Performance Evaluation by Aggregation of ON-OFF Traffic in Processor Sharing and Related Systems

By

Vindya Amaradasa

SUBMITTED FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

Supervised by Dr. John A. Schormans

Department of Electronic Engineering
Queen Mary
University of London

August 2008

DECLARATION

I hereby declare that the work presented in this thesis is solely my work and that to the best of my knowledge the work is original except where indicated by reference to respective authors.

………………………
Vindya Amaradasa

Date:

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my supervisor, Dr. John Schormans, for his expert guidance, support and encouragement given to me and always being there for me throughout the course of this research. Many thanks go to Professor Jonathan Pitts and Dr. Chris Phillips for the valuable discussions which greatly contributed to my progress. I am also grateful to Dr. Raul Mondragon for the expert Maths help and to Professor Laurie Cuthbert for the continuous support and encouragement.

My appreciation goes to EPSRC for the financial funding provided. I would also like to thank the numerous staff and students in the Dept. of Electronic Engineering, specially Lynda Rolfe, Melissa Yeo, Marie Campbell, Kok Ho Huen and the IT team, Phil Wilson, Sharon Cording for their support and Maheen, Sharifah, Athen, Jimmy, Costa, Janny, Akram, Victor, Touseef, Ali, Matt, Oliver, Lina, Tony, Ling, Rupert and everyone else at the Thursday Lunch Club, aka TLC:) who have made this experience a joy. Special thanks go to Exeter Gradschool Team 'W-ers' and Shona and Anushini who, despite the distances and differences in fields, have constantly been in touch with me sharing the ups and downs of PhD life to-date. I also want to thank all my other friends and relatives for their understanding and well-wishes.

I feel a deep sense of gratitude to all my teachers I have met during the past 21 years of school and university life, who are great contributors to what I am today. Above all, my heartfelt appreciation goes to my ammi Badra, thaththi Ama and aiya Lashan who have supported me through all those years, always giving priority to my work and endless deadlines; specially my parents who I am deeply indebted to. Many thanks go to Suchy akki for making me take things easy and Deva and Leslie uncle for the lovely chats and patience. And last but not least, a big thank-you goes to my loving fiancé Isuru, without whom this thesis would have not been possible.

ABSTRACT

Even though networks such as the Internet traditionally provided a simple best-effort service, there is a growing need for networks with hard guarantees. However, currently researchers are finding it challenging to keep up with these requirements mainly due to lack of tools that can facilitate the modelling and analysis of large-scale packet networks. Motivated by the requirement for such tools, this research develops novel packet traffic aggregation techniques that enable the performance evaluation of queueing systems via Accelerated Simulation (AS) models: these can provide the required results many times faster than standard simulation models by reduction of events and therefore facilitate the simulation of larger networks with fewer resources.

Aggregation techniques are developed in the presence of bursty traffic modelled by ON-OFF sources, as this type of traffic is commonly observed in packet networks. Further, the primary scheduling discipline considered is Generalised Processor Sharing (GPS) which is the key aspect of novelty in this research as previous aggregation techniques have only dealt with FIFO traffic (and their variations). Non-FIFO schedulers such as GPS are increasingly important, mainly due to the emergence of multi-service queues in converged IP networks (e.g. VoIP). The developed techniques are also extended to be applicable to the Weighted Fair Queueing (WFQ) scheduling scheme, which is the most widely used practical approximation to the ideal GPS discipline.

Validation procedures are carried out for the novel aggregation techniques developed in this research by comparison with standard, non-accelerated simulations, which are themselves validated against theoretical models where available.

In the course of this research, an interesting discovery was made with regard to scheduler behaviour under congestion (i.e. when very large queues are present in the buffer): the tested schedulers, which are variants of Processor Sharing (PS) and FIFO, tend to exhibit delay performance that converge in this limit.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

AQM	Active Queue Management
AS	Accelerated Simulation
ATM	Asynchronous Transfer Mode
BSDRS	Burst Scale Decay Rate of System time
BSDRX	Burst Scale Decay Rate of X
CI	Confidence Interval
CoS	Class of Service
CPU	Central Processing Unit
DC	Deficit Counter
DPS	Discriminatory Processor Sharing
DRR	Deficit Round Robin
fBM	fractional Brownian Motion
FIFO	First In First Out
GoS	Grade of Service
GPS	Generalised Processor Sharing
IID	Independent and Identically Distributed
IP	Internet Protocol
IS	Importance Sampling
LRD	Long Range Dependent
MANET	Mobile Ad Hoc NETworks
NS2	Network Simulator 2
PADS	Parallel And Distributed Simulation
PEBS	Probability of Entering Burst Scale
PGPS	Packet-by-packet GPS
PIRS	Parallel Independent Replicated Simulations
PQ	Priority Queueing
PS	Processor Sharing
PSDRS	Packet Scale Decay Rate of System time
PSDRX	Packet Scale Decay Rate of X
RESTART	REpetitive Simulation Trials After Reaching Thresholds

RNG	Random Number Generator
RR	Round Robin
SCFQ	Self-Clocked Fair Queueing
SFQ	Start-time Fair Queueing
SSIM	Session-level SIMulator
ST	System Time
TCL	Tool Command Language
TSFO	Time-Stepped Fluid Oriented
TU	Time Unit
VoIP	Voice over Internet Protocol
VT	Virtual Time
WFQ	Weighted Fair Queueing
WRR	Weighted Round Robin
WT	Waiting Time

LIST OF MATHEMATICAL SYMBOLS

a	PEBS given by the best-fit
$a[k]$	Pr{k arrivals in a Time Unit}
α	Significance level (for Confidence Intervals)
B_j	Set of backlogged classes in the time interval (t_{j-1}, t_j)
c_1	Lower bound of Confidence Interval
c_2	Upper bound of Confidence Interval
C_x	Overall service rate received by class x
C	Total service rate
d^2	Variance of x'
D'	Steady-state mean delay
D_i	Delay of the ith packet
e^b	BSDRX given by the best-fit
δ	Service time
$F(j)$	Pr{j queues are non-empty}
$f(t)$	Pr{$t \leq ST < (t+1)$}
h	Arrival rate of an ON-OFF source during the ON period
H_o	Number of events in the original model
H_A	Number of events in the AS model
K_x	Number of traffic flows in class x
λ_x	Mean arrival rate of class x
λ	Mean arrival rate
m	Number of replications of a simulation
μ	Mean value of a property in the real system
n_0	Length of transient phase
n	Length of simulation excluding transient phase
N	Number of classes
ξ	Speed-up factor

ρ_x	Load on the buffer of class x
ρ	Load on the buffer
$p(x)$	Pr{ x packets in the system}
P_B	Probability of experiencing burst scale X
P_{BS}	Probability of experiencing burst scale ST
$P_{overflow}$	Buffer overflow probability
q_x	Pr{class x buffer is empty}
r	Packet Scale Decay Rate of X (PSDRX)
R	Burst Scale Decay Rate of X (BSDRX)
$s[k]$	Pr{k packets in the system} in a single class system
s^2	Variance of $\overline{\overline{x}}$
T_{on}	Mean ON time of an ON-OFF source
T_{off}	Mean OFF time of an ON-OFF source
ϕ_x	Weight factor allocated to class x
ψ	Activity factor of an ON-OFF source
v	Packet Scale Decay Rate of System time (PSDRS)
V	Burst Scale Decay Rate of System time (BSDRS)
X	Number in the system
X_0	Number in the system at time '0'
x_k	Number in the system at knee-point
x_{ij}	jth observation of the ith replication of a simulation
$\overline{x_i}$	Mean produced by the ith replication of a simulation
$\overline{\overline{x}}$	Overall mean of all replications of a simulation
x'	(True) mean value of a property in the simulation model

CHAPTER 1

INTRODUCTION

1.1 Motivation

Performance evaluation of telecommunication networks is vital for service providers. The ability to evaluate the performance and introduce improvements in a timely manner is the key to providing positive customer experience. Hence, considerable research has gone into improving the efficacy of existing performance evaluation techniques, and developing new ones.

One of the most common methods employed for the performance evaluation of a network is simulation, where a computer program is used to imitate the operations of a real-world system. The main problem with this approach is the large amount of computer resources i.e. time and memory required by such programs; which is worsened as networks grow larger and more complex (e.g. to include CoS, AQM mechanisms). This is because, the higher the number of events that need to be simulated in order to obtain the desired result (e.g. estimate the probability of rare events), the greater the amount of computer resources required. Also, the increasing demand for better services requires Research and Development teams to be able to analyse networks and introduce new developments at an increasing pace. For example, the ongoing transition from circuit-switched telephone networks to packet-switched Voice-over-IP (VoIP) is a good indicator of how much the population at large is relying on packet networks such as the Internet. These are the reasons why acceleration (or speeding up) of simulations is rapidly becoming important, as a means of enabling researchers to obtain required results faster than with ordinary simulations.

Existing Accelerated Simulation (AS) techniques have achieved reasonable levels of speed up (specific details are given in Chapter 2); however, they are only applicable to First In First Out (FIFO) scheduling disciplines (and their variations). Due to the need for fair bandwidth allocation and the provision of differentiated services on the Internet, alternative scheduling mechanisms such as Generalised Processor Sharing (GPS) are currently attracting interest over FIFO. In addition to scheduling at the outgoing links, the routeing processors also involve PS scheduling: this is because most modern micro-processors serve oncoming requests in a processor-sharing manner. These are the motivations behind researching on AS models for processor-sharing disciplines. Accordingly, it will be possible to apply the results presented in this thesis for the performance evaluation of modern networks, enabling researchers to move a step closer to keeping up with the ongoing demand for improvements.

1.2 Objectives

The main objective of this research is to develop novel traffic aggregation techniques fundamentally with a view to performance evaluation by Accelerated Simulation (AS). Aggregation is defined as reducing the number of traffic flows, e.g. from *N* to *1*, while ensuring that the 'aggregated' model has buffer performance equivalent to that of the original model.

The aggregation techniques developed in this research are applicable in the presence of bursty traffic scheduled by processor-sharing disciplines such as Generalised Processor Sharing (GPS), also their practical equivalents such as Weighted Fair Queueing (WFQ). This thesis describes how each of the novel aggregation techniques can be used to develop Accelerated Simulation (AS) models, i.e. speeding up simulations by reduction of events which is made possible by the aggregation process.

The specific objectives are to:

1. Evaluate via novel traffic aggregation techniques,
    a. The queue state of a single flow and 'System Time' (denote by ST) in a queueing system fed by multiplexed homogeneous Poisson traffic sources scheduled by GPS.
    b. 'Number in the system' (denote by X) and ST in a queueing system fed by multiplexed homogeneous ON-OFF traffic sources (i.e. bursty traffic) scheduled by GPS.
    c. X and ST in a queueing system fed by multiplexed heterogeneous ON-OFF traffic sources scheduled by WFQ.

2. Demonstrate the accuracy of the aggregation techniques by comparison with standard simulations (i.e. non-accelerated), which are first validated against theoretical models where such models are available.

3. Describe how the aggregation techniques provide the base for AS models, giving an indication of the magnitude of speed-up they can achieve, when compared to equivalent ordinary simulations.

1.3 Novelty and contribution

The novelty of the research presented in this thesis mainly comes from the applicability of the traffic aggregation techniques to the GPS (and related) disciplines, since all aggregation techniques that exist in the literature (so far) are only applicable to the FIFO discipline (and its variations). The specific contributions are:

1. Novel aggregation techniques to evaluate the performance (e.g. 'number in the system' and 'system time') by Accelerated Simulation (AS) of a buffer fed by bursty traffic scheduled by both GPS and WFQ, are developed. Further, for WFQ (which is the most widely used practical approximation of GPS), the novel techniques are extended in a novel manner, such that they are applicable in the presence of heterogeneous traffic classes.

The substantial event reduction made possible by the aggregation techniques can be exploited in developing AS models, i.e. speeding up simulations to ensure required results are obtained within reasonable time limits. In addition, these techniques can also be used for the simplification of analytical models employed for performance evaluation.

2. A novel approximation of the GPS scheduler is proposed, implemented and validated. This implementation is based on a simple concept and proves to be an excellent means of studying the ideal GPS scheme, as shown by the validations presented in the thesis.

1.4 Thesis outline

Chapter 2 reviews techniques of network performance evaluation with the focus on 'simulation' which is the most common one. This chapter also explains how to ensure the credibility of simulation results, with references to how it was done throughout this research. Chapter 2 then explains the need to develop AS models and presents a review of existing AS techniques. Finally, Network Simulator 2 (NS2) which is the simulation tool used in this research, is introduced.

Chapter 3 discusses the problems caused by employing FIFO-scheduling in packet networks such as the vulnerability to ill-behaved sources. Then, the ideal concept of Generalised Processor Sharing (GPS) which is the solution to the problems of FIFO (and the main scheduling discipline used in this research), is introduced. This chapter also discusses the most commonly used practical approximations to GPS along with their advantages and disadvantages, and proposes a novel approximation which was made use of in this research.

Chapter 4 evaluates the performance of a queueing system (similar to the one at the output port of a router in a packet network) fed by homogeneous Poisson traffic flows, scheduled by the GPS discipline. Novel traffic aggregation techniques to evaluate the queue state and System Time (ST) of a single flow are developed, which lead to Accelerated Simulation (AS) models which can substantially reduce the number of events that need to be simulated. The novel aggregation techniques are validated by comparison with standard (non-accelerated) simulations. Note that techniques developed in this chapter (i.e. for Poisson traffic) cover the evaluation of the packet-scale component of bursty traffic which is the focus of this research, considered in the next chapter.

Chapter 5 concentrates on the burst-scale behaviour (which is the major component) of ‘bursty’ traffic, modelled by multiplexed ON-OFF sources. The burst-scale behaviour of homogeneous traffic flows sharing an output port scheduled by the GPS discipline is analysed in comparison with FIFO equivalents, which provides the base for the aggregation techniques. The ‘Number in the system’ (X) and ST are evaluated via aggregation techniques, where ST is the key performance metric. The novel techniques are validated by comparison with simulations. This chapter then describes the substantial event reduction that can be achieved by applying these aggregation techniques to Accelerated Simulation (AS).

Chapter 6 provides extensions to the novel aggregation techniques developed (in the previous Chapters) in two ways. They are the use of a Weighted Fair Queueing (WFQ) scheduler (instead of GPS) and heterogeneous traffic classes (instead of homogeneous ones); both extensions greatly enhance the ease of applicability of these techniques in practical networks. Review of the literature shows that performance evaluation in the presence of heterogeneous classes has been a challenge. In this chapter, novel aggregation techniques are developed to evaluate ST of a class of interest, which are again validated against simulations. Similar to the previous chapters, the significance of applying these techniques to speed-up simulations, is explained.

Chapter 7 describes some of the major extensions that result from the work presented in this thesis which is a remarkable first step in the performance evaluation of packet networks with non-FIFO schedulers. This chapter also presents the final conclusions drawn from this research.

CHAPTER 2

SIMULATION AND ACCELERATION

2.1 Introduction

This chapter first reviews network performance evaluation techniques (section 2.2), then details the 'simulation' technique which is the most commonly used technique, in section 2.3. This section also explains how to create and make use of valid simulation models. Section 2.4 discusses the need for Accelerated Simulation (AS) and presents a review of existing AS techniques, as the novel aggregation techniques developed in this thesis are aimed to be used in AS models. Finally in section 2.5, Network Simulator 2 (NS2), which is the simulation tool used throughout this research, is introduced.

2.2 Network Performance Evaluation

Performance evaluation is a constant requirement in the development of communication networks. The specific measures of interest may depend on factors such as the type of the network being considered and its applications. For example, important performance measures for circuit-switched networks are probability of blocked calls, Grade of Service (GoS) etc. while examples of performance measures specifically used in packet-switched networks are packet-delay, delay variation, loss probability and throughput.

The three broad techniques used for network performance evaluation are measurement, analytical modelling and simulation. The selection is made based on various considerations such as the life-cycle stage in which the system is (e.g. does the system exists at all), the time and resources available and the level of detail needed [1].

Measurement on a real system provides the most direct means of network performance evaluation. Although this method is practised by many network vendors, it is often prone to errors. A series of recent papers by Schormans et al ([2-4]), also ref. [5] have explored the inaccuracy inherent in packet level measurement, which are caused mainly by inappropriate probing patterns and rates. It has been discovered that even for static traffic in simple buffering scenarios there are practical load limits beyond which measurement accuracy degrades very rapidly. Additionally, it would not normally be accurate to draw general conclusions from measurement results, as many of the environmental parameters (e.g. system configuration, time of measurement etc.) may be unique to the experiment and may not represent the range of variables present in the real world. Moreover, measurement is expensive and cannot be done until the real system is built, unlike the other two approaches.

Analytical models can generally be solved rather quickly; however, a tractable analytical model often restricts the range of system characteristics that can be explicitly considered. Nonetheless, they can be effective when carefully applied. For example, this method can be considered as the best approach to determine the effects of various parameters and their interactions (provided a valid analytical model is available). Analytical modelling is usually the cheapest and fastest of the three techniques.

Simulation techniques (i.e. using a computer to model the operations of a real world system) can model a network to an arbitrary degree of detail, and may therefore be closer to reality than analytical models. Even though the solution of a simulation model requires significantly more computer resources than an analytical model, in some cases, simulation is the only viable modelling approach and is the most common approach to the evaluation of network performance [1, 6].

In some cases, both analytical and simulation models may be used. For example, simulation can be used to check the impact of the assumptions needed in an analytical model, while an analytical model can suggest appropriate parameters to investigate in a simulation study [7]. Further, combinations of simulation and analytical techniques have recently been used in developing AS models, for which this research is an example (see section 2.4.2.2 for more examples). Additionally, the three techniques of performance evaluation are often used as a form of validation[1] for one another; for instance, in this thesis, known standard analytical solutions were used as a form of validation for simulations (e.g. packet-scale queue state), while novel analytical solutions were validated by comparison with simulations (e.g. burst-scale system time).

2.3 Simulation

2.3.1 Introduction

This section first defines a few terms which are frequently used in simulation modelling, in section 2.3.2, while the types of models are identified in section 2.3.3. Then, section 2.3.4 discusses an extremely important topic, i.e. ensuring the credibility of simulation models, which is often given inadequate consideration by researchers. This is followed by a short description of the simulation clock in section 2.3.5.

2.3.2 Definitions

The most basic definition is a 'system', which can be defined as the collection of hardware, software and firmware components [1]. The 'state' of a system is the collection of variable values necessary to describe the system at a particular time; and those variables are called 'state-variables'. Finally, an 'event' is defined as an instantaneous occurrence that may change the state of the system [7].

For example, define the output port of a router (with a single server and single buffer) as the 'system'. Examples of 'state-variables' are 'number in the system',

[1] See section 2.3.4.1 for the definition of 'validation' and how it is addressed in this thesis

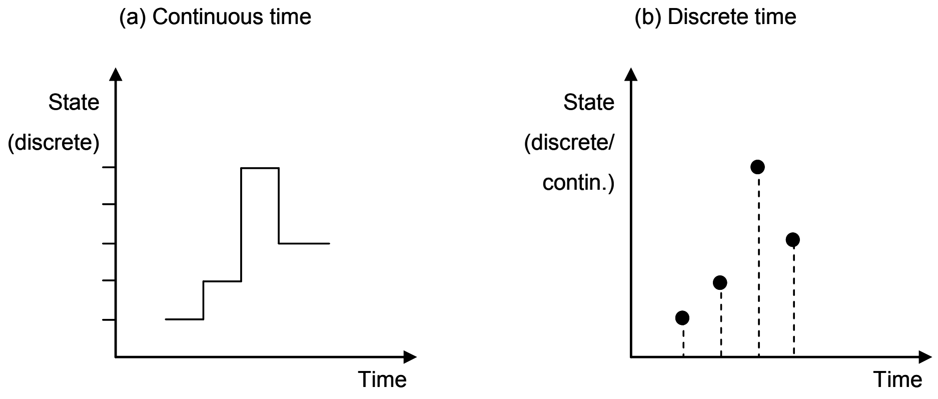
'number in the buffer' and 'waiting time (of an arbitrary arrival)', etc. The 'state' of this system at time *t* could be *k* jobs in the system and t_w expected waiting time etc. Examples of 'events' that would change the state of the system are the arrival of a job and the departure of a job, etc.

'Simulation' is done by designing a 'model' of the real system, which usually contains only those elements of the real system which are necessary for the purpose of the study. For instance in the above example of the router, the model defined for a queueing simulation study would only contain the buffer and the server (i.e. the LAN interfaces and any other hardware details of the router are left out). All the models employed in the experiments of this research are described in the relevant chapters (Chapter 4-6).

