication Technologies (ICCT-2021)*, 2091(012003):4–7, 2021.

[151] M. Kwiatkowska, G. Norman, and D. Parker. Prism 4.0: Verification of probabilistic real-time systems. In *Proceeding of the 23rd International Conference on Computer Aided Verification (CAV'11)*, pages 585–591, 2011.

[152] Y.-C. Lai, A. Ali, M. Hassan, S. Hossain, and Y.-D. Lin. Performance modeling and analysis of tcp connections over software defined networks. In *Proceeding of the 2017 IEEE Global Communications Conference*, pages 1–6, Singapore, 2017. IEEE.

[153] M. Lauridsen, R. Krigslund, M. Rohr, and G. Madueno. An empirical NB-IoT power consumption model for battery lifetime estimation. In *Proceeding of the 2018 IEEE 87th Vehicular Technology Conference (VTC)*, pages 1–5. IEEE, 2018.

[154] A. Lavric, A. I. Petrariu, and V. Popa. Long range sigfox communication protocol scalability analysis under large-scale, high-density conditions. *IEEE Access*, 7:35816–35825, 2019.

[155] S. K. Lee, M. Bae, and H. Kim. Future of iot networks: A survey. *Applied Sciences*, 7(10), 2017.

[156] T. Lei, Z. Yang, Z. Lin, and X. Zhang. State of art on energy management strategy for hybrid-powered unmanned aerial vehicle. *Chinese Journal of Aeronautics*, 32(6), 2018.

[157] H. Li, T. Huang, T. Yang, W. Li, and G. Zhang. A fast flow table engine for open vswitch with high performance on both lookups and updates. In *Proceedings of the ACM SIGCOMM 2019 Conference Posters and Demos*, pages 125–127, Beijing, China, 2019. ACM.

[158] H. Li and I. L.-J. Thng. Edge node buffer usage in optical burst switching networks. *Photon Network Commununication*, 13:31–51, 2007.

[159] G.-H. Liaw, S.-Y. Wang, T.-L. Kao, and T.-C. Chen. A novel packet aggregation mechanism for enhancing voip performance on IEEE 802.11 wireless mesh networks. *International Journal of New Technology and Research*, 6(7):247–264, 2009.

[160] C. Lin, K. Wang, and G. Deng. A qos-aware routing in sdn hybrid networks. *Procedia Computer Science*, 110:242–249, 2017.

[161] S.-C. Lin, I. F. Akyildiz, P. Wang, and M. Luo. Qos-aware adaptive routing in multi-layer hierarchical software defined networks: A reinforcement learning approach. In *Proceeding of the 2016 IEEE International Conference on Services Computing (SCC)*, pages 25–33, San Francisco, CA, 2016. IEEE.

[162] Y.-B. Lin, S.-Y. Wang, C.-C. Huang, and C.-M. Wu. The sdn approach for the aggregation/disaggregation of sensor data. *Sensors*, 18(2025):1–15, 2018.

[163] J. Liu and N. Ansari. The impact of the burst assembly interval on the obs ingress traffic characteristics and system performance. In *Proceedings of the 2004 IEEE International Conference on Communications (IEEE Cat. No.04CH37577)*, pages 1559–1563, Paris, France, 2004. IEEE.

[164] X. Liu, M. Zhao, A. Liu, and K. K. L. Wong. Adjusting forwarder nodes and duty cycle using packet aggregation routing for body sensor networks. *Information Fusion*, 53:183–195, 2020.

[165] Y. Lu. Diffusion approximation for random walk reflected away from boundary. *Mathematics*, August 5, 2003.

[166] X. Ma, Y. Duan, X. Meng, Q. Zhu, Z. Wang, and S. Zhu. Optimal configuration for photovoltaic storage system capacity in 5g base station microgrids. *Global Energy Interconnection*, 4(5):465–475, 2021.

[167] R. Maaloul, R. Taktak, L. Chaari, and B. Cousin. Energy-aware routing in carrier-grade ethernet using sdn approach. *IEEE Transactions on Green Communications and Networking*, 2(3):844–858, 2018.