2.3.3 Types of models

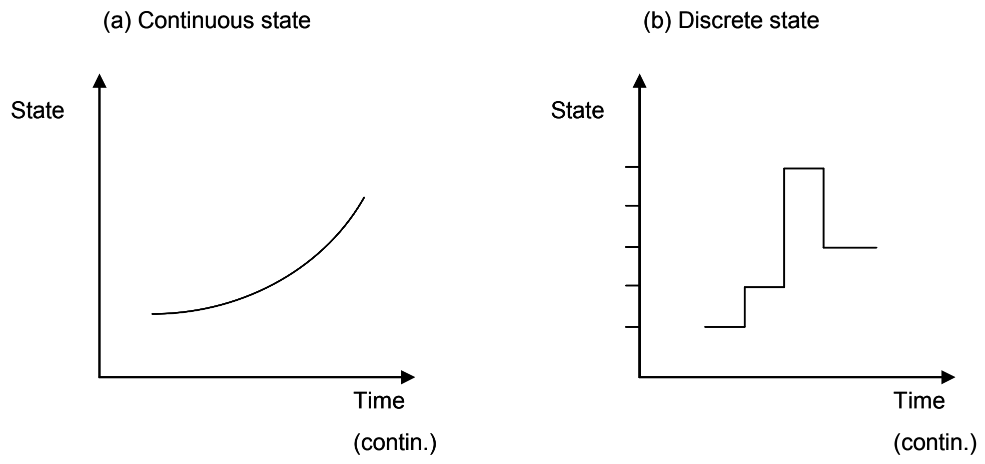
It is useful to identify the characteristics of a model prior to considering simulation. These characteristics fall under the following categories:

- Continuous-time Vs discrete-time models:



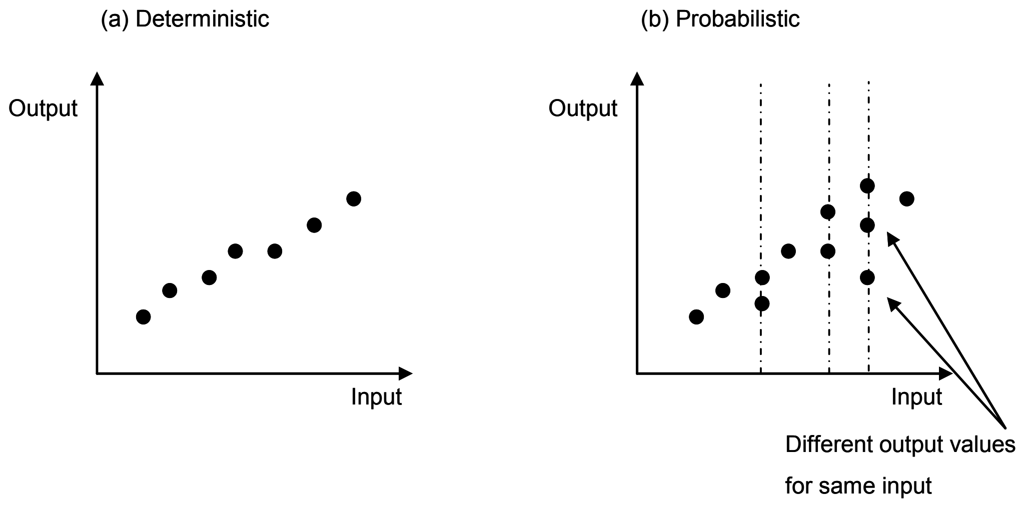
Continuous-time models are where the system state is defined at all times, while discrete-time models are where the system state is defined only at particular instants in time (the state can either be continuous or discrete – see next category).

- Continuous-state Vs discrete-state models:



A model is said to be of continuous-state or discrete-state depending on whether the state-variables are continuous, i.e. they can take an uncountable number of values (e.g. time spent in the system), or discrete, i.e. they can only take certain values (e.g. number of jobs in the system can only take integer values). Further, continuous or discrete states can be measured in either continuous or discrete time.

- Deterministic Vs probabilistic models:



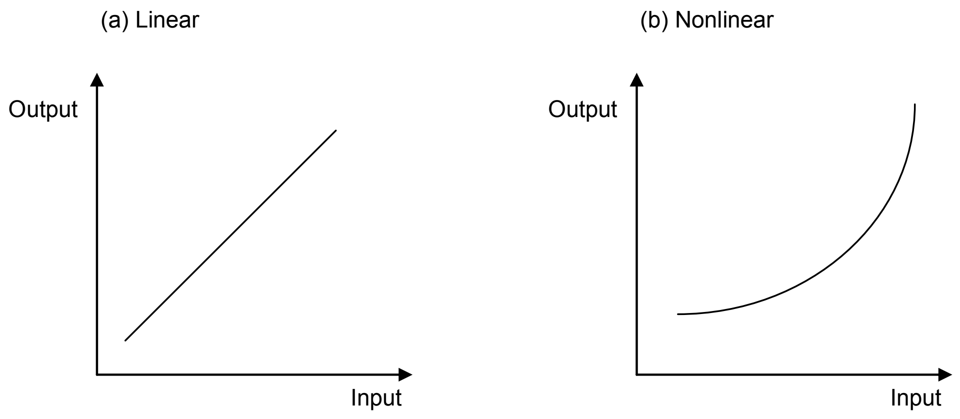
In a deterministic model, the output can be predicted with certainty (i.e. all repetitions with the same input parameters produce the same output), while in a probabilistic (also called stochastic) model, different results may be produced on repetitions for the same set of input parameters (with varied seeds in the Random Number Generator (RNG)[1]).

[1] See section 2.3.4.2 for a description of RNG.

- Static Vs dynamic models:

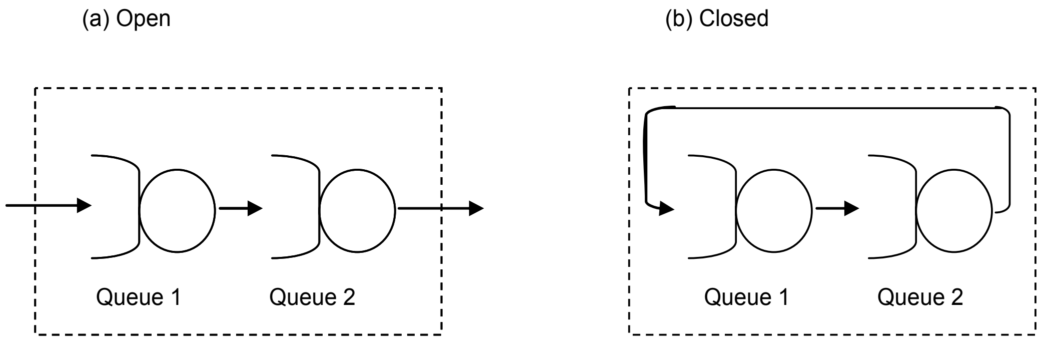
A model in which time is not a variable is called static. If the system state changes with time, the model is called dynamic.

- Linear Vs nonlinear models:



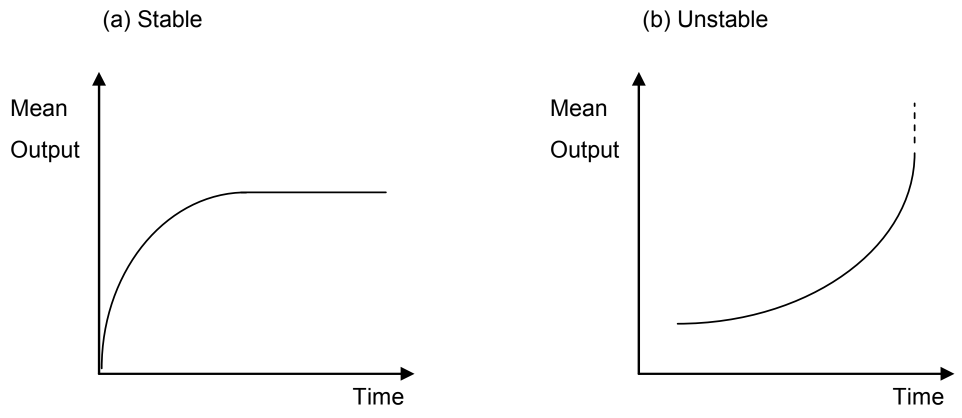
If the output parameters are a linear function of the input parameter, the model is linear; otherwise it is nonlinear.

- Open Vs closed models:



An open model is where the input is external to the model and independent of it, while a closed model has no external input.

- Stable Vs unstable models:



If the dynamic behaviour of the model settles down to a steady state, it is called a stable model. On the other hand if the behaviour is continuously changing, the model is said to be unstable.

Models of communication networks are typically continuous-time, discrete-state, stochastic, dynamic, nonlinear, open and stable models.

2.3.4 Credibility of simulation results

Once the required type is identified, the model can now be built and used in simulation experiments and results can be obtained. There are two main factors that must be considered in this process in order to ensure the credibility of produced results [8]. They are;

1. Verification and validation of simulation model
2. Use of model in a valid simulation experiment

The above factors are discussed in the order shown in Figure 2-1.

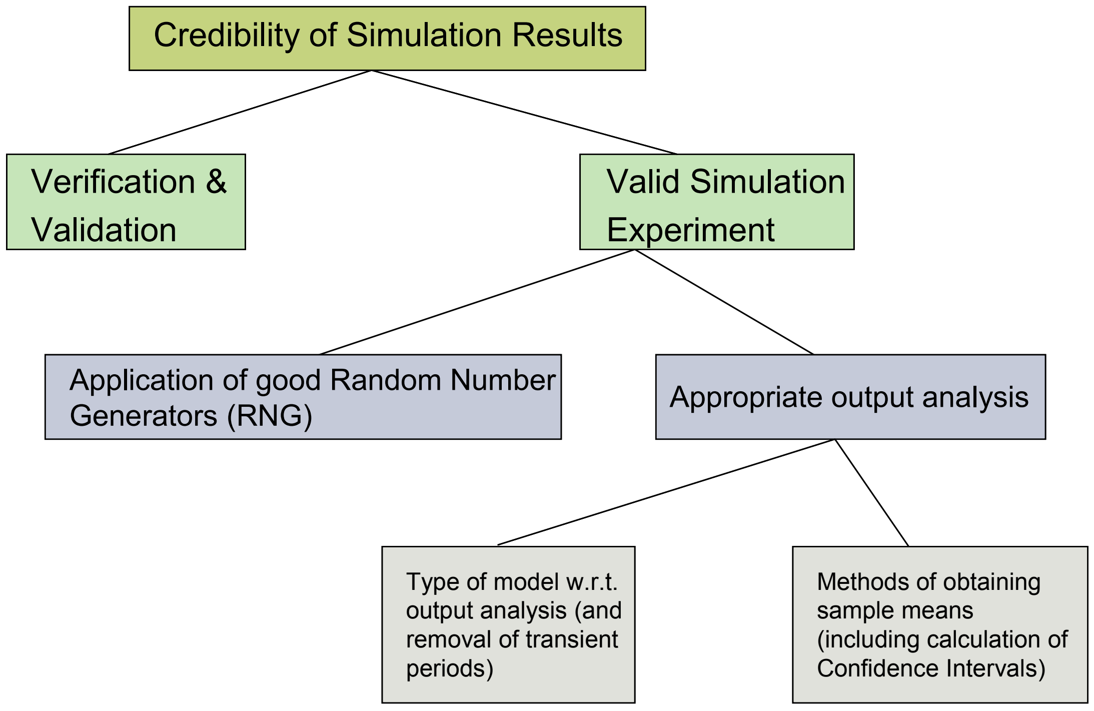

Credibility of Simulation Results
Verification & Validation
Valid Simulation Experiment
Application of good Random Number Generators (RNG)
Appropriate output analysis
Type of model w.r.t. output analysis (and removal of transient periods)
Methods of obtaining sample means (including calculation of Confidence Intervals)

Figure 2-1 Ensuring credibility of simulation results

2.3.4.1 <u>Verification and validation</u>

Verification refers to determining whether the proposed simulation model has been correctly translated into a computer program. Validation is concerned with determining if the simulation model is a correct representation of the actual system for the particular objectives of the study. Note that a model that is valid for one purpose may not be valid for another [7]. Therefore, validation techniques may differ from one simulation to the other; however, the same verification techniques could normally be applied in most cases [1].

The following verification techniques which are given in ref. [1] were applied to the simulation models used in this research:

- Anti-bugging (having additional checks and outputs in the program, e.g. packet drops when drops are not expected)
- Structured walk-through (explaining the code to another person)
- Simplified test cases (e.g. running with only one traffic source)
- Trace (print out a time-ordered list of events, e.g. time of entering and leaving the queue for each packet, which is checked through manually)

- Continuity test (running the simulation several times with slightly different values of input parameters which should produce only slight changes in the output, e.g. increase the load by 5%)
- Seed independence (checking if similar results are produced for different seed values in the RNG)

As mentioned, the exact techniques of validation may depend on the particular study; however, in all cases, there are three main aspects of the model that need to be validated; they are: assumptions, input values and output values. Each of these aspects may be validated by comparison with one or more of the following: expert intuition, real system measurements and theoretical tests, depending on which ones are possible and feasible [1].

In this research, the assumptions (e.g. use of a buffer that can hold an unlimited number of packets) were validated by expert intuition; input values were validated by expert intuition and by keeping to standard values as much as possible (e.g. standard VoIP talkspurt and silence periods); finally, output values were validated by comparison against theory where possible (e.g. packet-scale queue state distributions) and by expert intuition otherwise. Real system measurements were not available.

2.3.4.2 Use of model in a valid simulation experiment

Ensuring the validity of a simulation experiment involves two main aspects, which are, the application of appropriate Random Number Generators (RNG) and the appropriate analysis of simulation output data. RNGs are discussed first.

2.3.4.2.1 Random Number Generators (RNG)

As with the simulation of any system with random components, the simulation experiments of queueing models described in this thesis involve a number of random variables, e.g. packet inter-arrival times, burst times, idle times etc. This requires generating random values that appear to be "drawn" from specified distributions such as exponential, Pareto etc. Law and Kelton [7] refer to this process as ‘generating random variates’, which is defined as obtaining observations on a random variable, from the desired distribution. Further, random variates generated from the uniform distribution [0,1] are called ‘random numbers’. Random variates of all other distributions are obtained by transforming random numbers in a way determined by the desired distribution [7]. Therefore, RNGs play a vital role in all (stochastic) simulation experiments.

RNGs essentially consist of mathematical formulae by which ‘random’ numbers are generated. The prefix ‘pseudo’ is often added to random (i.e. pseudo-random) to indicate that the values are computed, not observed with some real-world chance mechanism. Moreover, it should be noted that ‘random’ here does not imply the values are unpredictable [9].

An RNG can be thought of as a long sequence or stream of numbers, where producing a random number corresponds to retrieving the next number from that sequence. If this sequence comes to an end before the end of the simulation, the RNG will repeat the sequence, which may result in undesired correlation. Therefore, the maximum number of values that can be generated by an RNG before it starts repeating itself, called the ‘period’, is an extremely important characteristic. The starting point of the sequence can normally be chosen by the user: this starting value is called the ‘seed’. Different seeds can be chosen for the different random processes which may exist in the simulation (called multiple processes), e.g. packet arrivals and service times[1].

The RNG employed in this research is the combined multiple recursive generator called MRG32k3a proposed by L'Ecuyer [10] and implemented in NS2 (from version ns-2.1b9 onwards) by Weigle in April 2002 [11, 12]. MRG32k3a is known to have satisfactory uniformity in the random numbers generated [8, 13]. It is also fast, in comparison to other RNGs [13]. The period is 3.1×10^{57} which can provide ample values (without repeating sequences) for the simulation of systems requiring multiple processes [14]. See Appendix A for a detailed explanation on how

[1] ‘Service-time’ is the time it takes to serve a packet.

MRG32k3a provides multiple independent processes across multiple runs of simulations.

In NS2, it is not normally necessary to explicitly set the seeds as this is done automatically. However, if required, the seed can also be set by the user: this provides reproducibility, one of the most important properties of a ‘good’ RNG [7]. The ability to reproduce an exact copy of a given stream of random numbers is essential when carrying out certain comparisons precisely, e.g. comparing the operation of two scheduling disciplines for the exact same process of arrivals (e.g. see simulation results of WRR and WFQ in section 6.2). Reproducibility also helps the verification process [15], e.g. in tracing.

2.3.4.2.2 Appropriate analysis of output data

The primary factor that determines the type of analysis process is the type of simulation experiment with regard to output analysis, which is considered first, which includes a discussion on transient periods. Thereafter, methods of obtaining sample means from raw data are discussed.

Types of simulations:

There are two types of simulations with regard to output analysis. It is important to identify which one is the most appropriate for the study in concern as certain aspects of setting up the simulation depends on the specific category [7]. They are terminating and non-terminating simulations.

A terminating simulation is one for which there is a specific “final” condition which could either be the system reaching a particular state (e.g. an empty queue) or a time point beyond which no useful information could be obtained (e.g. end of busy hour or a system that shuts down at 5pm everyday). In a terminating simulation, the initial conditions generally affect the desired measures; therefore, it is important to ensure that these conditions are representative of those of the actual system [7, 16].

A non-terminating simulation is one for which there is no specific final condition. Measurements are taken in the “long run” or when the system reaches “steady-state"[1], e.g. steady-state buffer occupancy of a queue fed by a continuous flow of traffic. Steady-state is theoretically reachable by a system after an infinitely long period. However, simulations must be executed and measurements taken within a finite period of time, which means obtaining accurate measurements in a non-terminating simulation is challenging [8].

All simulations presented in this thesis belong to the non-terminating category. This is because there is no specific final condition; therefore, it is the steady-state distributions of performance measures such as queue state, system time etc. that should be compared with the theoretical or analytical equivalents.

An important issue that must be taken into account with non-terminating simulations is the initial transient (also called "warm-up" or "burn-in") period, which refers to the period during which any observations made of the system are considerably affected by its initial state, i.e. biased. See Figure 2-2 for an example where D_i denotes the delay experienced by the ith packet entering an M/M/1 queue, for various initial states (i.e. 'number in the system at time 0', denoted by X_0). $E(D_i)$ is the 'transient' mean delay while D' is the 'steady-state' mean delay. Notice that $E(D_i)$ during the so called transient period is significantly affected by the corresponding initial state of the system. $E(D_i)$ eventually converges to D', which is the value that will be of interest.

Note that while the steady-state distribution of the target variable does not depend on the initial state, the rate of convergence of the transient distribution to the steady-state distribution, does. This is observable in Figure 2-2 where e.g. the X_0=15 case converges much faster than the X_0=0 does. Also note that steady-state does not imply that the random variables D_i, D_{i+1} will all take the same value in a particular simulation run; rather, it means that they will all have approximately the same distribution across multiple runs, irrespective of the initial state (unlike in the transient period) [7].

[1] Non-terminating simulations are often referred to as 'steady-state simulations'.

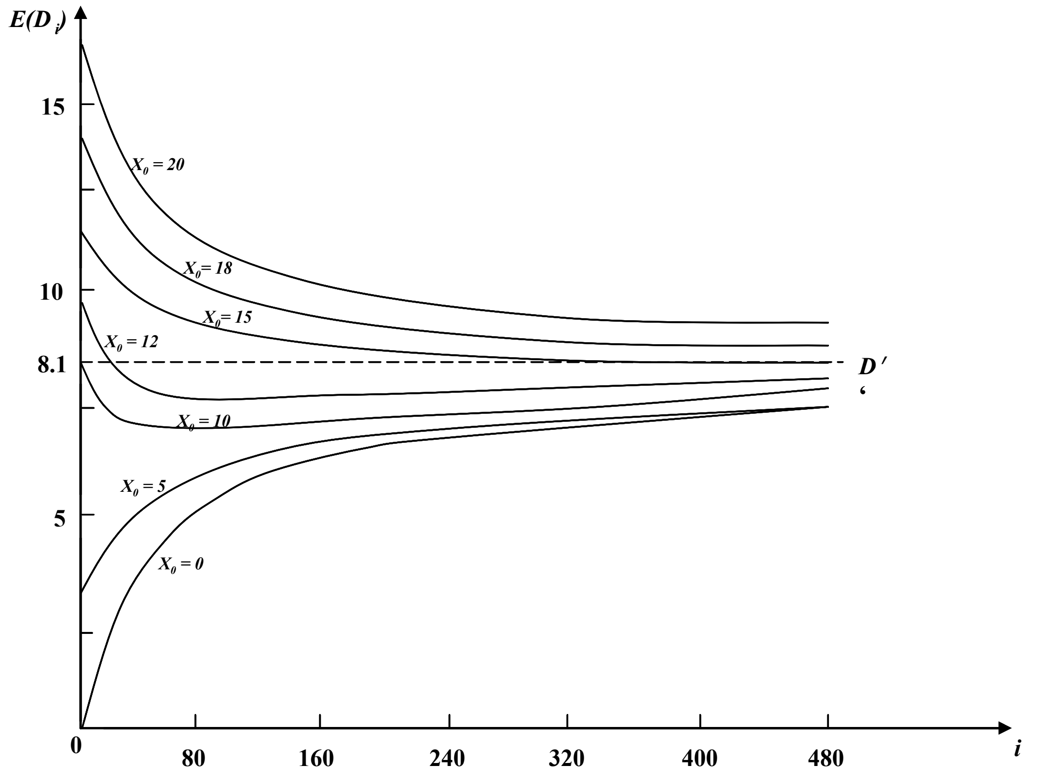


Figure 2-2 transient mean delay: M/M/1 queue with ρ = 0.9 [7]

The obvious way of obtaining a less biased result is to exclude the data collected during the transient period, when calculating steady-state estimates. However, determination of the lengths of the transient periods has proved to require quite elaborate statistical techniques [8]. In this thesis, the transient period of a single replication was taken as 1/50 of the total simulation time, which proves to be satisfactory as seen with the validations carried out at various stages (e.g. see Chapters 4-6). The transient data was excluded by resetting the relevant data distributions at 1/50 of the simulation length.