[168] A. L. R. Madureira, F. R. C. Araújo, and L. N. Sampaio. On supporting iot data aggregation through programmable data planes. *Computer Networks*, 177(2025), 2020.

[169] K. Mahmood, A. Chilwan, O. Osterbo, and M. Jarschel. Modelling of openflow-based software-defined networks: the multiple node case. *The Institution of Engineering and Technology Journal*, 4(5):278–284, 2015.

[170] W. G. Marchal. An approximate formula for waiting time in single server queues. *AIIE Transactions*, 8(4):473–474, 1975.

[171] G. A. Marin, X. Mang, E. Gelenbe, and R. O. Onvural. Statistical call admission control. *IEEE communication Letters*, 222(824), 2001.

[172] G. A. Marin, X. Mang, E. Gelenbe, and R. O. Önvural. Statistical call admission control. *US Patent 6,222,824*, 2001.

[173] B. Martinez, M. Montón, and J. D. Prades. The power of models: Modeling power consumption for iot devices. *IEEE Sensors Journal*, 15(10):5777–5789, 2015.

[174] S. N. K. Marwat, Y. Mehmood, A. Khan, S. Ahmed, A. Hafeez, T. Kamal, and A. Khan. Method for handling massive IoT traffic in 5G networks. *Sensors*, 18(3966):1–16, 2018.

[175] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: Enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, 2008.

[176] T. Meng, X. Li, S. Zhang, and Y. Zhao. A hybrid secure scheme for wireless sensor networks against timing attacks using continuous-time markov chain and queueing model. *Sensors*, 16(1606):1–15, 2016.

[177] W. Miao, G. Min, Y. Wu, H. Wang, and J. Hu. Performance modelling and analysis of software defined networking under bursty multimedia traffic. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 12(55):24–36, 2018.

[178] F. Michelinakis, A. S. Al-selwi, M. Capuzzo, A. Zanella, K. Mahmood, and A. Elmokashfi. Dissecting energy consumption of NB-IoT devices empirically. *IEEE Internet of Things Journal*, 8(2):1224–1242, 2021.

[179] R. Min and A. Chandrakasan. Mobicom poster: top five myths about the energy consumption of wireless communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 7(1):65–67, 2003.

[180] V. Misra, W.-B. Gongnad, and D. Towsley. A fluid-based analysis of a network of aqm routers supporting tcp flows with an application to red. In *Proceeding of the Conference on Applications, Technologies, Architectures and Protocols for Computer Communication (SIGCOMM 2000)*, pages 152–160. ACM, 2000.

[181] X. Mountrouidou and H. Perros. On the departure process of burst aggregation algorithms in optical burst switching. *Advances in Electronics and Telecommunications*, 53(3):38–45, 2020.

[182] X. Mountrouidou and H. G. Perros. Characterization of the burst aggregation process in optical burst switching. In F. B. et al., editor, *Networking 2006: Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications Systems*, volume 3976, pages 752–764. Springer, Berlin, Heidelberg, 2006.

[183] C. B. Mwakwata, H. Malik, M. M. Alam, Y. L. Moullec, S. Parand, and S. Mumtaz. Narrowband internet of things (NB-IoT): From physical (phy) and media access control (mac) layers perspectives. *Sensors*, 19(2613):1–35, 2019.

[184] T. M. Nam, N. H. Thanh, N. Q. Thu, H. T. Hieu, and S. Covaci. Energy-aware routing based on power profile of devices in data center networks using sdn. In *Proceeding of the 2015 12th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, pages 1–6, Hua Hin, 2015. IEEE.

[185] E. Newton. How to optimise your iot device's power consumption, 2021.

[186] V.-L. Nguyen and R.-H. Hwang. Energy depletion attacks in low power wireless networks. *IEEE Access*, 7:51915–51932, 2019.

[187] A. Nieto and J. Lopez. Security and qos tradeoffs: Towards a fi perspective. In *Proceedings of the 2012 26th International Conference on Advanced Information Networking and Applications Workshops*, pages 745–750. IEEE, 2012.

[188] P. Nolan. Trends in the global ict industry–globalization competition and the internet of things. In M. A. Wang H., editor, *Consensus or Conflict? China and Globalization*, pages 243–252. Springer, Singapore, 2021.

[189] T. Nycz. *Diffusion approximation in the description of the dynamics and assessing the quality of Internet broadcasts*. PhD thesis, Silesian University of Technology, Gliwice, Poland, 2014.

[190] M. N. P. Pecka, S. Deorowicz. Pefficient representation of transition matrix in the markov process modeling of computer networks. In T. et al., editor, *Man-Machine Interactions 2, Advances in Intelligent and Soft Computing*, volume 103, pages 457–464. Springer, 2011.

[191] P. Parthasarathy and N.Selvaraju. Transient analysis of a queue where potential customers are discouraged by queue length. *Math. Probl. Eng.*, 7:433–454, 2001.

[192] P. Parthasarathy and R. Sudhesh. Exact transient solution of a discrete time queue with state-dependent rates. *Am. J. Math. Manag. Sci.*, 26:253–276, 2006.

[193] G. O. Pérez, J. A. Hernández, and D. L. López. Delay analysis of fronthaul traffic in 5g transport networks. In *Proceeding of the 2017 IEEE 17th International Conference on Ubiquitous Wireless Broadband (ICUWB)*. IEEE, 2017.

[194] B. Pernici, M. Aiello, J. Vom Brocke, B. Donnellan, E. Gelenbe, and M. Kretsis. What is can do for environmental sustainability: a report from caise'11 panel on green and sustainable is. *Communications of the Association for Information Systems*, 30(1):18, 2012.

[195] B. Pfaff, J. Pettit, T. Koponen, E. J. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar, K. Amidon, and M. Casado. The design and implementation of open vswitch. In *Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation*, pages 117–130. ACM, 2015.

[196] N. Piovesan, A. F. Gambin, M. Miozzo, and M. R. and. Energy sustainable paradigms and methods for future mobile networks: A survey. *Computer Communications*, 119:101–117, 2018.

[197] D. Potier. New user's introduction to qnap2, rapport technique, 1984.

[198] A. Proto and T. C. M. de Brito Carvalho. Applying distance metrics for anomaly detection of energy-based attacks in iot sensors. *Brazilian Journal of Development*, 6(11):92412–92435, 2020.

[199] C. Pu and B. Groves. Energy depletion attack in low power and lossy networks: Analysis and defenses. In *Proceeding of the IEEE 2019 2nd International Conference on Data Intelligence and Security (ICDIS*, pages 14–21, South Padre Island, TX, USA, 2019. IEEE.

[200] M. Raeis, A. Burchard, and J. Liebeherr. Analysis of a queueing model for energy storage systems with self-discharge. *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*, 5(3):1–26, 2020.

[201] S. Rao, D. Chendanda, C. Deshpande, and V. Lakkundi. Implementing LWM2M in constrained iot devices. In *Proceedings of the 2015 IEEE Conference on Wireless Sensors (ICWiSe)*, pages 52–57, Melaka, Malaysia, 2015. IEEE.

[202] M. Raval, S. Bhardwaj, A. Aravelli, J. Dofe, and H. G. and. Smart energy optimization for massive iot using artificial intelligence. *Internet of Things*, 13:100354, 2021.

[203] D. R. Raymond, R. C. Marchany, M. I. Brownfield, and S. F. Midkiff. Effects of denial-of-sleep attacks on wireless sensor network mac protocols. *IEEE Transactions on Vehicular Technology*, 58(1):367–380, 2009.

[204] P. Reinecke, T. Krauß, and K. Wolter. Hyperstar: Phase-type fitting made easy. In *Proceeding of the 9th International Conference on the Quantitative Evaluation of Systems (QEST 2012)*, pages 201–202, 2012.

[205] K. C. Reinhardt, T. R. Lamp, and J. W. Geis. Solar-powered unmanned aerial vehicles. In *Proceedings of the IECEC 96. Proceedings of the 31st Intersociety Energy Conversion Engineering Conference*, pages 41–46, Washington, DC, USA, 1996. IEEE.