Obtaining sample means (for appropriate output analysis):

Output analysis is concerned with estimating a simulation model's (i.e. not necessarily the actual system's) true characteristics. Further, it is important to distinguish between validation and output analysis: this can be clarified with the following example:

Consider a simulation model constructed in order to estimate the mean of a certain property (denoted by μ) of a system. Note that the mean produced by the simulation model (denoted by x') will not necessarily be the same as μ. Suppose an estimate x_i of x' is made by a single simulation run. The error in x_i as an estimate of μ is given by;

$$\begin{aligned} \mathit{Error\ in}\ x_i &= |x_i - \mu| \\ &= |x_i - x' + x' - \mu| \\ &= |x_i - x'| + |x' - \mu| \end{aligned}$$

Validation is concerned with making the second absolute value (i.e. $|x' - \mu|$) small, while output analysis is concerned with making the first absolute value (i.e. $|x_i - x'|$) small. Therefore, analysing output data to correctly estimate the model's true characteristics is an important step in simulation experiments [1, 7].

The three main methods of analysing the output of stochastic simulations are, independent replications, batch means and regeneration [1].

The method of independent replications is based on repeating the simulation with a different seed value in the RNG, which was the method used throughout this research. This method is based on the assumption that the mean values of independent replications are independent even though the observations in a single replication are often correlated, e.g. the waiting times of packet i and $(i + 1)$ in a single replication are highly correlated, whereas the waiting time of packet i in two distinct replications are independent (and therefore the mean values are independent).

The procedure is to conduct m replications of size $(n_o + n)$ each, where n_o is the length of the transient phase. The first n_o observations of each replication are discarded and the rest are used to compute the mean of each replication ($\overline{x}_i$) as follows:

$$\overline{x}_i = \frac{1}{n} \sum_{j=n_0+1}^{n_0+n} x_{ij} \qquad i = 1, 2, \ldots, m$$

Where x_{ij} is the jth observation of the ith replication.

Then the overall mean for all *m* replications ($\overline{\overline{x}}$) is computed as follows:

$$\overline{\overline{x}} = \frac{1}{m}\sum_{i=1}^{m}\overline{x_i}$$

Finally, the Confidence Intervals (CI) need to be calculated, due to the following reason. Obtaining the true mean of the model (i.e. x') would require an infinite number of independent observations, which is clearly impossible. The best that can be done is to employ a finite number of observations (called a sample) and get probabilistic bounds; i.e. it is possible to get two bounds c_1 and c_2 (see Def 2.1), such that there is a high probability $1-\alpha$, that x' is in the interval (c_1, c_2).

$$\Pr\{c_1 \le x' \le c_2\} = 1-\alpha \text{ ....Def 2.1}$$

The interval (c_1, c_2) is called the Confidence Interval (CI) for x', while α is called the significance level. Further, $100 \cdot (1-\alpha)$ is called the confidence level and is traditionally expressed as a percentage (typically near 100%).

Let Z_m be the random variable $\left[\overline{\overline{x}} - x'\right] / \sqrt{d^2/m}$ where d^2 denotes the variance of x'. The central limit theorem states that if m is 'sufficiently large', Z_m will be approximately distributed as a standard normal variable (regardless of the distribution of $\overline{x_i}$'s). Further, for large m, $\overline{\overline{x}}$ is approximately normally distributed with mean x' and variance d^2/m. Since d is not generally known, d^2 is replaced by s^2 (sample variance) and the random variable $t_m = \left[\overline{\overline{x}} - x'\right] / \sqrt{s^2/m}$ is defined. Since s^2 converges to d^2 as m gets large, it follows from the central limit theorem that t_m is approximately normally distributed for large m. Therefore,

$$\Pr\left\{-z_{1-\alpha/2} \le \frac{\overline{\overline{x}} - x'}{\sqrt{s^2/m}} \le z_{1-\alpha/2}\right\}$$

$$= \Pr\left\{\overline{\overline{x}} - z_{1-\alpha/2}\sqrt{s^2/m} \le x' \le \overline{\overline{x}} + z_{1-\alpha/2}\sqrt{s^2/m}\right\}$$

$$\approx 1-\alpha$$

where $z_{1-\alpha/2}$ is the $(1-\alpha/2)$ quantile of the standard normal distribution such that the probability of the random variable being less than $-z_{1-\alpha/2}$ is $\alpha/2$; similarly, the probability of the random variable being more than $z_{1-\alpha/2}$ is also $\alpha/2$; therefore, the probability that the variable will lie between $\mp z_{1-\alpha/2}$ is $1-\alpha$.

Hence, a $100\cdot(1-\alpha)$ confidence interval (c_1, c_2) is given by;

$$\left(\bar{\bar{x}} - z_{1-\alpha/2}.\sqrt{s^2/m}, \bar{\bar{x}} + z_{1-\alpha/2}.\sqrt{s^2/m}\right) \text{..................Def 2.2}$$

When m is not 'sufficiently large' (typically $m \leq 30$ [1]), the actual probability that the population mean would be in the interval calculated using Def 2.2 will generally be less than 1-α. Therefore, the following alternative definition is used for smaller samples:

$$\left(\bar{\bar{x}} - t_{[1-\alpha/2;m-1]}.\sqrt{s^2/m}, \bar{\bar{x}} + t_{[1-\alpha/2;m-1]}.\sqrt{s^2/m}\right) \text{...........Def 2.3}$$

where $t_{[1-\alpha/2;m-1]}$ is the $(1-\alpha/2)$ quantile of the *t* distribution with $m-1$ degrees of freedom. Def 2.3 yields an exact $100\cdot(1-\alpha)$ CI in the case where the underlying $\bar{x}_i$ are normally distributed (as the random variable t_m has a *t* distribution with $m-1$ degrees of freedom when $\bar{x}_i$ is normally distributed); in other cases, it will result in approximate CIs. Note that $t_{[1-\alpha/2;m-1]} \rightarrow z_{1-\alpha/2}$ as $m \rightarrow \infty$ [7].

The confidence level used when presenting results in this report is 90% (i.e. α = 0.1); i.e. there is a 90% probability that the true mean of the model would be in the interval (c_1, c_2). Chapter 4 makes use of Def.2.3 (as m =30), while Chapters 5 and 6 use Def. 2.2 (as m =100).

Further, width of CI is inversely proportional to $\sqrt{mn}$; thus, a narrower CI could be obtained by increasing either m or n. However, increasing n is recommended in order to reduce the waste involved in discarding the transient events, n_o [1]. Therefore, in this research, n, i.e. the length of a single run, was increased as high as the available resources permitted.

The second method of analysing the output of stochastic simulations is the method of batch means. It works by running a single long simulation, removing the transient period and dividing the remaining observations into 'batches'. Each batch is treated similar to an independent run in the previous method [1]. While the batch means method incurs less waste as the transient period occurs only once, there is a risk of significant correlation among batches if the batch size is not chosen carefully. In addition, the national supercomputing service used for the majority of the simulations presented in this thesis[1] was designed for a large number of short jobs rather than a small number of long jobs. Due to these reasons, the batch means method was not employed in this research.

The other method of output analysis is the method of regeneration. This method relies on so called 'regeneration points', which essentially refer to the system returning to the initial state [1] (e.g. in a model with *N* queues, a regeneration point would be when all *N* queues are empty). The mean values of the regeneration cycles are used to calculate the overall mean; however, this calculation is more complex than in the other two methods since the length of a cycle is variable; also, extra processing is required in order to determine the regeneration points. Additionally, similar to the batch means method, this requires running a very long simulation which was not possible with the resources that were available for this research. Hence, the method of regeneration was not chosen.

2.3.5 Simulation clock

For dynamic models (i.e. state changes with time), it is necessary to keep track of the simulation time (notice this is different to real time) throughout the entire simulation. This is achieved by means of a simulation clock. There are primarily two approaches for advancing the simulation clock. The first approach, namely fixed-increment time advance, increments the simulation time by a fixed amount and checks to see if there are any events that are scheduled to occur. The second one, called next-event time advance, increments the time to the scheduled time of the next earliest occurring event. This is the approach currently used by most simulation software (including NS2) as events in real systems do not generally occur at fixed time intervals; thus saving on simulation time (as periods of inactivity

[1] The national supercomputing service employed for this research is HPC at University of Manchester (and later HPCx at University of Edinburgh).

are skipped over). These are called event-driven simulators. Naturally, the real time taken to complete an event-driven simulation depends on the number of events to be simulated.

2.4 Accelerated Simulation

2.4.1 Introduction

Event-by-event simulation modelling of present day packet networks is often intractable with available resources. For example, taking into account probabilities in the order of 10^{-3} which are currently used to specify packet loss probabilities in IP networks [17], obtaining just one thousand of these rare events by an event-by-event simulation would require the simulation of at least one million events, which would require a substantial amount of computer time and memory. As an example on a larger scale, it has been estimated that simulating just hundred seconds of Internet activity would require more than a year of CPU time, about 300 terabytes of main memory, and 1.4 petabytes of disk storage [18][1], which is impractical. Therefore, it is essential to explore methods that would minimize the amount of computer resources required for simulations and produce acceptable results within a reasonable time limit: this is where Accelerated Simulation (AS) comes in.

AS enables desired results to be obtained faster than in an ordinary simulation. This is mainly achieved by one (or the combination) of two methods, which are;

- Method 'a': Decreasing the number of total events required to obtain desired results
- Method 'b': Increasing the number of events per unit-time

The AS techniques developed in this research are based on method ‘a’; therefore, this is where the focus is on (section 2.4.2), and the term AS is commonly used throughout the rest of the thesis to refer to techniques of this type. Techniques based

[1] This estimation has been made in the year 2002. Given the growth of the Internet between then and now, a 100-second simulation of the current Internet would require even more computer resources than estimated in [18].

on method 'b' are not the focus of this research; however, they were utilized in simulating the majority of the scenarios presented in this thesis; therefore, method 'b' is also briefly described (section 2.4.3). Note that while method 'b' reduces the real time taken to complete a simulation, it does not reduce the amount of computer resources (e.g. memory) used; whereas method 'a' does both, i.e. reduces both the required resources and (therefore) the time.

2.4.2 Decreasing the number of events (Method ‘a’)

2.4.2.1 Speed-up factor

The speed-up factor (denote by ξ) is defined in this thesis as the number of events that need to be simulated in order to obtain a particular result (i.e. for the system to reach a given state) in the original model (i.e. non-accelerated), as a ratio to that of the AS model[1].

i.e. $\xi = \dfrac{H_O}{H_A}$ ………………………………………………Def. 2.4

where H_o and H_A are the total number of simulated events in the original and AS models respectively.

Def. 2.4 denotes that the AS model will have ξ times less events to simulate, which means in an event-driven simulator, the AS model would theoretically run ξ times faster (i.e. takes ξ times less wall-clock time) than the original one.

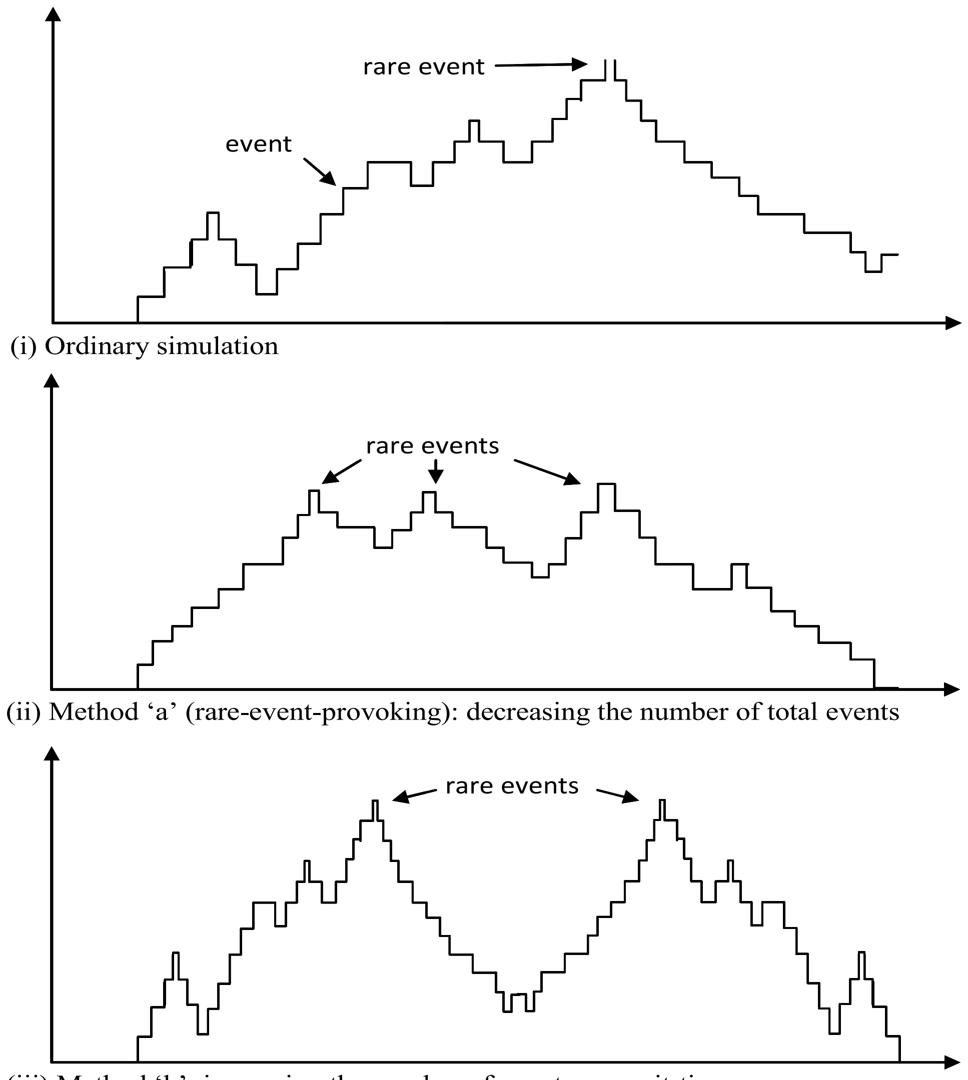
2.4.2.2 Techniques

AS techniques based on method ‘a’ are normally achieved through creating a simpler model of the system by removing or summarising detail considered less important. Consequently, the corresponding computer program(s) would be simpler and would require less computer memory and run faster.

[1] Speed-up factor is sometimes defined based on the state that can be reached by a given number of events in the original and AS models (therefore called ‘state-coverage’), e.g. ref [19].

Researchers have experimented with numerous AS techniques which reduce the number of events and achieved various degrees of success in terms of complexity and accuracy. One of the earliest acceleration techniques called 'Importance Sampling' (IS), is where the probabilities of the events are modified based on their relative importance to the required result [19]. Therefore in this method, a modified (or biased) probability distribution is introduced, in which the rare events of interest are forced to occur more frequently. Hence, IS is called a 'rare-event-provoking' (Figure 2-3(ii)) technique. Increasing the rare event probability means that the overall number of events to be simulated is reduced, requiring less computer resources. Further, the simulation outputs are weighted in order to adjust for any undesired effects caused by employing the biased distribution. Ref. [20] and [21] present extensions of IS for estimating cell-loss probabilities in ATM switches. Ref. [22] presents two efficient algorithms based on IS for heavy-tailed traffic. Ref. [23] and [24] are also examples of application of IS to Long Range Dependent (LRD) traffic.

Another well-known rare-event-provoking method is the RESTART (REpetitive Simulation Trials After Reaching Thresholds) technique. Consider a rare event A, the probability of which must be estimated, and an event C is defined such that $C \supset A$ and $\mathrm{p}\{A\} << \mathrm{p}\{C\} << 1$. Then, from Bayes Theorem, $\mathrm{p}\{A\} = \mathrm{p}\{C\} . \mathrm{p}\{A/C\}$. In an ordinary simulation, one cannot obtain a very good estimation of $\mathrm{p}\{A/C\}$ since it is only estimated from the small portion where C occurs. Therefore, in RESTART, this estimation is improved by having repeated simulation trials of the portion where C occurs, giving a greater confidence in estimating $\mathrm{p}\{A\}$ [25]. Ref. [26] has developed a fast simulation technique based on RESTART, which is capable of accelerating the simulation up to six orders of magnitude. Further, it includes a heuristic to determine the RESTART thresholds (which determines the points at which to repeat the simulation) which runs transparently to the user.



(i) Ordinary simulation

(ii) Method ‘a’ (rare-event-provoking): decreasing the number of total events

(iii) Method ‘b’: increasing the number of events per unit-time

Figure 2-3 Accelerated Simulation [27]

Another method of reducing the number of events required to obtain results is ‘variance reduction’. Here, the RNG streams are manipulated to introduce correlation among multiple runs so that the variance of the sample mean is reduced [1]. This is difficult as it requires considerable familiarity with the model in order to choose which parameters the correlation should be introduced to [27]. Also, if this is not done carefully, the results may turn out to have increased variance than in the ordinary simulation.

In addition to the above techniques and their extensions, there have also been various novel developments on AS techniques which reduce the number of events.

One such technique that was mainly aimed at ATM networks is the cell-rate method, where an ‘event’ is characterised by the change in the cell arrival rate (rather than the arrival of an individual cell) largely reducing the number of events to be simulated. This technique has demonstrated speed-up factors up to five (when compared with the cell-by-cell simulation) [28].

Another method, classifies the network traffic as foreground traffic and background traffic, and only the foreground traffic (which is of interest to us) is simulated. The background traffic is handled analytically; the service times of the foreground traffic is adjusted to compensate for the missing background traffic [29]. This is an example of a hybrid model, where both analysis and simulation are used to achieve AS. This gives a substantial reduction in the overall number of events to be simulated; however, validation tests have shown that there are some differences between the original and accelerated models. Therefore, the accelerated model is further adjusted with the use of a neural network [30]. In this technique, speed-up factor depends on the ratio of the number of events in the foreground and background traffic.

Yet another combination of analysis and simulation uses the concept of aggregating traffic sources. Note that the number of sources has a major impact on the run-time of a simulation. This technique replaces *N* multiplexed Short Range Dependent (SRD) traffic sources by a single equivalent ON-OFF source, which not only reduces the simulation time but also simplifies the scenario [31, 32][1]. Ref. [33, 34] extends the this technique to be applicable to Power-law traffic, where it has also been successfully demonstrated in conjunction with the RESTART technique, achieving significant levels of speed-up (in the form of ‘state-coverage’ – see section 2.4.2.1) as well as accuracy. Hence, this is a good example of successful concatenation of AS techniques. Experimental results have shown the state coverage of this combined technique to be $\approx 3X10^5$ which is a substantial improvement compared with the $\approx 10^4$ state coverage of the original model for the same number of events. An extension of this technique defines an event to be many packet-by-packet events of the technique in ref. [33, 34], achieving a further reduction of ‘events’

[1] This technique is used later in this thesis – see Chapter 5.

[35]. This technique has been successfully demonstrated on a Priority Queueing discipline[1], which is a variation of FIFO.

Another similar traffic aggregation technique exists in the literature where the single aggregate source is modelled based on fractional Brownian Motion (fBM). fBm is a Gaussian process with mean rate λ, and variance of increments over time t given by $\sigma^2 t^{2H}$, where σ and H (0,1) are constants. H is the Hurst parameter; depending on whether $H>1/2$ or $H<1/2$, the process is long or short-range dependent, respectively [36].

Ref. [37] presents SSIM (Session-level SIMulator), a new simulator based on the Time-Stepped Fluid Oriented (TSFO) approach for the acceleration of Mobile Ad Hoc NETworks (MANET) simulations. 'Fluid Oriented' means a close group of packets is modelled as a packet stream or packet train. Events are only generated when the rate of a traffic flow (packet stream) changes (similar to [28]). Since the rate of the traffic fluid generally changes at a rate much slower than the packet sending rate, the simulator is expected to handle fewer events. 'Time-stepped' means the simulation clock is divided into equal-sized units called 'time-steps' and a constant arrival rate is assumed within one time-step (i.e. ignoring the traffic variation during that period); thus achieving a reduction in the number of events.

2.4.3 Increasing the number of events per unit-time (Method 'b')

The obvious way to increase the number of events per unit-time, i.e. to shorten the real time taken to complete the simulation, would be to use a faster processor (Figure 2-3(iii)); however, it is more common to employ a number of processors simultaneously, i.e. parallelise. This can be done in two ways, which are; 'Parallel And Distributed Simulation' (PADS) and 'Parallel Independent Replicated Simulations' (PIRS) [27].

PADS (e.g. [38, 39]) is done by identifying sub-systems in the system to be modelled (e.g. nodes in a network), so that each sub-system can be modelled using a

[1] See section 3.2 for the definition of 'Priority Queueing'.

separate processor. While this can theoretically achieve a speed-up[1] equal to the number of processors, in practice it will be much less as the need for synchronization adds to the processing required. Further, it can be difficult to allocate sub-systems which result in the efficient use of resources [27].

PIRS (e.g. [40, 41]), on the other hand, is where the separate processors are used to carry out concurrent randomised replications of the same simulation, which is the method used in this research[2]. This method achieves a speed-up equal to the number of processors (as synchronisation or communication among processors is not necessary) [27]. Note that multiple processors can also be used to explore different sample paths (i.e. different input parameters) for the same model. Figure 2-4 illustrates the difference between PADS and PIRS.