[206] M. Reiser and H. Kobayashi. Accuracy of the diffusion approximation for some queuing systems. *IBM Journal of Research and Development*, 18(2):110–124, March 1974.

[207] A. Rezvanian, B. Moradabadi, M. Ghavipour, M. M. D. Khomami, and M. R. M. M.R, editors. *Introduction to Learning Automata Models*, volume 820, pages 1748–1748. Springer, Cham, 2019.

[208] P. Rygielski, M. Seliuchenko, S. Kounev, and M. Klymash. Performance analysis of sdn switches with hardware and software flow tables. In *Proceeding of the 10th EAI International Conference on Performance Evaluation Methodologies and Tools on 10th EAI International Conference on Performance Evaluation Methodologies and Tools*, pages 80–887. ACM, 2017.

[209] Y. M. Safdar Nawaz Khan Marwat, A. Khan, S. Ahmed, A. Hafeez, T. Kamal, and A. Khan. Method for handling massive iot traffic in 5G networks. *Sensors*, 18(3966):1–16, 2018.

[210] A. A. M. Saleh and J. M. Simmons. All-optical networking-evolution, benefits, challenges, and future vision. *Proceedings of the IEEE*, 100(5):1105–1117, 2012.

[211] T. Sanislav, G. D. Mois, S. Zeadally, and S. Folea. Energy harvesting techniques for internet of things (iot). *IEEE Access*, 9:39530–39549, 2021.

[212] D. Sattar and A. Matrawy. An empirical model of packet processing delay of the open vswitch. In *Proceedings of the 2017 IEEE 25th International Conference on Network Protocols (ICNP)*, pages 1–6. IEEE, 2017.

[213] V. Shakhov and I. Koo. Depletion-of-battery attack: Specificity, modelling and analysis. *Sensors*, 18(1849):1–20, 2018.

[214] V. Shakhov, I. Koo, and A. Rodionov. Energy exhaustion attacks in wireless networks. In *Proceeding of the 2017 International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON)*, pages 1–3. IEEE, 2017.

[215] P. Shakthipriya and A. R. Bevi. Network protocol-based qos routing using software defined networking. In S. S. Dash, K. Vijayakumar, B. K. Panigrahi, and S. Das, editors, *Artificial Intelligence and Evolutionary Computations in Engineering Systems*, pages 363–374, Singapore, 2017. Springer Singapore.

[216] A. Sharma, C. M. Basnayaka, and D. N. K. Jayakody. Communication and networking technologies for uavs: A survey. *Journal of Network and Computer Applications*, 168:102739, May 2020.

[217] O. P. Sharma and U. Gupta. Transient behaviour of an M/M/1/N Queue. *Stochastic Processing and Application*, 13:327–331, 1982.

[218] V. Sharma and R. Rajesh. Queuing theoretic and information theoretic capacity of energy harvesting sensor nodes. In *Proceedings of 2011 Conference Record of the Forty Fifth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, pages 383–388, Pacific Grove, CA, 2011. IEEE.

[219] M. Shirvanimoghaddam, K. Shirvanimoghaddam, M. M. Abolhasani, M. Farhangi, V. Z. Barsari, H. Liu, M. Dohler, and M. Naebe. Towards a green and self-powered internet of things using piezoelectric energy harvesting. *IEEE Access*, 7:94533–94556, 2019.

[220] R. Sidje. and W. J. Stewart. A numerical study of large sparse matrix exponentials arising in markov chains. *Computational Statistics & Data Analysis*, 29(012003):345–368, 1999.

[221] D. Singh, B. Ng, Y.-C. Lai, Y.-D. Lin, and W. K. Seah. Analytical modelling of software and hardware switches with internal buffer in software-defined networks. *Journal of Network and Computer Applications*, 136:22–37, 2019.

[222] D. Singh, B. Ng, Y.-C. Lai, Y.-D. Lin, and W. K. G. Seah. Modelling software-defined networking: Software and hardware switches. *Journal of Computer Network and Computer Applications*, 122:24–36, 2018.

[223] A. Sivanathan, H. H. Gharakheili, F. Loi, A. Radford, A. V. Chamith Wijenayake, and V. Sivaraman. Classifying iot devices in smart environments using network traffic characteristics. *IEEE Transactions on Mobile Computing*, 18(8):1745–1759, 2019.