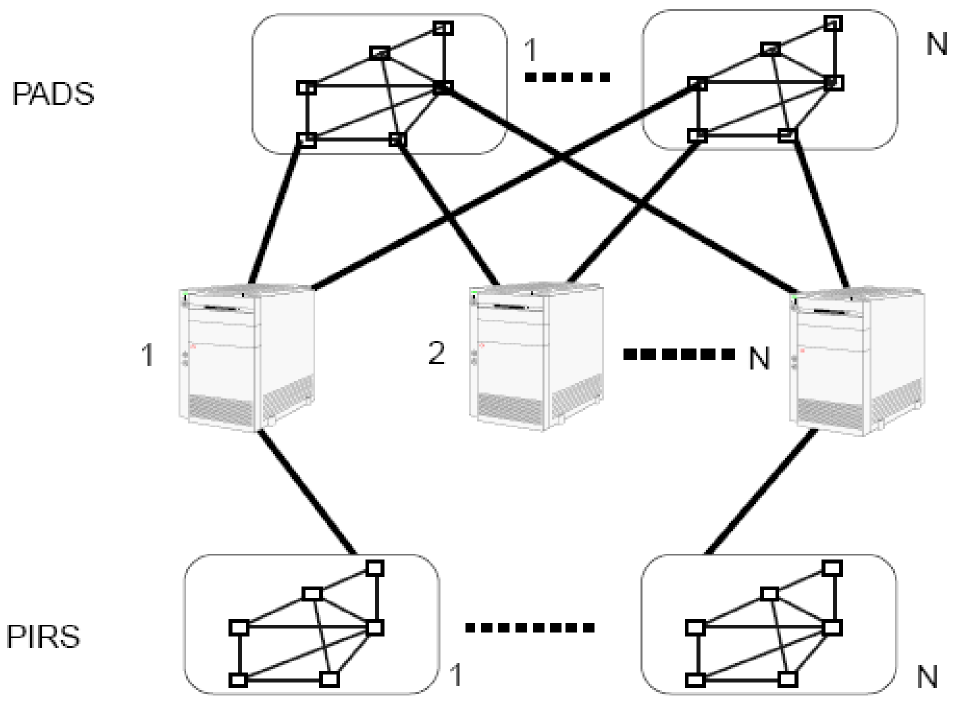


Figure 2-4 Methods of simulation parallelisation: PADS Vs PIRS [27]

[1] For method ‘b’, ‘speed-up’ means the reduction in wall-clock time, as there is no event reduction.

[2] The national supercomputing service was employed in order to achieve PIRS in this research.

2.5 *Simulation tool: Network Simulator 2*

The simulation tool used throughout this research is Network Simulator 2 (NS2)[1] [42], which is the most widely accepted event-driven network simulator today. It is of open source nature[2]. The core implementation is done in C++ (hence it is object oriented), while simulations can be set up via OTcl (the object-oriented version of Tcl) scripts (see Figure 2-5). The combination of the two languages offers a good compromise between performance and ease of use.

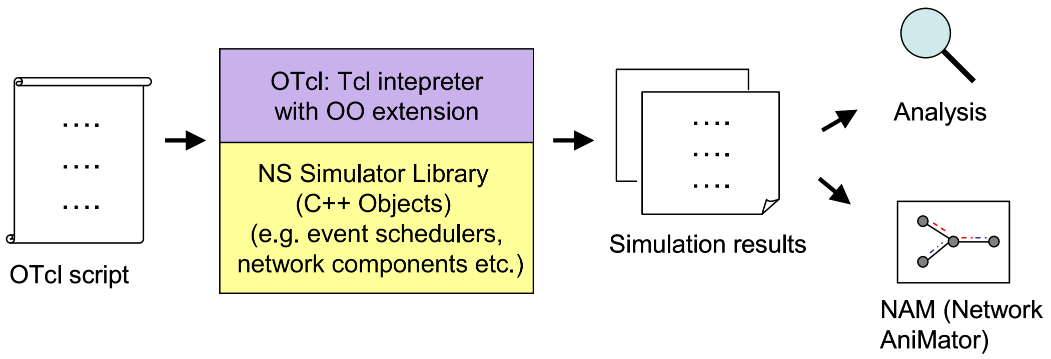

OTcl script
OTcl: Tcl intepreter with OO extension
NS Simulator Library (C++ Objects) (e.g. event schedulers, network components etc.)
Simulation results
Analysis
NAM (Network AniMator)

Figure 2-5 Layout of NS2 [43]

A graphical tool called Network AniMator (NAM) can be used to obtain a visual representation of the topology and an animation of events (e.g. flow of packets); however, it is not appropriate for statistical analyses. The standard way of performing statistical analysis on NS2 is to generate 'trace' files (i.e. text files which record events) and carry out post-scripting (e.g. awk, perl etc.) on these files to obtain summarised statistics (e.g. queue state probability distribution). However, trace files usually contain large amounts of data as they're in a raw format (i.e. not summarised), which results in consuming large amounts of disk-space and drastically slowing down the simulation. Also, the post-processing is an additional workload. Therefore, a more sensible way of obtaining statistical results from an NS2 simulation, which is the approach used throughout this research, is to extend the C++ code to directly output the summarised statistics of interest; thus avoiding

[1] NS2 version 2.27 was used at the start of this research, and version 2.29 later.

[2] Two other known network simulators are Opnet and QualNet, both of which are licensed software.

the creation of trace files and post-processing[1]. The amount of additional processing required for generating these summarised statistics is negligible, unlike when generating trace files. All extended code were verified by using the techniques described in section 2.3.4.1, e.g. anti-bugging, simplified test-cases etc.

One major disadvantage of NS2 is limited documentation, when compared with commercial software. Nevertheless, as experienced throughout this research, most problems can be solved by inspection of the source code and/or via the mailing list, which is effective because of the large user community. Another difficulty is, NS2 (as most simulators do) consumes a large amount of memory: the access to high performance computers in this research, minimised this issue to a large extent.

2.6 Summary

Simulation is the most common approach to network performance evaluation. The simulations typically used to model communication networks are of continuous time, discrete state, stochastic and dynamic nature. Advancing the simulation clock is done in an event-driven manner, as a result of which the real time taken to complete a simulation depends on the number of events to be handled in the simulator.

The main difficulty in simulating large networks using event-driven simulators is caused by the very large amount of computer time and memory required in order to obtain results. Hence, AS techniques attempt to obtain required results faster than in an ordinary simulation. This can either be done by increasing the number of events per unit-time or by decreasing the total number of events to be simulated: the latter (which is commonly referred to as AS in the rest of the chapter) is the focus of this research, which is achieved via traffic aggregation techniques. Reducing the number of events that need to be simulated cuts down the amount of resources required, and as a result enables the simulation of larger networks with the use of fewer resources. Existing AS techniques which reduce the number of events such as IS, RESTART, cell-rate simulation, traffic aggregation and TSFO have achieved considerable levels

[1] The only occasion where additional scripting was used in this research was to calculate the mean values from independent replications of simulations.

of speed up; however, these techniques are only applicable to FIFO scheduling disciplines (and its variations), whereas AS models based on the aggregation techniques developed in this research are applicable to Processor-Sharing and related disciplines.

Chapter 3

Non-FIFO Scheduling

3.1 Introduction

This chapter first presents the drawbacks of FIFO-scheduling in packet networks where traffic flows from many applications share the same links. Alternative scheduling mechanisms are discussed next, starting with Polling Systems in section 3.3. Then the ideal concept of Processor Sharing (PS) is discussed in section 3.4, where Generalised PS (GPS) is introduced. Finally, practical approximations to GPS and their advantages and disadvantages are briefly discussed.

3.2 Drawbacks of FIFO

When dealing with elastic traffic (i.e. under closed-loop control), performance mainly depends on the way the bandwidth is shared by the contending flows [44]; in FIFO, this would depend on the sending rate of each flow. This situation is vulnerable to ‘unfair’ allocation of bandwidth, where some flows may get a larger share of the bandwidth than the others, depending on their sending rates. For example in an extreme case where there is a severely ill-behaved source on the network, it could be continuously sending packets at a very high rate, occupying all of the available bandwidth, forcing the packets from other sources to wait further down in the queue experiencing excessive delays.

Priority Queueing (PQ) which is a variation of FIFO, attempts to provide protection from ill-behaved sources and minimise the delay experienced by packets with higher priority levels. In this discipline, the lower priority packets will only be served when all of the higher priority ones have finished service. The priority levels of packets are normally assigned based on the applications they belong to. PQ offers some guarantee for the packets with higher priority; however, the lower priority packets may be starved by the higher priority ones. Therefore, there still may not be a fair allocation of bandwidth.

The term ‘fairness’ is used here to mean its simplest form, i.e. all flows get the same share of the available bandwidth at a bottleneck. If a particular flow(s) cannot fully use its (their) share (e.g. due to low sending rates), then the excess bandwidth is shared equally among the remaining flows [45].

As discussed above, FIFO queueing does not guarantee a fair share of bandwidth for all the flows. Therefore, researchers have explored alternative scheduling mechanisms, which are discussed next.

3.3 Polling Systems

Polling systems have been extensively studied in the literature and are used in many fields. From the point of view of communication networks, a polling system is where a number of sub-queues (each containing packets from a particular traffic class) are served either in a cyclic (called cyclic polling systems) or random (called random polling systems) manner. Assume a total of N sub-queues. After serving sub-queue i, a cyclic polling system would serve (or poll) sub-queue $i+1$, while a random polling system would serve sub-queue j ($1 \leq j \leq N, j \neq i$) [46].

The number of packets served from a sub-queue during each cycle (also called period) depends on the type of service offered, which falls into the following three categories: i) exhaustive service ii) gated service iii) limited service. In the exhaustive discipline, the server will continue to serve the same sub-queue until its buffer is emptied, i.e. even the packets which arrive after the sub-queue was polled are also served within the same period. In the gated discipline, only those packets which were waiting in the buffer at the time the sub-queue was polled are served during the current period (this is similar to a ‘gate’ closing behind the packets already waiting, hence the name). In the limited approach, a sub-queue is served until either a) the buffer is emptied, or b) a pre-defined number of packets are served, whichever occurs first [47][1].

Most practical polling systems go through a switchover time which is the time needed to switch from one sub-queue to another (sometimes called reply interval or walking time). Zero switchover times are sometimes assumed for the sake of simplicity in analyses.

Albeit an improvement from FIFO, the exhaustive and gated disciplines still run the risk of being affected by ill-behaved sources. The limited discipline in contrast, does not suffer from this problem as the number of packets served from any sub-queue in a cycle is limited. Therefore, a source sending packets at a very high rate would be effectively increasing the length of its own sub-queue and delaying its own service, but cannot affect others. However, one major drawback is the lack of consideration of packet lengths. Since one or more 'packets' are served in each cycle (irrespective of packet lengths), sources sending large packets would naturally receive more bandwidth than the ones sending small packets. The solution to this problem lies in the ideal concept called Processor Sharing (PS), discussed in the next section.

3.4 Processor Sharing (PS)

3.4.1 Time-sharing and processor sharing

Time-sharing systems were first introduced in the early 1960s as an efficient method of providing 'simultaneous' computing services to multiple users. Time-shared computer usage is achieved by means of a Round Robin (RR) scheduler i.e. by giving each request a fixed quantum Q of time on the processor, after which the request is placed at the end of a queue of other requests. In other words, the queue of requests is constantly cycled, giving each user Q seconds on the machine during

[1] A cyclic polling system which follows the limited service discipline with the pre-defined number of packets to be served at a time set to one, gives rise to the well-known Round Robin scheduling mechanism used in communication networks.

each cycle (Figure 3.1). The case for which $Q \rightarrow 0$ is called a processor-shared system (also referred to as zero-quantum RR [48]) in which users are cycling around at an infinite rate, receiving an infinitesimal quantum of service infinitely often. When the total service time received equals a user's required processing time, he then leaves the system. When k customers are in the system, each is receiving service at the rate C/k (where C is the total capacity of the server), and hence the name processor-sharing as all customers indeed appear to be (simultaneously) sharing the capacity of the processor equally [49]. Kleinrock summarises the two schemes as follows: “processor-sharing is when the customers are given a fractional-capacity of the processor on a full-time basis, while time-sharing is when customers are given the full capacity of the processor on a part-time basis” [50].

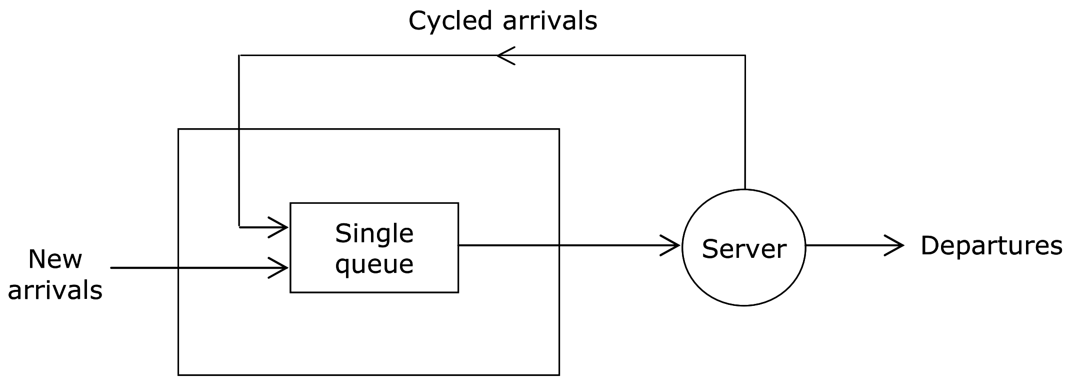


Figure 3-1 The round-robin system [50]

3.4.2 Extensions of PS

3.4.2.1 Discriminatory PS and Generalised PS

PS divides the server capacity equally among all the users; however, this kind of fairness is not necessarily the most preferred in communication networks. For example, as the Internet evolves to support an ever increasing range of services, there is a growing need for service differentiation to satisfy the diverse requirements of heterogeneous applications [51]. In order to provide service differentiation, traffic classes with weighting factors are introduced such that an increase in the weight for a particular traffic class leads to an increase in its received share of the bandwidth [52]. This can be achieved by extending the PS model to a multi-class system. Two such disciplines have emerged: they are, Discriminatory Processor Sharing (DPS) and Generalised Processor Sharing (GPS).

The DPS discipline (introduced in [49] under the name Priority PS) is a multi-class extension of the ordinary PS policy, with positive weight factors assigned to various classes of users[1]. Similar to PS, the service capacity is shared among all users present, but at a rate calculated based on the respective weights and the total number of users present. Where the weight and number of users in class x are ϕ_x and K_x respectively *(x=1,…,N)*, each class x user receives $\phi_x / \sum_{y=1}^{N} \phi_y K_y$ of the total service capacity [49, 52, 53].

As the service rates depend on the actual number of users present in the system, DPS does not guarantee a minimum service rate per class.

Similar to DPS, the GPS discipline divides the service capacity in accordance with class-dependent weight factors. The difference is that the capacity is not divided among all users present, but distributed across the classes, irrespective of the actual number of users present, and the users at the head-of-the-line of each (non-empty) class share the capacity. Therefore, GPS is known as Generalised Head-Of-the-Line (HOL) PS [51]. Given the weight associated with class x is ϕ_x, class x receives ϕ_x (a minimum of ϕ_x to be more specific – see section 3.4.2.2) of the total capacity whenever it is backlogged, where $\sum_{x=1}^{N} \phi_x = 1$ [54].

Unlike DPS, the GPS discipline guarantees a minimum service rate to each class [51, 53] and has emerged as an important step towards achieving differentiated quality-of-service in integrated-service networks [54].

The cyclic manner of PS disciplines (similar to the limited polling system) ensures that an ill-behaved source continuously sending traffic cannot affect other sources.

[1]Note that in a communication network, the equivalent of a ‘user’ would be a traffic flow (assuming fluid traffic), e.g. traffic generated at a single source, while a ‘class’ may consist one or more traffic flows having similar performance requirements.

In addition, serving a fixed amount (in terms of infinitesimally small units rather than packets, i.e. traffic is considered to be a fluid flow) in each cycle ensures that the fairness prevails even in the case of dissimilar packet sizes across sub-queues. In fact, PS (and its extensions) schedule(s) in favour of shorter packets and discriminates against longer packets [55] (in a manner that differs from one discipline to the other, e.g. DPS and GPS); thus providing protection from ill-behaved sources, also overcoming the problems of both FIFO and polling systems. The rest of this chapter focuses on GPS due to its relevance to this research.

3.4.2.2 GPS and its practical approximations

The GPS model can be described as follows. Assume *N* classes and a total service rate of *C*. Each of the *N* classes has its own buffer (see Figure 3-2). Class *x* is assigned a weight factor ϕ_x, ($\phi_x > 0, x = 1, \ldots, N$), and as mentioned before, the sum of the weights is 1. Hence, class *x* has a guaranteed minimum rate of $\phi_x \cdot C$, i.e., when all classes are backlogged (i.e. have non-empty buffers, hence requiring service), class *x* receives service at rate $\phi_x \cdot C$. If some of the classes do not require service (i.e. empty buffers), then the surplus capacity is redistributed among the other classes in proportion to their respective weights. Therefore, a backlogged class *x* receives service at a rate $\frac{\phi_x}{\sum_{y \in B_j} \phi_y} C \geq \phi_x C$, where B_j is the set of backlogged classes in the time interval $(t_j - t_{j-1})$

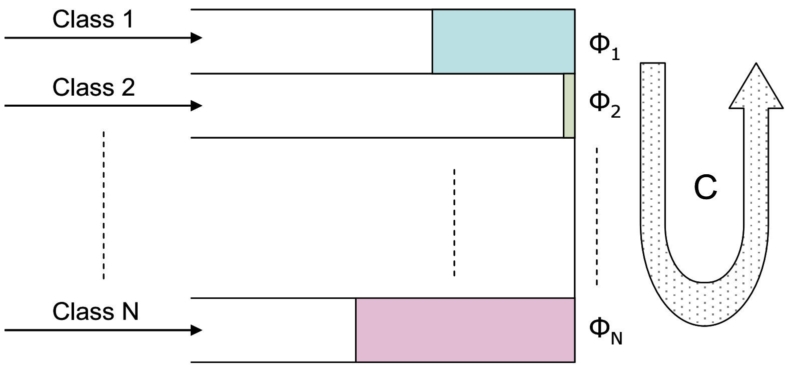


Figure 3-2 GPS System

The GPS discipline is an ideal that cannot be realised in practice as it requires packets to be infinitesimally divisible (due to the fluid traffic assumption)[1]. Various practical approximations have been introduced, which are discussed next.

3.4.2.2.1 Weighted Fair Queueing (WFQ)

WFQ maintains separate buffers similar to GPS, the difference is that the service in WFQ is packet-based (i.e. the fluid traffic assumption is not required). Fixed or variable length packets are served in an order determined so that, as far as possible, they complete service in the order which would prevail if they were in fact served using GPS; this is achieved by estimating the time at which each packet would have finished service in the ideal GPS system and then serving them in the increasing order of these finishing times (or service starting times in some WFQ algorithms).

Several WFQ algorithms exist, which differ from each other in the mechanisms used for estimating the service finishing (or starting) times. The Packet-by-packet GPS (PGPS) scheme (first proposed in [56] and implemented in [57]) is considered the closest approximation so far, to the theoretical (ideal) GPS Queue. It works by maintaining a hypothetical ideal GPS reference system to calculate the 'Virtual Time' (VT), which in turn is used to estimate the service finishing times and the packet with the smallest finishing time is served first[2]. Appendix B details the PGPS scheme, which was employed for some of the simulations presented in this thesis (see Chapter 4).

It has been demonstrated that the work accomplished on a given sub-queue in any interval in PGPS differs from that of the equivalent GPS system by at most one packet [58] (the largest packet, in the case of variable packet-lengths [59]). However, this tightly-constrained performance is achieved at the expense of a high computational complexity of *O(log(N))*, where *N* is the number of sub-queues [60].

[1] This statement holds true for PS and all of its extensions (e.g. DPS, GPS etc.); however the focus is on GPS.

[2] In a preemptive version of this algorithm, newly arriving packets whose finishing time is smaller than that of the one currently in service preempt the packet in service [56]. Due to practical reasons, the non-preemptive version is implemented in[57].

This is a result of the complications in keeping track of the VT and maintaining a sorted queue of finishing times.

Another popular WFQ algorithm is the Self-Clocked Fair Queueing (SCFQ) scheme (also known as Virtual Spacing [61]). Instead of linking VT (and therefore service finishing times) to a hypothetical reference system, SCFQ introduces a VT function which depends on the work in progress in the actual packet-based queueing system [62]. The elimination of the reference system makes SCFQ a much simpler technique to implement and it also achieves the fairness; however, the drawback is that it results in a higher end-to-end delay bound than PGPS. Another scheme called Start-time Fair Queueing (SFQ) schedules packets in the increasing order of the service starting times rather than the finishing times. VT is defined as the start tag of the packet in service at the time, which again eliminates the need for a reference system and is therefore computationally inexpensive [63]; however, this too suffers from the high end-to-end delay problem. Yet another technique, namely the Virtual Clock scheme [64] provides the same end-to-end delay bound as PGPS does, but has the disadvantage of unfairness [62].

3.4.2.2.2 Round Robin (RR) and Weighted Round Robin (WRR)

RR is another packet-based scheme which maintains a separate sub-queue for each traffic class. In the simplest approach, the HOL packet from each non-empty sub-queue is served per cycle. This gives rise to a limited polling system with the pre-defined number of packets set to one; which as explained in section 3.3, results in unfair allocation of bandwidth in the case of dissimilar packet sizes, and will not be a good approximation of GPS. As a solution, weights can be assigned to each sub-queue based on their respective packet sizes so that an appropriate number of packets (rather than one) can be served during each cycle such that the bandwidth is shared equally irrespective of the packet sizes: this is called WRR. However, if the packet sizes are not known or if there are variable packet sizes within the same sub-queue, then it is not possible to accurately define these weights. Therefore, a WRR system of this type, i.e. serving one or more ‘packets’ per cycle (hereafter referred to as ‘standard-WRR’) is not a good approximation of GPS.[1] Nevertheless, modifying standard-WRR to make the amount served per cycle (Q) a small, fixed value, rather than a whole packet and making Q infinitely small compared to the packet size (similar to the zero-quantum definition of PS – see section 3.4.1) results in an excellent approximation to GPS (which is illustrated later in this thesis – see Chapters 4 and 5). This approximation is named ‘WRR (Q $\rightarrow$ 0)’.

3.4.2.2.3 Deficit Round Robin (DRR)[2]

DRR provides an alternative way of overcoming the variable packet length problem in standard-WRR while keeping the complexity at a minimum (as it is a packet based scheme); hence it is simpler than WFQ. As the name suggests, DRR uses a ‘Deficit Counter’ (DC) in order to schedule services [60]. The DC for each sub-queue is initialised to a predefined value. The sub-queues are served in a round robin manner; if the HOL packet length is less than that queue’s DC, then the packet is served, and DC is decremented by the packet size. If the packet size is larger than its DC, then it is queued and the DC is incremented by the quantum value. If a particular sub-queue is empty, its DC is reset. A variation of DRR called Deficit Weighted RR (DWRR) assigns different quantum values to each sub-queue based on weights. A further extension, namely Modified DWRR (MDWRR) is where the DCs are not reset when a sub-queue is empty [65].

DRR can be used as an approximation to GPS[3] and is simpler to implement than WFQ algorithms, with a complexity of O(1); however, the accuracy of the approximation to GPS is not as good.

[1] The ‘byte-mode WRR’ available in some simulators may come to one’s mind at this point; however, it should be noted that while byte-mode calculates certain values (e.g. average queues size, packet drop probability) in bytes rather than packets, the service is still carried out packet by packet; therefore it is still the standard WRR.

[2] Variations of DRR are implemented by CISCO.

[3] DRR can also be used to model DPS[53].

3.5 Summary

Albeit the simplicity, FIFO scheduling cannot guarantee the fair allocation of bandwidth to all the contending flows; the ideal solution is PS, where each user receives an infinitesimally small quantity of service per cycle. GPS and DPS are multi-class extensions of PS where the quantity served per cycle is generalised by assigning positive weight factors to each traffic class (which typically consists of one or more traffic flows). GPS, which guarantees a minimum service rate to each traffic class, is currently attracting interest as an important base for achieving differentiated quality-of-service in integrated-service networks.

In spite of being ideal, GPS cannot be realised in practice as it requires the packets to be infinitesimally divisible. Several practical approximations exist, where there is a trade-off between the closeness to GPS and the simplicity of implementation. For example, WFQ algorithms (e.g. PGPS), which have been extensively studied in the literature, are complex mechanisms which provide good approximations of GPS, while simpler mechanisms such as WRR, DRR do not resemble GPS very well (see section 3.4.2.2 for specific details). The novel GPS approximation employed in this in Chapter 4 (section 4.5) and Chapter 5 is a modified version of WRR, namely 'WRR(Q $\rightarrow$ 0)', where the amount served per cycle (Q) is modified to be infinitesimally small compared to the packet size. This implementation is validated at various stages in the relevant chapters.

CHAPTER 4

PERFORMANCE EVALUATION IN THE PRESENCE OF POISSON TRAFFIC

4.1 Introduction

This chapter investigates the behaviour of a queueing system (e.g. at the output port of a router) fed by homogeneous Poisson traffic flows, scheduled by the GPS discipline. Novel traffic aggregation techniques to evaluate the queue state and System Time (ST) of a single flow are developed, which lead to Accelerated Simulation (AS) models which can substantially reduce the number of events that need to be simulated.

Section 4.2 defines the Poisson process. The topology and model employed in this chapter are described in section 4.3. The novel aggregation techniques for the evaluation of queue state (section 4.4) and ST (section 4.5) via AS models, are then presented, along with validation results. This is followed by a brief analysis of the speed-up factor that can be achieved by application of these aggregation techniques in AS models (section 4.6).

4.2 Poisson process

Poisson processes are often used to model the number of arrivals over a given interval, e.g. number of packet-arrivals to a queue, number of queries to a database (over a time interval *t*).

Consider the process shown in Figure 4-1 with arrivals at time t_1, t_2, etc.

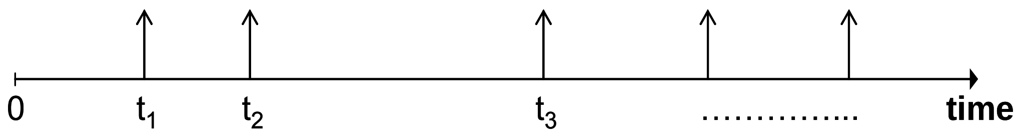


Figure 4-1 Arrival process

If the inter-arrival times 't_2-t_1', t_3-t_2' , etc. are IID and exponentially distributed, the number of arrivals over a given time interval has a Poisson distribution (which is the limiting case of a binomial distribution) [1]: this is called a 'Poisson process'. For a length of time t, the probability of k arrivals in a Poisson process ($P_k(t)$) is given by;

$$P_k(t) = \frac{(\lambda \cdot t)^k}{k!} \cdot e^{-\lambda \cdot t} \qquad \text{Eq. 4.1}$$

The mean of the exponentially distributed inter-arrival times is $1/\lambda$, i.e. λ is the average number of events per unit time.

An important property (used in this chapter) of a Poisson process is that merging of N Poisson streams with mean rate λ_i results in a Poisson process with mean rate λ given by [1];

$$\lambda = \sum_{i=1}^{N} \lambda_i \qquad \text{Eq. 4.2}$$

4.3 Simulation Topology And Model

The simulation experiments presented in this chapter are based on the topology shown in Figure 4-2. The 'Edge' node is a router with an output port which can route packets to the 'Destination' node.

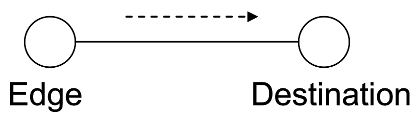


Figure 4-2 Topology

The ‘original model’ considered throughout this chapter is a model of the output port at the Edge node, consisting of a single queueing system (i.e. buffers and single server). See Figure 4-3. The output port is designed to allow *N* homogeneous traffic flows[1] (*flow #1 – flow #N*) to share the capacity of the server. Incoming packets are scheduled by the GPS discipline where there is a separate buffer for each traffic flow (with equal weights). The queue in a single buffer is referred to as a ‘sub-queue’; i.e. there is a maximum of *N* sub-queues at a time. The buffers are assumed to have unlimited capacity to hold packets.

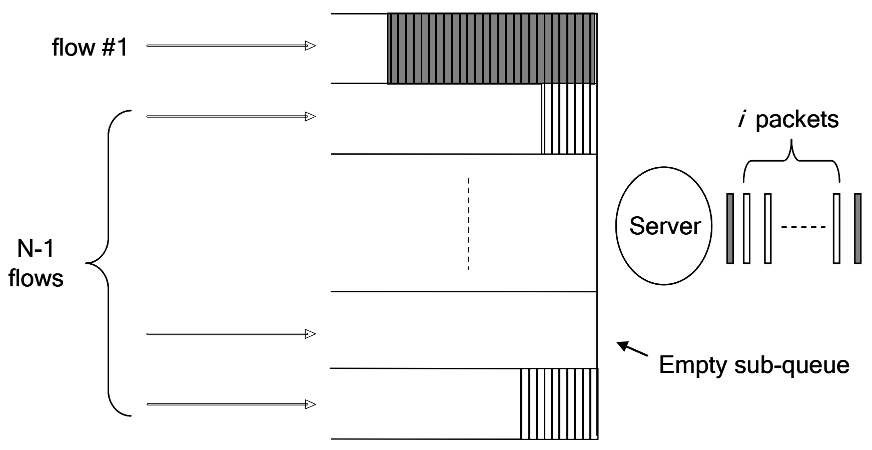


Figure 4-3 Original Model

The ‘aggregate model’ introduced in this chapter consists of a single flow (i.e. one of the *N* flows in the original model) which is the traffic flow of interest (Figure 4-4). Due to the homogeneity, the single flow in the aggregate model can be any arbitrary one from the original model. Therefore, *flow #1* can be chosen without loss of generality. Novel aggregation techniques are developed in the rest of this chapter, which ensure the queue state (section 4.4) and ST (section 4.5) of the aggregate model approximate those of *flow #1* (i.e. any single flow) in the original model.

[1] In this chapter, one traffic flow belongs to one class; therefore, for simplicity, the term ‘flow’ is used to mean both ‘flow’ and ‘class’.

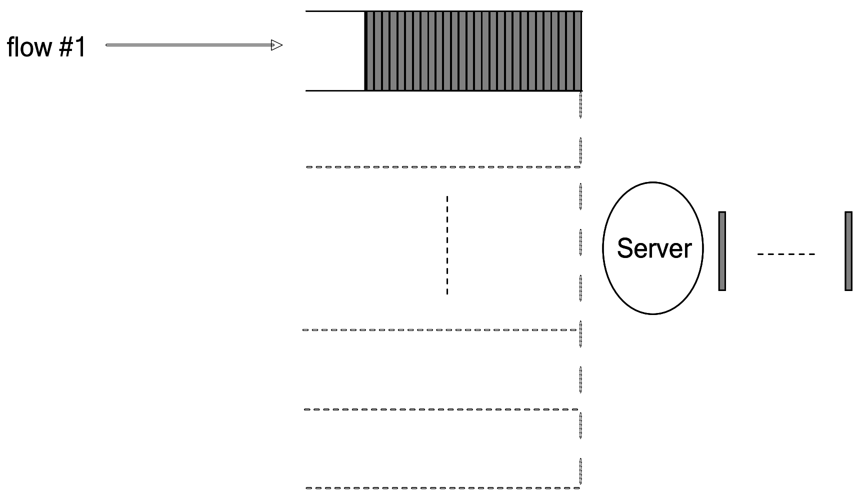


Figure 4-4 Aggregate Model

4.4 Queue state distribution of a single sub-queue: Poisson traffic

4.4.1 Introduction

'Queue state' (or queue length) is defined as the number of packets in the queue (i.e. sub-queue in this chapter). Section 4.4.2 presents a novel aggregation technique which ensures the queue state distribution of the aggregate model is the same as that of the original model. Section 4.4.3 describes the simulation set-up, while section 4.4.5 compares the original and aggregate models.

4.4.2 Novel Aggregation technique for queue state of single flow

As the aggregate model consists of *flow #1* only, the original model is analysed from the point of view of *flow #1*. Define the service time per packet to be the fundamental Time Unit (TU). In the case of homogeneous flows scheduled with equal weights, the overall effect of the GPS scheduler will be approximately the same as the standard-WRR (see section 3.4.2.2.2). In (standard) WRR, consider the time at which a *flow #1* packet has just completed service. Then, the next *flow #1* packet will not start service immediately, but only after *i* time units, where *i* is the number of other packets (i.e. from *flow #2* – *flow #N*) that will be served between these two *flow #1* packets (see Figure 4-3). When the weights are equal, *i* is the number of non-empty sub-queues (excluding *flow #1*) in the system.[1] Further, *i* is not a constant, but varies over time, ranging from *0* to *N-1*. This is because some sub-queues may be empty at times. Therefore, the instants at which the server will be available to *flow #1* is determined by the stationery distribution of *i*, which can be determined as follows:

In the case of *N* homogeneous sources generating Poisson arrivals with mean rate λ, the probability of having *j* active flows at a time would be binomially distributed with parameters *N* and λ. Therefore, the probability of having *j* non-empty sub-queues, *F(j)* can be approximated by the same distribution. i.e.

$$F(j) \approx {}^{N}C_{j}.\lambda^{j}.(1-\lambda)^{N-j} \qquad \text{Eq. 4.3}$$

Hence, the distribution of *i* can be approximated as shown in Eq. 4.3. Further, for large *N*, *F(j)* (therefore *i*) will approach a Poisson distribution.

The server taking a binomial/Poisson distributed vacation time between serving two *flow #1* packets can also be considered as spending a binomial/Poisson distributed time serving each *flow #1* packet. This realisation is useful as it enables the aggregation of the *N* flows into a single flow, resembling an M/G/1 queue with a binomial/Poisson service time distribution. This gives rise to the aggregate model. The M/G/1 approximation makes it possible to apply well-known queueing solutions which can be used in estimating the queue state probabilities. The Excess Rate (ER) queueing analysis for M/G/1 [32] is used here: this analysis presents a formula for *p(k)*, the probability that an arriving ER packet sees *k* packets already in the queue for an M/G/1 queue with utilisation ρ and mean arrival rate λ (in this chapter, $\rho = \lambda$ as the service rate is equal to one). Further, *p(k)* for a Poisson service time distribution is derived as a special case. Details are given in Appendix C.

[1] The *i* units of time can also be referred to as a 'vacation time' i.e. as far as *flow #1* is concerned, the server working on the other flows is equivalent to the server taking a 'vacation' and not being available for *flow #1* during this period.

4.4.3 Simulation set-up

GPS was approximated by implementation of PGPS (see section 3.4.2.2.1) which is currently the best known WFQ algorithm. PGPS is not included in the standard NS2 package; however, a contributed module is available [66]. In this module, the PGPS algorithm is implemented exactly as stated in Appendix B. The Virtual Time and B_j (set of active flows) are updated at every arrival and departure, which are then used to determine the order of service. The number of sub-queues must be pre-defined by the user, and the traffic flows pre-assigned to these sub-queues (one or more traffic flows can be assigned to each sub-queue). Weights can be defined for each sub-queue. A set of verification results for this PGPS module (independent to [66]) is available at [59].

The Poisson traffic source was implemented by using the exponential ON-OFF source in NS2 (as there is no pre-defined Poisson source in NS2). In order to generate Poisson distributed arrivals, the ON time is set to zero and OFF time is set to $1/\lambda$.

4.4.4 Validation of Poisson Traffic Source

The traffic source was validated as follows: Poisson traffic from a single source was fed in to a FIFO queue, and the queue state probabilities were compared with those of the M/D/1 theoretical values over a range of utilisations. The results (Figure 4-5) show that the traffic source is accurate.

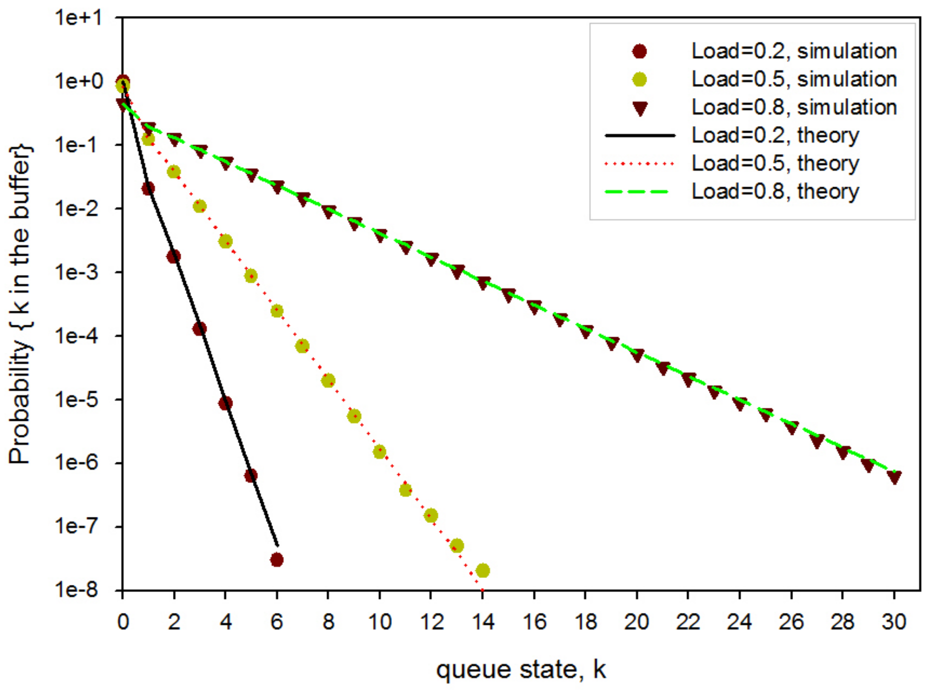


Figure 4-5 Validation of Poisson traffic source using an M/D/1 model

4.4.5 Results and Discussion

Simulation results based on a range of input parameters were obtained in order to validate the aggregate model proposed in section 4.4.2. The estimated queue state probabilities of the aggregate model are compared with those of *flow #1* of each corresponding simulation model, for the following 8 scenarios.

- N=20, N=100 (each with $\rho = \lambda = 0.3, 0.4, 0.5, 0.6$)

The simulated *flow #1* queue state probabilities were obtained as follows. Rather than considering *flow #1* only, the queue state probabilities of all *N* sub-queues were measured, and their average values are presented as the queue state probabilities of *flow #1*. This method yields statistically better results and is accurate because of the homogeneity of the *N* flows and sub-queues. The results are shown in Figure 4-6 - Figure 4-9.