[224] V. Sivaraman, A. Vishwanath, Z. Zhao, and C. Russell. Profiling per-packet and per-byte energy consumption in the NetFPGA gigabit router. In *Proceedings of the 2011 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 331–336, Shanghai, 2012. IEEE.

[225] N. Sklavos and P. Souras. Economic models and approaches in information security for computer networks. *International Journal of Network Security*, 2(1):14–20, 2006.

[226] R. Smith, D. Palin, P. P. Ioulianou, V. G. Vassilakis, and S. F. Shahandashti. Battery draining attacks against edge computing nodes in iot networks. *Cyber-Physical Systems*, pages 1–22, 2020.

[227] K. Sood, S. Yi, and Y. X. and. Performance analysis of software-defined network router using m/geo/1. *IEEE communication Letters*, 20(12):27–51, 2016.

[228] A. Sørensen, H. Wang, M. J. Remy, N. Kjettrup, R. B. Sørensen, J. J. Nielsen, P. Popovski, and G. C. M. no. A modelling and experimental framework for battery lifetime estimation in nb-iot and lte-m. *CoRR abs/2106.13286*, 2021.

[229] H. Stehfest. Numeric inversion of laplace transform. *Communication of ACM*, 13(1):47–49, 1970.

[230] R. Sudhesh. Transient analysis of a queue with system disasters and customer impatience. *Queueing Systems*, 66(1):95–105, 2010.

[231] K. G. Suila, T. Czachórski, and A. Rataj. A queueing model of the edge node in ip over all-optical networks. In P. G. et al., editor, *Proceedings of the Computer Network Conference 2018*, pages 258–271, Gliwice, Poland, 2018. Springer.

[232] W. sung Kim, H. Eom, and Y. Kwon. Optimal design of photovoltaic connected energy storagesystem using markov chain models. *MDPI*, 13(3837):1–16, 2021.

[233] M. N. Tadeusz Czachórski and T. Nycz. Modelling transient states in queueing models of computer networks: A few practical issues. In V. V. et al, editor, *Distributed Computer and Communication Networks. Communications in Computer and Information Science*, volume 279, pages 58–72. Springer International Publishing Switzerland, 2014.

[234] L. Takâcs. *Introduction to the Theory of Queues*. Oxford University Press, 1960.

[235] H. Takagi and A. M. Tarabia. *Explicit Probability Density Function for the Length of a Busy Period in an M/M/1/K Queue*. Springer, Spring Street, New York, NY 10013, USA, 2009.

[236] L. Tao, J. Ma, Y. Cheng, A. Noktehdan, and J. C. C. Lu. A review of stochastic battery models and health management. *Renewable and Sustainable Energy Reviews*, 80:716–732, 2017.

[237] S. R. Thakur and R. M. Goudar. Improving network I/O virtualization performance. *Internation Journal of Engineering Trends and technology (IJETT)*, 11(2):79–83, 2014.

[238] A. Thibbotuwawa, P. Nielsen, Z. Banaszak, and G. Bocewicz. Energy consumption in unmanned aerial vehicles: A review of energy consumption models and their relation to the uav routing. In *Proceedings of the ISAT 2018*, pages 173–184. Springer, 2019.

[239] M. A. Toksöz and N. Akar. Dynamic threshold-based assembly algorithms for optical burst switching networks subject to burst rate constraints. *Photon Network Commununication*, 20:120–130, 2010.

[240] F. Tonini, B. M. Khorsandi, S. Bjornstad, R. Veisllari, and C. Raffaelli. C-RAN traffic aggregation on latency-controlled ethernet links. *Applied Sciences*, 8(2279):1–12, 2018.

[241] R. Tucker and K. Hinton. Energy-efficient networking, 2014.

[242] C. Tunc and N. Akar. Markov fluid queue model of an energy harvesting IoT device with adaptive sensing. *Performance Evaluation*, 111:1–16, 2017.

[243] G. Vasconcelos, G. Carrijo, R. Miani, J. Souza, and V. Guizilini. Uavs path deviation attacks: Survey and research challenges. In *Proceeding of the 2020 IEEE International Conference on Sensing, Communication and Networking (SECON Workshops)*, pages 1–6. IEEE, 2010.

[244] S. Vinoski. Advanced message queuing protocol. *IEEE Internet Computing*, 10(6):87–89, 2006.

[245] A. Vishwanath, K. Hinton, R. W. A. Ayre, and R. S. Tucker. Modeling energy consumption in high-capacity routers and switches. *IEEE Journal on Selected Areas in Communications*, 32(8):1524–1532, 2014.

[246] C.-X. Wang, F. Haider, X. Gao, X.-H. You, Y. Yang, D. Yuan, H. M. Aggoune, H. Haas, and S. F. E. Hepsaydir. Cellular architecture and key technologies for 5g wireless communication networks. *IEEE Communications Magazine*, 52(2):122–130, 2014.

[247] H. Wang, H. Li, L. Ye, X. Chen, H. Tang, and S. Ci. Modeling, metrics, and optimal design for solar energy-powered base station system. *Journal on Wireless Communications and Networking*, 39:1–17, 2015.

[248] L. Wang and E. Gelenbe. Adaptive dispatching of tasks in the cloud. *IEEE Transactions on Cloud Computing*, 6(1):33–45, 2018.

[249] S.-Y. Wang, J.-Y. Li, and Y.-B. Lin. Aggregating and disaggregating packets with various sizes of payload in P4 switches at 100 gbps line rate. *Journal of Network and Computer Applications*, 165:1–15, 2020.

[250] S.-Y. Wang, C.-M. Wu, Y.-B. Lin, and C.-C. Huang. High-speed data-plane packet aggregation and disaggregation by p4 switches. *Journal of Network and Computer Applications*, 142:98–110, 2019.

[251] Y. Wang, C. Zhang, and Z. Chen. Model-based state-of-energy estimation of lithium-ion batteries in electric vehicles. *Energy Procedia*, 88:998–1004, 2016.

[252] S. Wijeratne, A. Ekanayake, S. Jayaweera, D. Ravishan, and A. Pasqual. Scalable high performance router architecture on fpga for core networks. In *Proceeding of the 2019 ACM/SIGDA International Symposium*. ACM, 2019.

[253] S. J. Wilton and N. P. Jouppi. Cacti: An enhanced cache access and cycle time model. *IEEE Journal of Solid-State Circuits*, 31(5):677–688, 1996.

[254] C. Woodside, B. Pagurek, and G. Newell. A diffusion approximation for correlation in queues. *Journal of Applied Probability*, 17(4):1033–1047, 1980.

[255] L. Wu, C. Wen, and H. Ren. Reliability evaluation of the solar power system based on the markov chain method. *Energy Research*, pages 1–8, 2017.

[256] L. X.CaI and et al. Dimensioning network deployment and resource management in green mesh networks. *IEEE Wireless Communications*, pages 58–65, 2011.

[257] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie. A survey on software-defined networking. *IEEE Communication Surveys & Tutorials*, 17(1):27–51, 2015.

[258] XIDASIOT. Iot's power problem.

[259] B. Xiong, K. Yang, J. Zhao, W. Li, and K. Li. Performance evaluation of openflow-based software-defined networks based on queueing model. *Computer Networks*, 102:172–185, 2016.

[260] H. Yamaki. Efficient cache architecture for table lookups in an internet router. *International Journal of Advanced Computer Science and Applications*, 11(5):665–672, 2020.

[261] Q. Zhang, N. Cui, Y. Shang, B. Duan, and C. Zhang. An improved peukert battery model of nonlinear capacity considering temperature effect. In *Proceeding of the 5th IFAC Conference on Engine and Powertrain Control, Simulation and Modeling E-COSM*, pages 665–669, Changchun, China, 2018. Elsevier.

[262] K. Zheng, C. Hu, H. Lu, and B. Liu. A tcam-based distributed parallel ip lookup scheme and performance analysis. *IEEE/ACM Transactions on Networking*, 14(4):863–875, 2006.