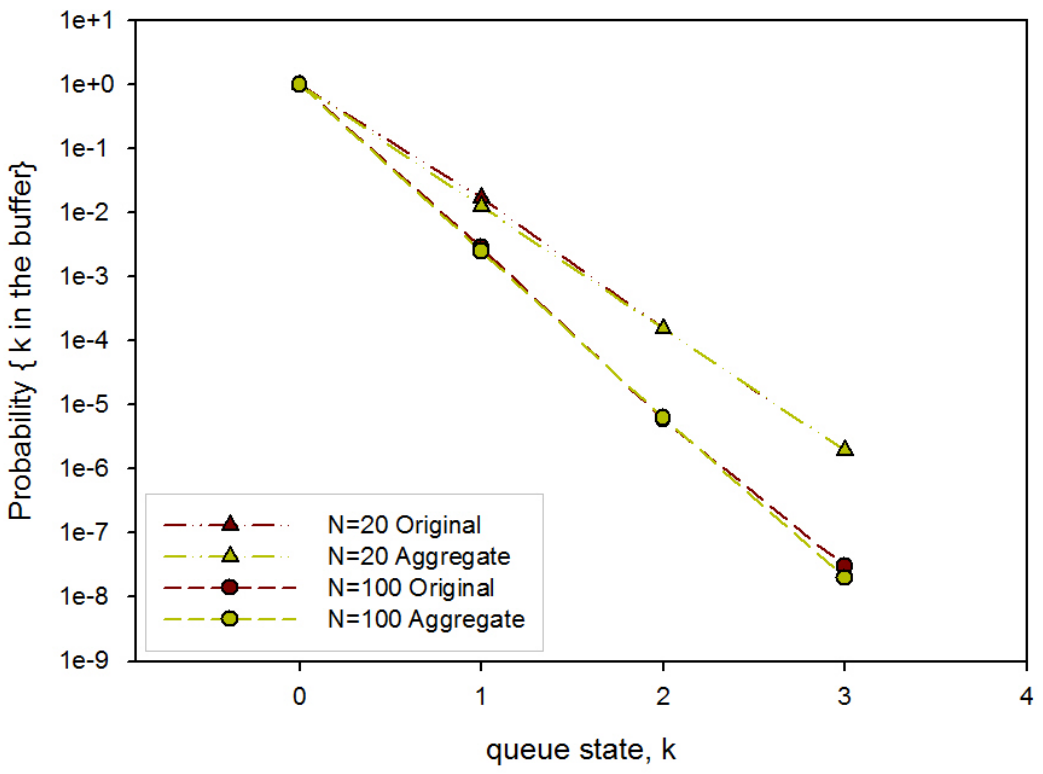


Figure 4-6 Comparison of queue state probabilities: ρ = 0.3

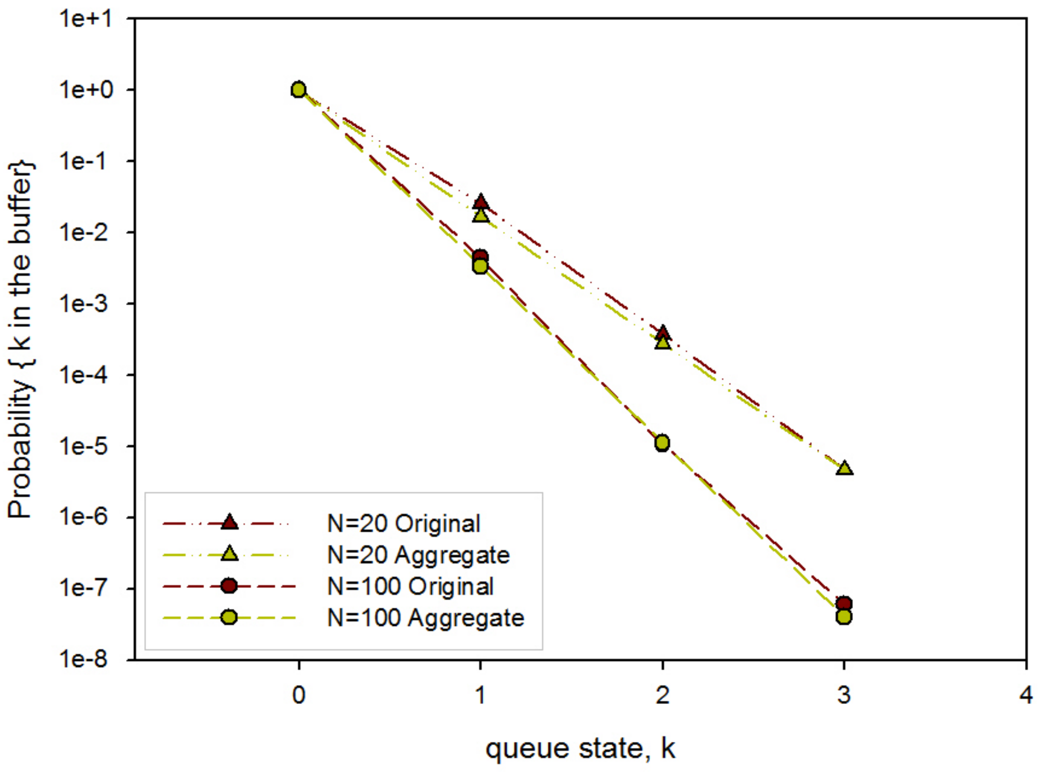


Figure 4-7 Comparison of queue state probabilities: ρ = 0.4

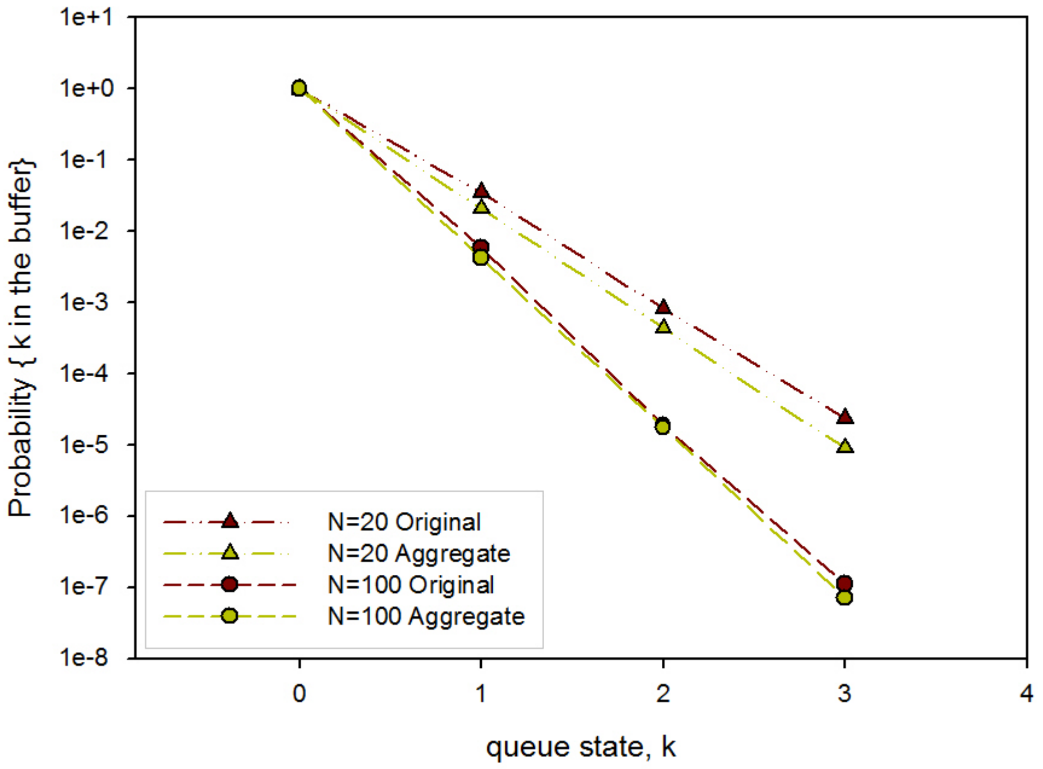


Figure 4-8 Comparison of queue state probabilities: ρ = 0.5

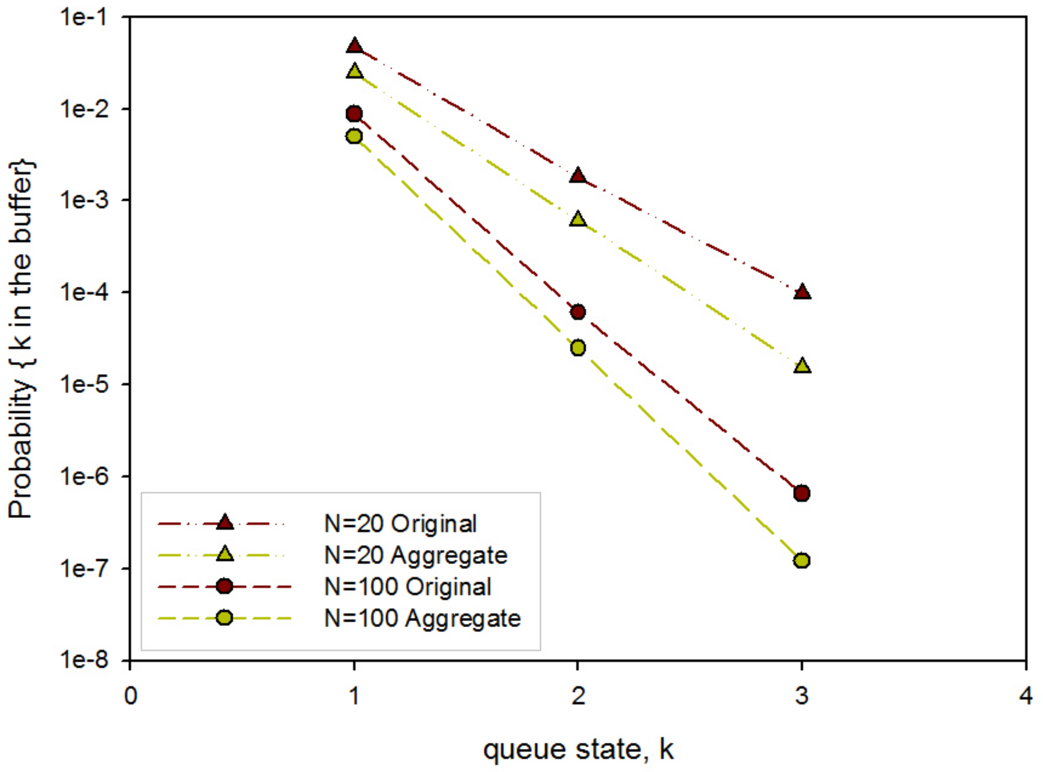


Figure 4-9 Comparison of queue state probabilities: ρ = 0.6

Note that the maximum queue state reached by any of these simulations (with $\approx 10^9$ events in each) is 3, as this is the queue state of a single sub-queue.

As illustrated, *flow #1* queue state probabilities of the original model are very closely approximated by the aggregate model, i.e. M/G/1 queue with Poisson service time distribution, over a range of ρ and N. Therefore, flow #1 could now be simulated on its own (i.e. without the remaining N-1 flows), with the service times of flow #1 adjusted based on the aggregation technique: this is an AS model. See section 4.6 for the speed-up factor achieved by this AS model.

4.5 Waiting Time (WT) and System Time (ST): Poisson traffic

4.5.1 Introduction

The most important performance measure of a GPS system (as perceived by users) is the system time (sometimes referred to as sojourn time), which is the total time a customer spends in the system.

In a FIFO queueing system, Waiting Time (WT) is the time spent in the queue waiting for service to begin (i.e. the difference between the time a packet enters the queue and begins service), while System Time (ST) is WT plus service time (denote by δ). However, in the ideal PS system, there is no waiting line. This is because the server shares its capacity equally among all packets present in the system. In GPS, waiting lines may exist; however, (similar to PS) waiting does not come to an end when a packet starts service (unless there are no other packets in the system). Therefore, in PS and its extensions, the term 'Waiting Time' refers to the total time a packet spends in the system without being served; in other words, WT is the total time the server spends serving other packets during the ST of the packet in question (i.e. the difference between ST and δ).

The ST of PS systems (with Poisson traffic) has been extensively investigated in the literature. Ref. [1] gives the mean ST of an M/G/1 queue with PS and utilisation ρ as:

$$E[ST] = \frac{E[\delta]}{(1-\rho)} \qquad \text{Eq 4.4}$$

Note that *E[ST]* is directly proportional to *E[δ]* which agrees with the fact that PS schedules in favour of shorter requests (see section 3.4.2).

The mean WT can be derived as follows:

$$E[WT] = \frac{E[\delta]}{(1-\rho)} - E[\delta]$$

$$E[WT] = \frac{E[\delta].\rho}{(1-\rho)} \qquad \text{Eq 4.5}$$

Notice the mean value of WT and ST are the same as those of the M/M/1 FIFO queue; however, the distributions are different [1]. Further, determining the distributions of WT and ST of PS systems has proved to be rather complex. For the M/M/1 PS queue, ref. [67] has derived the LST of the distribution of WT and ref. [68] has derived an integral representation of the distribution of ST. Ref. [69] has derived the LST of the distribution of ST for the M/G/1 PS queue and also M/D/1 as a special case. Based on [69], ref. [70] has derived the distribution $pr\{ST > x\}$ for the M/D/1 PS queue and shown this to be in the form $\alpha e^{-\gamma x}$ (α and γ being constants) as x becomes large.

The main result in [70] is as follows:

As $t \to \infty$

$$pr\{ST > t\} \approx \alpha e^{-\gamma t} \qquad \text{Eq 4.6}$$

where γ is given by

$$\frac{\lambda\delta(\lambda-\gamma) + \gamma - \gamma e^{(\lambda-\gamma)\delta}}{\delta(\lambda-\gamma)(\lambda-\gamma e^{(\lambda-\gamma)\delta})} = \frac{1}{\rho} \qquad \gamma \geq 0$$

and $\alpha = \dfrac{(1-\rho)(\lambda-\gamma)}{2\lambda(1-\rho)-\gamma\rho(2-\rho)}$

Eq 4.6 is used (along with the solutions given in [1]) in section 4.5.3 for the validation of the novel implementation of a GPS system which is presented in section 4.5.2. Then in section 4.5.4, a novel aggregation technique which can be used to derive the ST values in the aggregate model (i.e. of *flow #1*), is developed. Finally, in section 4.5.5, the aggregate model is validated by comparison with the original simulation model.

4.5.2 Simulation set-up: Novel approximation of GPS

First, the possibilities of using the PGPS approximation was explored by comparing its mean WT for an M/D/1 PS queue to the theoretical values (Eq 4.5). The results are given in Table 4.5.1. The unit of time is the service time per packet.

Table 4.5.1 Comparison of mean WT: theory and PGPS simulation

Utilisation (ρ)	Mean WT	
	Eq 4.5	PGPS simulation
0.3	0.4286	0.2198
0.4	0.6667	0.3333
0.5	1.0000	0.5002
0.6	1.5000	0.7499

It is apparent that the simulated mean WT is half of the theoretical value for the M/D/1 PS system, i.e. similar to M/D/1 FIFO[1]. The reason for this is that in the PGPS system as the name suggests (in fact in all WFQ algorithms), the service is carried out packet-by-packet; which results in a behaviour similar to FIFO[2]. Therefore, the PGPS approximation is not used in this section.

[1] Note that the mean WT of an M/D/1 FIFO queue is exactly half that of an M/M/1 FIFO queue.

[2] To be more specific, FIFO and WFQ have the same mean WT, but WFQ has tighter worst-case bounds [59].

Next, the WRR (Q→ 0) approximation to PS was considered as this is a simple approximation that doesn't serve the packet as a whole. As explained in section 3.4.2.2.2, for the WRR system to be a good approximation to GPS, the amount served at a time must be very small relative to the service requirements of the units being served [67].

In order to fulfil the above requirement, the simulator was modified as follows: a 'packet' (hereafter referred to as 'large packet') is represented by a train of smaller (consecutive) packets, where one small packet is served during each cycle of the WRR (Q → 0) scheduler, which was implemented using the Diffserv module in NS2.

A new ON-OFF source with a constant ON-period and exponentially distributed OFF-period was developed; so that it generates Poisson arrivals of equal-sized 'groups' of packets, where each group represents a large packet. The number of small packets per large packet was set to 1000 (in order to closely approximate GPS). Also, the time interval between the arrival of two consecutive small packets is made negligibly small, so that the ON-period is significantly small when compared to the OFF-period. This traffic source is named 'ExpoConst'.

Described next is the method used for the measurement of ST of the large packets. See Figure 4-10.

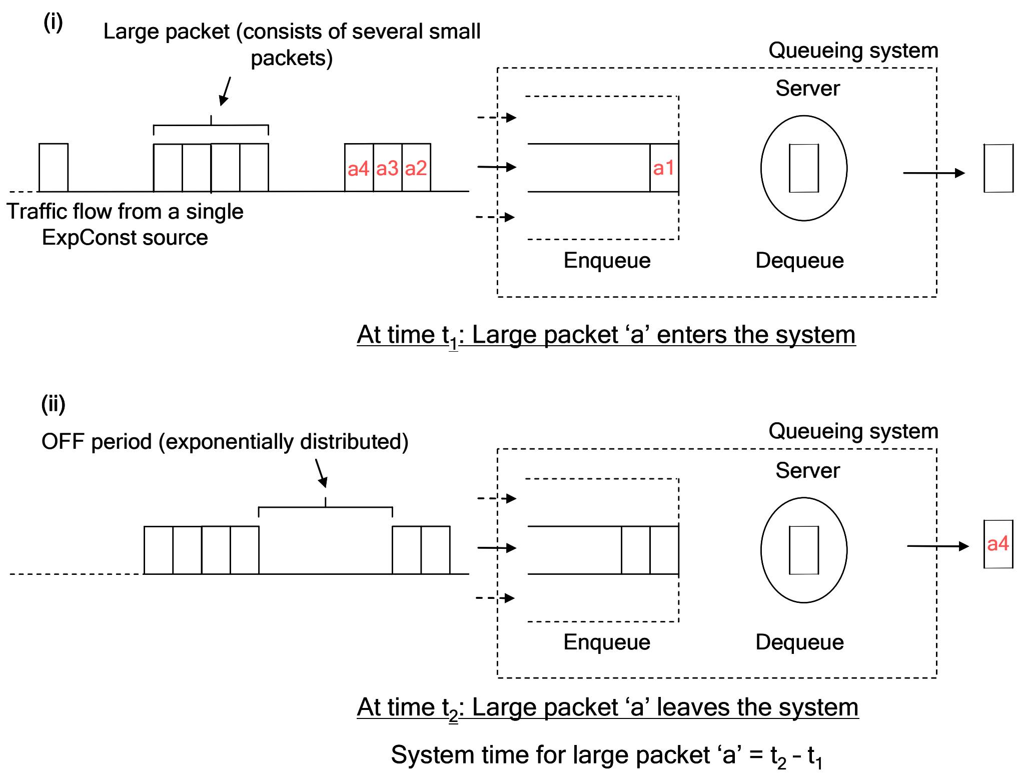

(i)
Large packet (consists of several small packets)
Queueing system
Server
a4
a3
a2
a1
Traffic flow from a single ExpConst source
Enqueue
Dequeue
At time t_1: Large packet 'a' enters the system
(ii)
OFF period (exponentially distributed)
Queueing system
Server
a4
Enqueue
Dequeue
At time t_2: Large packet 'a' leaves the system
System time for large packet 'a' = $t_2 - t_1$

Figure 4-10 Calculation of System Time, ST in the WRR (Q --> 0) system

ST (l.p.)= time at which the l.p. leaves the system (t_2)

– time at which the l.p. enters the system (t_1)

= time at which the last s.p. (of l.p. in question) leaves the system

– time at which the first s.p. (of l.p. in question) enters the system[1]

where *l.p. = large packet*

s.p. = small packet

At packet creation time, the ExpoConst source fills a packet-header field (called id) with the values '1' or '2' for the first and the last packets of each ON-period respectively, and '0' for all the other packets. Whenever a 'first' packet arrives at the buffer (i.e. enqueue), the current time (t_1) is temporarily recorded[2]. When the 'last' packet (from the same source, which is identified by the flow id) arrives, the previously recorded time is copied into a field (called enq_time) in its header, so that when it leaves the system, the ST for the large packet can be easily determined.

[1] Note that the time difference between the arrival of the first s.p. an last s.p. (i.e. ON period of the ExpoConst source) is made negligibly small.
[2] This is recorded separately for each traffic flow.

For instance in the illustration in Figure 4-10, a large packet consists of 4 small packets e.g. large packet ‘a’ consists of small packets ‘a1-a4’. The time t_1 (enqueue time of a1) is recorded in the header of a4. Thus, when a4 leaves the system at time t_2, ST for large packet ‘a’ can be calculated as ‘$t_2 - t_1$’.

4.5.3 Validation of WRR (Q → 0) approximation for GPS

For the validation process, simulation experiments of the WRR (Q → 0) system were carried out based on the topology and original model described in section 4.3, with *N=30*. All N sources were of the type ‘ExpoConst’ with equal sending rates i.e. the flows were identical.

Table 4.5.2 shows the comparison of the simulated mean ST values and the theoretical values given by Eq 4.4.

The TU is defined as the service time per large packet.

Table 4.5.2 Comparison of mean ST: Theory and WRR (Q -> 0) simulation

Utilisation (ρ)	Mean ST	
	Eq 4.4	WRR (Q→ 0)
0.2	1.2500	1.2680
0.4	1.6667	1.6647
0.6	2.5000	2.4348
0.8	5.0000	4.6922

The comparison of mean ST suggests that RR (Q → 0) system is (so far) a good approximation to the ideal GPS system. Next, the ST distribution was considered. Figure 4-11 shows the simulated ST values (with 90% CI) and those given by Eq 4.6[1]: these are again in excellent agreement with each other.

[1] The solution for γ was worked out by Newton-Raphson method.

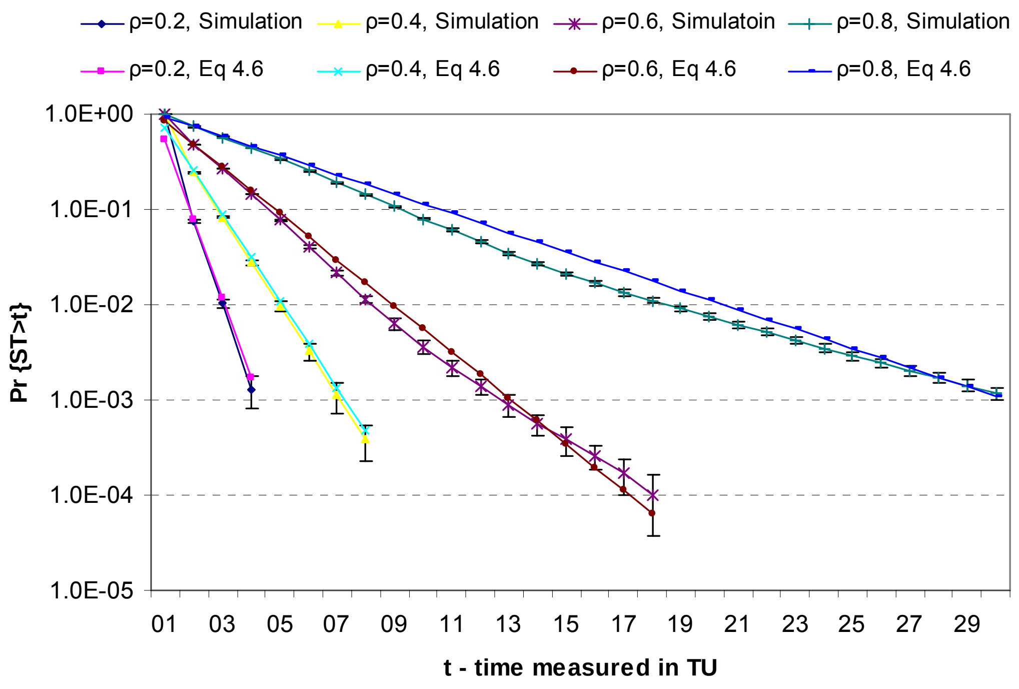


Figure 4-11 Validation of WRR (Q -> 0) system

Since the WRR (Q → 0) implementation is completely novel, further validation steps were carried out in order to ensure its credibility. Hence, the 'number in the system' was compared against the corresponding theoretical values. Ref. [1] gives the probability of n packets in the system ($p[n]$) as;

$$p[n] = (1-\rho).\rho^n \qquad \text{Eq 4.7}$$

where $n=0,1,2...$

(Note that this is the same as for an M/M/1 FIFO queue).

Due to simplicity of implementation, the 'number in the system' (N_R) was not measured directly. Instead, the 'number left in the system by a departing packet' (N_D) was recorded, which is equivalent to the 'number seen by an arriving packet' (N_A)in any system with single arrivals and single departures, i.e. $N_A=N_D$. Also, $N_A=N_R$ for Poisson arrivals due to its PASTA property[1]; therefore, $N_R=N_D$.

Table 4.5.3 shows the simulated probabilities of n jobs in the system along with the theoretical values given by Eq 4.7.

[1] PASTA: Poisson Arrivals See Time Averages i.e. the state of the system, e.g. distribution of no. of packets in the system, observed by Poisson arrivals equals the actual steady-state distribution.

Table 4.5.3 Comparison of p(n): theory and WRR (Q -> 0) system

n	ρ = 0.2		ρ = 0.4		ρ = 0.6		ρ = 0.8	
	RR (Q→ 0)	Eq 4.7	RR (Q→ 0)	Eq 4.7	RR (Q→ 0)	Eq 4.7	RR (Q→ 0)	Eq 4.7
0	0.7997	0.8000	0.6163	0.6000	0.4106	0.4000	0.1983	0.2000
1	0.1576	0.1600	0.2368	0.2400	0.2488	0.2400	0.1640	0.1600
2	0.0357	0.0320	0.0937	0.0960	0.1486	0.1440	0.1326	0.1280
3	0.0063	0.0064	0.0369	0.0384	0.0860	0.0864	0.1082	0.1024
4	0.0007	0.0013	0.0110	0.0154	0.0569	0.0518	0.0859	0.0819
5		0.0003	0.0043	0.0061	0.0270	0.0311	0.0692	0.0655
6		0.0001	0.0011	0.0025	0.0135	0.0187	0.0538	0.0524
7		0.0000		0.0010	0.0050	0.0112	0.0410	0.0419
8		0.0000		0.0004	0.0024	0.0067	0.0340	0.0336
9		0.0000		0.0002	0.0011	0.0040	0.0263	0.0268
10		0.0000		0.0001	0.0007	0.0024	0.0206	0.0215

The above results justify that the WRR (Q → 0) implementation is an accurate representation of a GPS system.

4.5.4 Novel Aggregation Technique for System Time

This section introduces an aggregation technique which ensures the ST distribution of *flow #1* packets in the aggregate model resemble those of the original model (even though the remaining *N-1* flows are absent).

As shown in section 4.5.1, the ST distribution *pr{ST>t}* is of the form $\alpha e^{-\gamma t}$, which means the WT/ST experienced by an arbitrary packet (which includes packets in *flow #1* as all flows are identical) entering the PS system is exponentially distributed. If the WT/ST of the *flow #1* packets could be accurately estimated, then it is possible to remove the remaining *N-1* flows and yet ensure *flow #1* behaves the same as in the original model. Figure 4-12 describes the technique used to estimate the ST values.

Step 1: Generate exponentially distributed values with mean given by (i.e. *E[WT]*)

Step 2: Add 1 to each value generated in Step 1, to obtain ST (as TU is equal to the service time)

Figure 4-12 Aggregation technique for ST

For the aggregate model, the values generated in step 2 would be the estimated ST values of *flow #1* packets. One might consider combining step 1 and 2, i.e. generating exponentially distributed values with the mean given by Eq 4.4 (*E[ST]*). However, this would result in a distribution that includes values less than one; which is incorrect as ST must be at least one.

4.5.5 Results and Discussion

In order to demonstrate the accuracy of the aggregate model, *pr{ST>t}* derived for the aggregate model are compared with corresponding results from the original (simulation) model, for a range of utilisations. Figure 4-13 illustrates the results (with 90% CI).

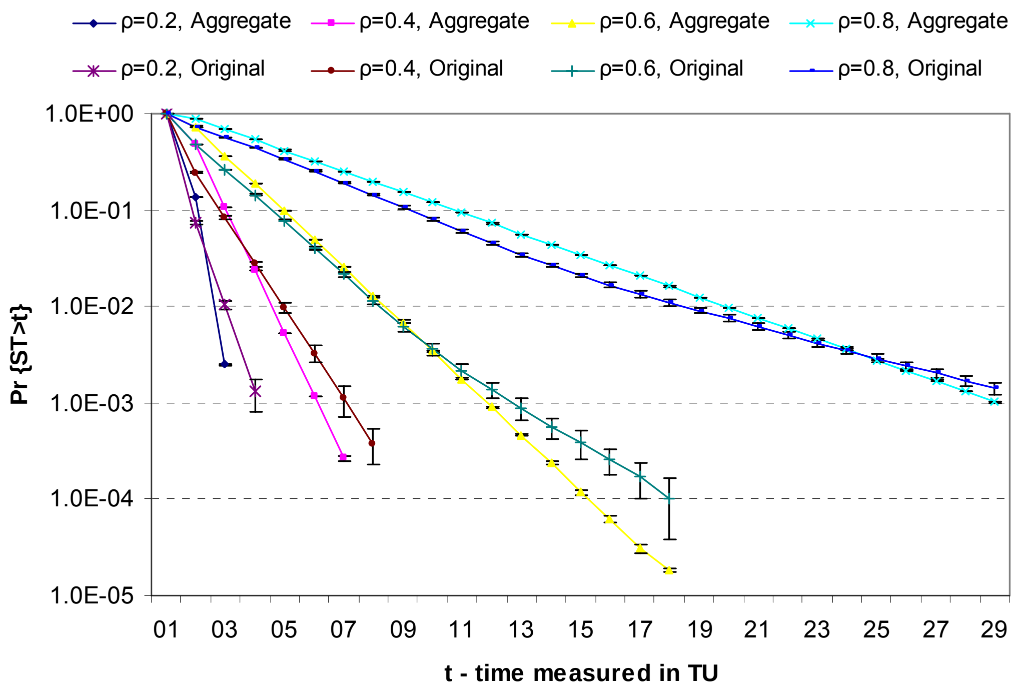


Figure 4-13 Validation of aggregate model: comparison of *pr{ST>t}*

The results show that the ST distribution of the original model is closely approximated in the aggregate model: this is significant as the need to simulate the original model (with N flows) is now eliminated, as the aggregate model could be simulated instead, i.e. simulate *flow #1*only, with the ST values adjusted to those estimated by the aggregation technique), which would accelerate the simulation, giving a substantial reduction of number of events as explained in the following section.

4.6 Application to Accelerated Simulation: speed-up factor

The aggregate model proposed in this chapter gives a substantial amount of event reduction when applied to Accelerated Simulation (AS). Following the definition given in Def. 2.4, the speed-up factor (ξ) achieved by such an AS model can be given by;

ξ = *Total number of simulated events in all flows / Total number of simulated events in flow #1*

(as the AS model consists of *flow #1* only).

Since *flow #1* in the aggregate model behaves approximately the same as in the original model, the number of *flow #1* packets that need to be simulated in the aggregate model in order to obtain a particular result is the same as the number of *flow #1* packets in the original model. Therefore, the simulation length required to obtain a particular result (or reach a particular state) will be the same for both models. Hence, for a simulation of length *L;*

$$\xi = \frac{\sum_{i=1}^{N} h_i \cdot L}{h_1 \cdot L} = \frac{\sum_{i=1}^{N} h_i}{h_1}$$

where h_i as the number of flow i events generated per unit (simulation) time ($i = 1,2\ldots N$).

Further, for *N* homogeneous Poisson traffic flows with mean rate *λ*,

$h_i = \lambda \qquad i = 1,2,\ldots N$

$\xi = N$

i.e. an AS model developed based on the aggregation techniques presented in this chapter will run *N* times faster than an equivalent original simulation model.

4.7 Summary

The original model considered in this chapter consists of *N* homogenous Poisson traffic flows with deterministic service times where *flow #1* is of interest to us. A GPS scheduler (with equal weights) is employed. This chapter has developed an aggregate model which consists of just *flow #1* instead of *N* flows, whilst ensuring the behaviour of *flow #1* in the aggregate model is approximately the same as in the original.

As explained in section 4.4.2, the service times of *flow #1* packets in the aggregate model are modified from the deterministic values to a Poisson distribution to compensate for the missing *N-1* flows: this makes the aggregate model resemble an M/G/1 queue. Hence, it is possible to employ well-known queueing solutions for calculating queue state probabilities; which has been successfully demonstrated by comparison of queue state probabilities estimated by the ER analysis for M/G/1 and those of *flow #1* in the original simulation model (Figure 4-6 - Figure 4-9) .

The most important performance measure of a PS system is the System Time (ST), i.e. the total time spent in the system. Section 4.5.4 presents a novel aggregation technique which ensures the ST distribution of *flow #1* in the aggregate model is the same as in the original model by estimating ST values based on an exponential distribution with the mean given by Eq 4.5. The accuracy of the aggregate model is demonstrated by comparison of these estimated values with the ST values of *flow #1* in the original simulation model.

In summary, this chapter presents novel aggregation techniques by which *N* homogeneous Poisson traffic flows could be reduced to a single flow whilst ensuring its buffer performance remains the same as when all *N* flows are present. Consequently, a substantial event reduction can be achieved (see section 4.6) by the application of these aggregation techniques to AS models.

Chapter 5

Performance Evaluation in the presence of Bursty traffic

5.1 Introduction

This chapter investigates the behaviour of bursty traffic, modelled by multiplexed ON-OFF sources, sharing a router output port scheduled by the GPS discipline. The burst-scale behaviour is analysed in comparison with FIFO equivalents, which leads to the evaluation of the 'number in the system' (X) via aggregation techniques. Further, a novel solution is developed for the System Time (ST), which is the most important performance metric. These solutions provide substantial event reduction when applied to Accelerated Simulation (AS) models.

Section 5.2 introduces packet-scale and burst-scale queueing and section 5.3 defines the topology and model used throughout this chapter. Section 5.4 describes the simulation set-up. The aggregation techniques for the evaluation of burst-scale X and ST are developed in sections 5.5 and 5.6 respectively, which also validates the aggregate models and explain how the developed aggregation techniques can facilitate event reduction, i.e. provide acceleration. Finally, in addition to the validations carried out for queue state and ST separately, section 5.8 present overall validation results for both performance metrics, based on Little's result.

5.2 *Packet-scale and burst-scale queueing*

Traffic generated by Poisson sources cause packet-scale queueing and is referred to as 'smooth traffic' [71]. The mean arrival rate from each source is the reciprocal of its inter-packet arrival time. At the packet-scale, the total packet arrival rate from all the sources is less than the server rate most of the time[1] [58]. Packet-scale queueing is caused by the accidental arrival of two (or more) packets within a Time Unit, TU (i.e. time taken to serve a single packet). The second packet (and any further ones) need(s) to wait in the buffer until the first one completes service (or in PS systems, all packets in the system alternate between being served and waiting while others are being served). At the packet-scale, since the arrival rate is only momentarily greater than the service rate and the average queue size will be in the order of tens of packets[2] [32].

It is well-known that Internet traffic cannot be successfully modelled by smooth traffic sources [72]. 'Bursty' sources such as on-off models are required instead. The ON-OFF source is the prototype of a bursty source and the superposition of on-off sources has been used extensively in Internet traffic modelling [71, 73-75]. It alternates between ON and OFF periods sending packets at a strictly deterministic rate (or according to a non-deterministic pattern such as a Poisson process) during ON periods and remaining inactive during OFF periods. The length of the ON and OFF periods could be exponentially, geometrically or arbitrarily distributed [32].

As shown in Figure 5-1, the distribution of X in a queue fed by bursty traffic consists of two parts. The first part, with a steeper slope is the 'packet-scale component' which coincides with the corresponding distribution of the M/D/1 queue provided packets are of fixed length[3]. The second part, which is called the 'burst-scale component' is more significant because of the appearance of large

[1] The exact proportion of time which the total arrival rate is less than the server rate depends on the load on the buffer.

[2] For very high loads (e.g. over 90%), the queue size could exceed 100 packets

[3] The GPS MD1 would in turn coincide with M/M/1 FIFO as shown in section 4.5.3)

queues with non-negligible probability. Therefore, in this chapter the evaluation of buffer performance is extended to the burst-scale.

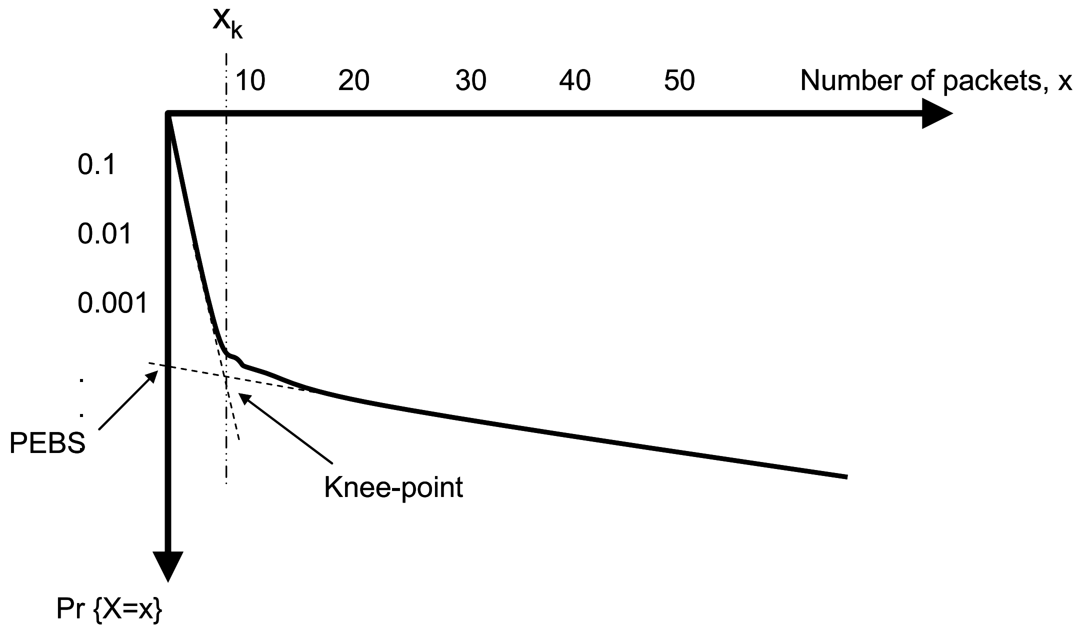


Figure 5-1 Distribution of X for a queue fed by a bursty source

The time-scale considered at the burst-scale is the cycle-time of an on-off source rather than packet inter-arrival times. At this coarse level, the discrete nature of packet arrivals can be ignored and the incoming packet stream can be regarded as a continuous fluid characterised by its instantaneous arrival rate [58]. Burst-scale queueing occurs when the instantaneous arrival rate exceeds the server rate over a significant period of time leading to a more or less constantly growing queue as long as the arrival rate excess lasts: this would result in an average queue size in the order of hundreds of packets[1] and is referred to as burst-scale queueing [32]. Note that Poisson traffic sources cannot cause burst-scale queueing as they send only one packet between idle periods.

The intersection point between the two queueing components is called the 'knee-point'. As a rule of thumb, the knee-point is where just enough sources are active to "fill the link"[71]. The point at which the interpolated burst-scale graph crosses the y-axis is called the 'Probability of Entering Burst Scale' (PEBS). Solutions for the knee-point and PEBS are presented in section 5.5.2.

[1] Could exceed 1000 packets for very high loads (e.g. over 90%)

5.3 Simulation Topology and Model

The topology and original model used in this chapter are the same as the ones used in Chapter 4, except for the traffic sources which are now ON-OFF with exponentially distributed ON and OFF periods. The arrival process of a single source during the on period is deterministic. This type of source produces bursty, Short Range Dependent (SRD) traffic.

Prior to defining an aggregate model, the burst-scale behaviour of ON-OFF traffic scheduled by GPS is observed and analysed by conducting a series of simulation experiments in the following sections.

5.4 Simulation set-up

The parameters used for the simulated scenarios in this chapter are listed in Table 5.4.1. Scenario 'VoIP70' features typical VoIP values with 70 traffic sources while the others are its parameter variations which cover a wide range of parameters. Each scenario was simulated with the load (ρ) values 0.4, 0.5, 0.6, 0.7, 0.8 and 0.9 which are hereafter referred to as the Scenario ID followed by the load e.g. VoIP50 (0.7) refers to the parameters listed for VoIP50, run with a 70% load.

Table 5.4.1 Input parameters

Scenario ID	N	T_{on}	T_{off}	h	ψ
VoIP70	70	0.96	1.69	170	0.3623
VoIP50	50	0.96	1.69	170	0.3623
VoIP30	30	0.96	1.69	170	0.3623
Ton*2.5	70	2.40	5.39	200	0.3079
Ton*5	70	4.80	12.35	220	0.2799

N = number of ON-OFF sources

T_{on} = mean ON time of a single ON-OFF source (sec)

T_{off} = mean OFF time of a single ON-OFF source (sec)

h = arrival rate of a single on-off source during the ON period (packets/sec)

ψ= activity factor i.e. $T_{on} /(T_{on} + T_{off})$

The WRR (Q $\rightarrow$ 0) approximation (see section 4.5.2) was employed for the GPS simulations. A new ON-OFF source named 'ExpoON' was developed. This is similar to the ExpoConst source (see section 4.5.2), the difference being that the number of large packets generated during each ON period can be specified by the user (rather than one large packet in ExpoConst). The number of small packets per large packet is set to 10[1] which appears adequate as shown by the validation results at various stages. Increasing this number improves closeness to the ideal GPS; however, it also greatly increases the computational power required.

5.5 Number in the system (X): bursty traffic scheduled by GPS

5.5.1 Introduction

The distribution of X in the original model (i.e. with GPS) is analysed in comparison with equivalent FIFO cases in section 5.5.2 which then leads to the development of aggregate models. The distribution of X is predicted based on the aggregate models, which is validated by comparison with simulation results of the original model, in section 5.5.3.

5.5.2 Novel Aggregation Technique for X

Statistics of X were collected for each scenario listed in Table 5.4.1. In the RR (Q $\rightarrow$ 0) implementation, X was measured by maintaining a global variable which was,

a) incremented by one when the first small packet (of a large packet) enters the system, and

b) decremented by one when the last small packet (of a large packet) leaves the system

First, the simulated Packet-Scale Decay Rate of X (PSDRX) values were compared with those of the theoretical M/D/1 PS (i.e. M/M/1 FIFO), as a form of validation. See Figure 5-2. It is apparent that the packet-scale behaviour follows M/D/1 PS.

[1] Even though the number of small packets per large packet was set to 1000 in the ExpoConst source, it is not feasible for the ExpoOn sources, as it generates a high number of large packets (e.g. 170) during an ON period, whereas ExpoConst generates only one large packet per ON period (i.e. Poisson traffic).

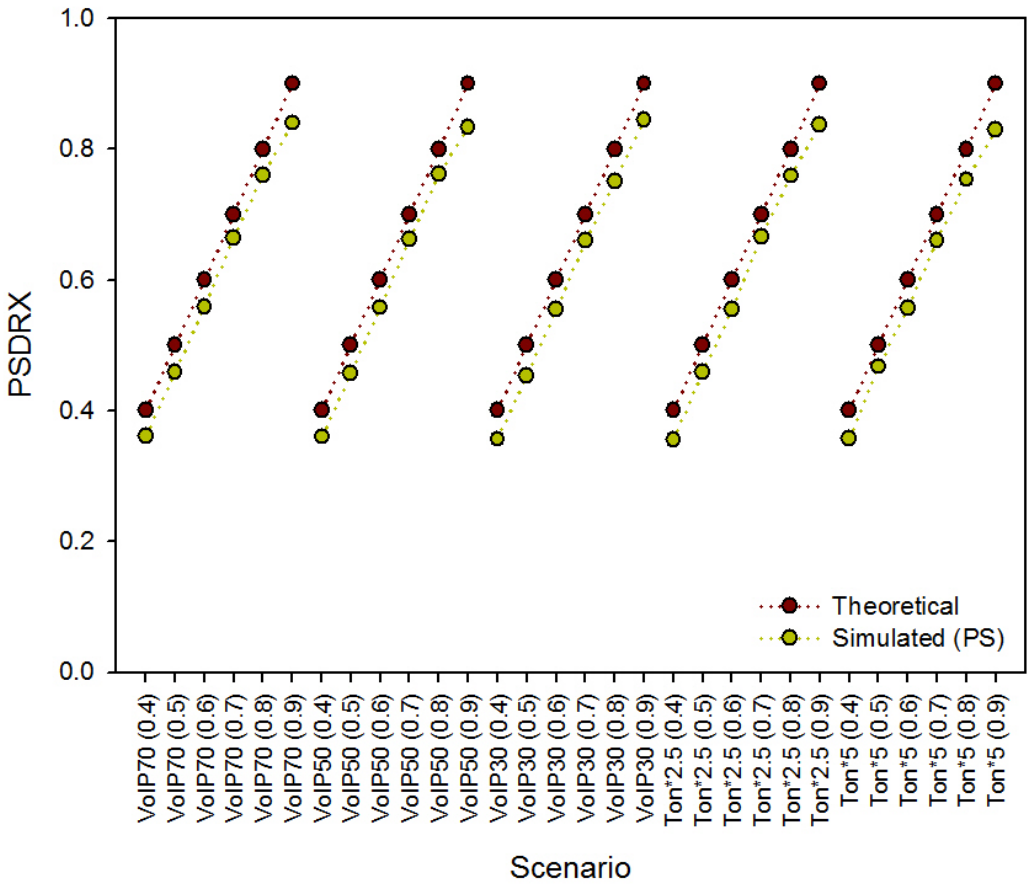


Figure 5-2 Comparison of PSDRX – Simulated and theoretical

Next, the entire simulated queue state distribution was considered, a large part of which is burst-scale behaviour. The results are presented in comparison with the corresponding FIFO cases (i.e. the input parameters listed in Table 5.4.1 using a FIFO scheduler) in order to reveal an important property (which is fully discussed in the rest of this section). Figure 5-3 shows the results for two of the load values of the VoIP70 scenario[1].

Observation suggests that the burst-scale queue state behaviour of GPS and FIFO coincide. Figure 5-4 clearly illustrates how GPS and FIFO follow distinct paths in the packet-scale region (as M/D/1 GPS coincides with M/M/1 FIFO i.e. not M/D/1 FIFO) and switch over to a common path at the knee point.

[1] Others are left out in order to keep the figure simple; however, Table 5.5.1 presents a complete set of results in a processed numerical form.

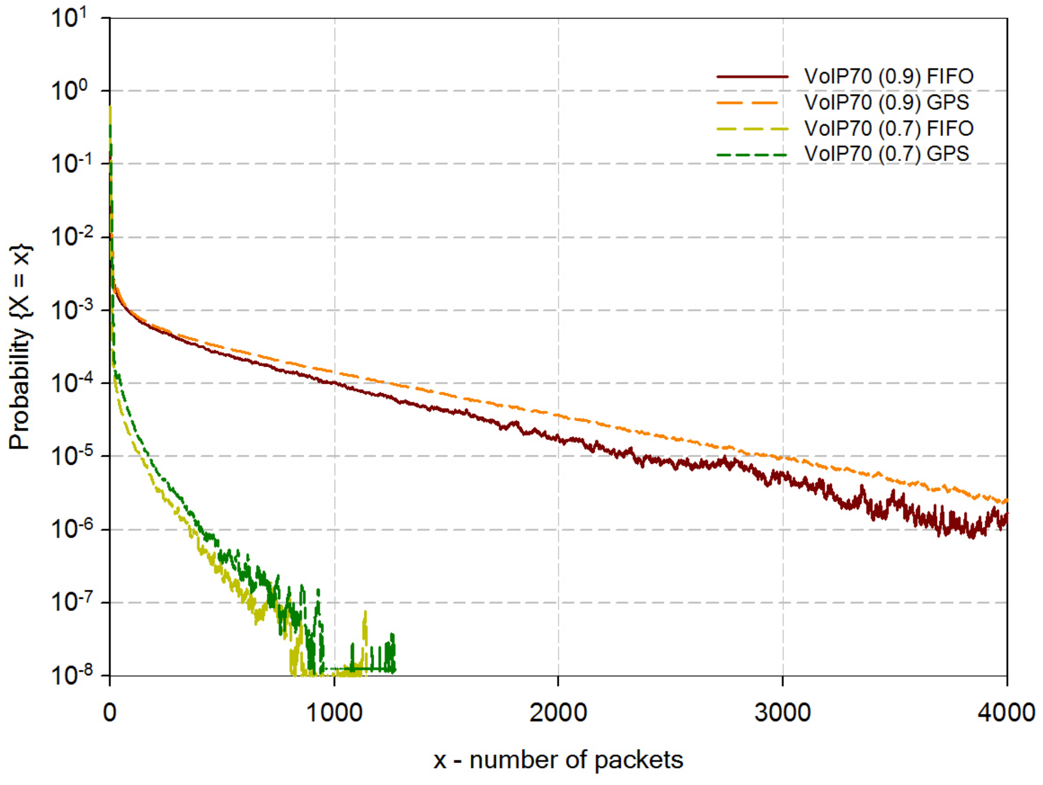


Figure 5-3 Number in the system (X) – FIFO and GPS

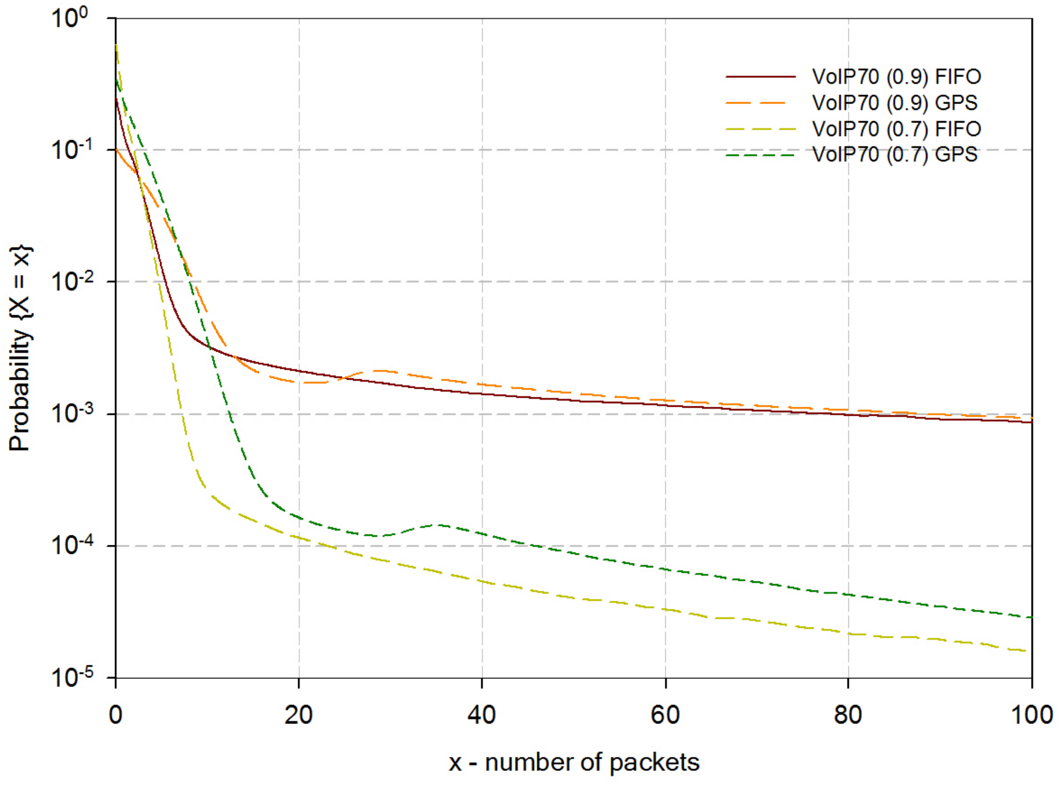


Figure 5-4 Number in the system – FIFO and GPS (Knee-point region)

Table 5.5.1 justifies the observation that the burst-scale distribution of X, of GPS and FIFO coincide, by comparison of Burst-Scale Decay Rate of X (BSDRX), of GPS and FIFO for all scenarios[1]. The values listed in the table were obtained as follows. A batch of 100 randomised runs was carried out for each scenario. The distribution of X for each randomised run was averaged to obtain an 'average distribution' per batch, for which a best-fit decay graph of the form $y=a.e^{(b.x)}$ was derived. The decay rate (BSDRX) is given by e^b.

Table 5.5.1 Comparison of BSDRX - GPS & FIFO (simulated)

Scenario	BSDRX		Scenario	BSDRX	
	GPS	FIFO		GPS	FIFO
VoIP70 (0.6)	0.9410	0.9563	VoIP30 (0.8)	0.9955	0.9961
VoIP70 (0.7)	0.9806	0.9861	VoIP30 (0.9)	0.9982	0.9982
VoIP70 (0.8)	0.9943	0.9943	Ton*2.5 (0.6)	0.9851	0.9771
VoIP70 (0.9)	0.9981	0.9987	Ton*2.5 (0.7)	0.9929	0.9922
VoIP50 (0.6)	0.9532	0.9620	Ton*2.5 (0.8)	0.9971	0.9972
VoIP50 (0.7)	0.9866	0.9865	Ton*2.5 (0.9)	0.9989	0.9995
VoIP50 (0.8)	0.9941	0.9948	Ton*5 (0.6)	0.9924	0.9930
VoIP50 (0.9)	0.9980	0.9986	Ton*5 (0.7)	0.9956	0.9955
VoIP30 (0.6)	0.9752	0.9767	Ton*5 (0.8)	0.9987	0.9987
VoIP30 (0.7)	0.9893	0.9899	Ton*5 (0.9)	0.9992	0.9994

The results justify that the burst-scale queue-state behaviour is scheduler-invariant over FIFO and GPS. This result is new to the literature and will be of significant value in the analysis of GPS systems. The need to simulate a GPS system (which is complex and consumes a large amount of resources) in order to analyse burst-scale X is now eliminated, as the much simpler FIFO equivalent could provide the same result. Moreover, aggregation techniques developed for the FIFO burst-scale could now be re-used for GPS. Ref. [76] is an example of such an aggregation technique, where N multiplexed ON-OFF sources scheduled by FIFO are replaced by a single equivalent ON-OFF source; however, the closeness of BSDRX (of this single source) to the original model (i.e. *N* sources) is not quite satisfactory (see section 5.5.3). A similar solution, also for FIFO, is given in [77], which is accurate,

[1] Results for load values 0.4 and 0.5 are not shown since burst-scale behaviour is almost non-existent for those cases. See section 5.6.3 for further details.

however, the formulae are complex and some of the parameters need to be determined via simulation. Ref. [31, 32] on the other hand, provides a similar single aggregate ON-Off source which is of approximately the same computational complexity as in [76], and is shown to demonstrate excellent agreement with the original model.

Ref. [31] gives BSDRX (denoted by R) of the single equivalent ON-OFF source as;

$$R \to \left[\frac{1 - \left[\frac{\ln\left(h/C\right)}{\ln(\rho)} + \frac{\left(h^2 T_{on} \rho\right)}{\left(C(1-\rho)^2\right)} \right]^{-1}}{1 - \left[\rho(1-\rho)^2 \Big/ (h/C) \cdot T_{on} .[(1-\rho)C + h \cdot \rho)] \right]} \right] \qquad \text{Eq 5.1}$$

Given the scheduler-invariant property of the burst-scale, Eq 5.1 is now applicable to the GPS case. Hence, the aggregate model (for GPS) can be defined as having a single equivalent flow of ON-OFF traffic, instead of the N flows in the original model. See section 5.7 for the speed-up that can be achieved by using this aggregate model to accelerate simulations.

In addition to BSDRX, the other parameter that needs to be worked out in order to evaluate the burst-scale distribution of X is the knee point.

Estimation of knee-point is very useful in itself, e.g. for buffer dimensioning. For instance, while a small buffer (in the order of hundreds of packets) may be sufficient for smooth traffic which only produces packet-scale queueing, a much larger buffer may be required in the case of bursty traffic. However, if PEBS (which depends on the knee-point) is very low (e.g. 10^{-10}), even bursty traffic may be considered as smooth (i.e. burst scale has not appeared) and a small buffer may be sufficient. Therefore, methods of estimating the knee-point and PEBS facilitate the efficient use of resources.

The following terms are defined:

p(x) – Pr{X=x}, $x = 0, 1, 2, \cdots\cdots\infty$

x_k – number in the system at knee-point

r – PSDRX i.e. $\frac{p(x+1)}{p(x)}\ for\ x < x_k$

R – BSDRX i.e. $\frac{p(x+1)}{p(x)}\ for\ x \geq x_k$

P_B – probability of experiencing burst-scale queueing

$p(x_k)$ is worked out first. See Figure 5-1. P_B is the summation of all *p(x)* from the knee-point onwards. Hence,

$$\sum_{x=x_k}^{\infty} p(x) = P_B$$
$$p(x_k) + p(x_k + 1) + p(x_k + 2) + \ldots\ldots = P_B$$
$$p(x_k) + R \cdot p(x_k) + R \cdot p(x_k + 1) + \ldots\ldots = P_B$$
$$p(x_k) + R \cdot p(x_k) + R^2 \cdot p(x_k) + \ldots\ldots = P_B$$
$$p(x_k) \cdot \left(1 + R + R^2 \ldots\ldots\right) = P_B$$
$$p(x_k) \cdot \left(\frac{1}{1-R}\right) = P_B \qquad (Geometric\ Series)$$
$$p(x_k) = P_B \cdot (1 - R) \qquad \text{Eq 5.2}$$

The knee-point can also be considered as part of the packet-scale. Hence,

$$p(x_k) = p(0) \cdot r^{x_k}$$
$$x_k = \frac{\ln(p(x_k) / p(0))}{\ln(r)} \qquad \text{Eq 5.3}$$

And $p(0) = (1 - \rho)$

Since the packet-scale X in the GPS queue is the same as in the M/M/1 FIFO queue;

$r = \rho$

Also by substituting Eq 5.2 into Eq 5.3;

$$x_k = \frac{\ln(P_B \cdot (1 - R)/(1 - \rho))}{\ln(\rho)} \qquad \text{Eq 5.4}$$

P_B, the probability of experiencing burst-scale queueing, can be quantified based on the concept of burst-scale loss systems [58]. Here, a small buffer (i.e. tens of packets) which is just enough to absorb packet-scale queueing is provided while burst-scale queueing will result in loss of packets. Therefore, the probability of an arriving packet being lost in a burst-scale loss system (which is given in [58]) is equivalent to the probability of experiencing burst-scale queueing P_B in an infinite-buffer system. Therefore;

$$P_B \approx \frac{1}{(1-\rho)^2 \cdot N_o} \cdot \frac{(\rho \cdot N_o)^{\lceil N_o \rceil}}{\lceil N_o \rceil !} e^{-\rho \cdot N_o} \qquad \text{Eq 5.5}$$

where $N_o = C / h$

C = server rate (packets/sec)

Note that Eq 5.1 considers the knee-point to be at $\ln(h / C)/\ln(\rho)$ which does not always give a good estimation of the knee-point especially at low loads. However, while the evaluation of the distribution of X requires an accurate estimation of the knee-point, ref. [78] shows that evaluation of decay rate is insensitive to the exact knee-point as long as it is large enough to be within the burst-scale. This is because the variation of *p(x)* is minimal in the burst-scale (i.e. $p(x+1)/p(x) \rightarrow 1$) especially at higher loads. Note that higher loads are indeed more significant because of the increased likelihood of burst-scale queueing.

5.5.3 Results and Discussion

The distribution of X in the aggregate model as a whole (i.e. both packet-scale and burst-scale), or the burst-scale on its own starting from the knee point onwards could now be derived based on the techniques developed in section 5.5.2. Introducing a solution for PEBS makes it possible to derive the burst-scale region without having to determine the knee point first. The probability at the knee-point can be given by;

$$p(x_k) = PEBS \cdot R^{x_k}$$

$$PEBS = \frac{p(x_k)}{R^{x_k}}$$

Substituting Eq 5.2,

$$PEBS = \frac{P_B \cdot (1 - R)}{R^{x_k}} \qquad \text{Eq 5.6}$$

For higher loads R → 1. Hence,

$$PEBS \rightarrow P_B \cdot (1 - R) \; as \; R \rightarrow 1 \qquad \text{Eq 5.7}$$

$$i.e. \; PEBS \rightarrow p(x_k) \; as \; R \rightarrow 1$$

Figure 5-5 illustrates examples of the closeness of the distribution of X in the aggregate model (derived based on Eq 5.7 and Eq 5.1 i.e. PEBS and BSDRX respectively) to the original one (i.e. simulation results of the original model with *N* flows scheduled by GPS, using the WRR (Q → 0) approximation).

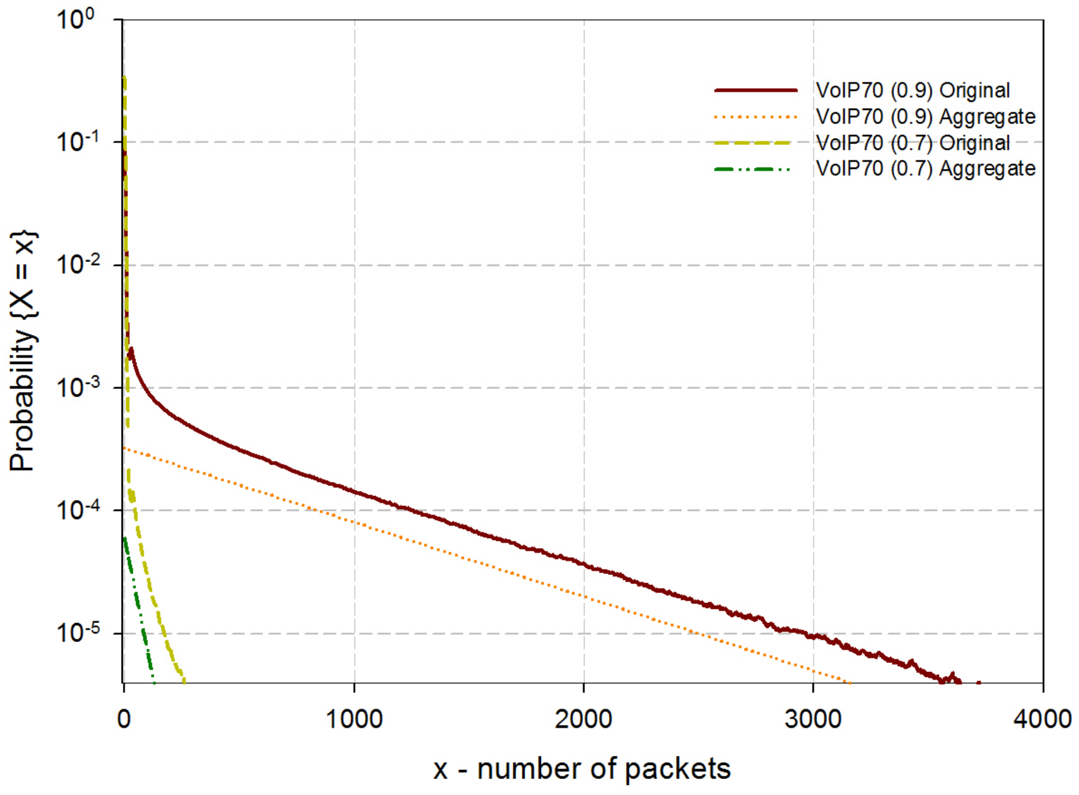


Figure 5-5 Validation of aggregate model: number in the system (X)

Figure 5-6 - Figure 5-10 compare PEBS and BSDRX (original and aggregate) for all scenarios. The simulated values were derived from the best-fit graph $y=a.e^{(b.x)}$, where PEBS and BSDRX were given by *a* and e^b respectively.

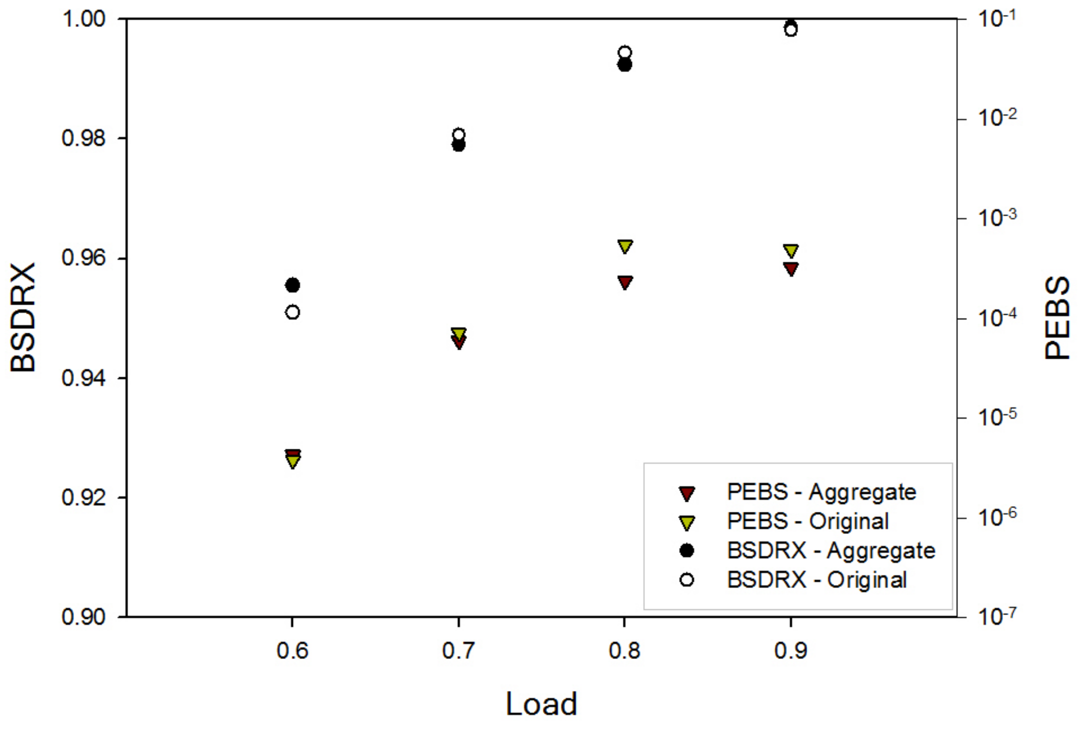


Figure 5-6 Comparison of PEBS and BSDRX: VoIP70

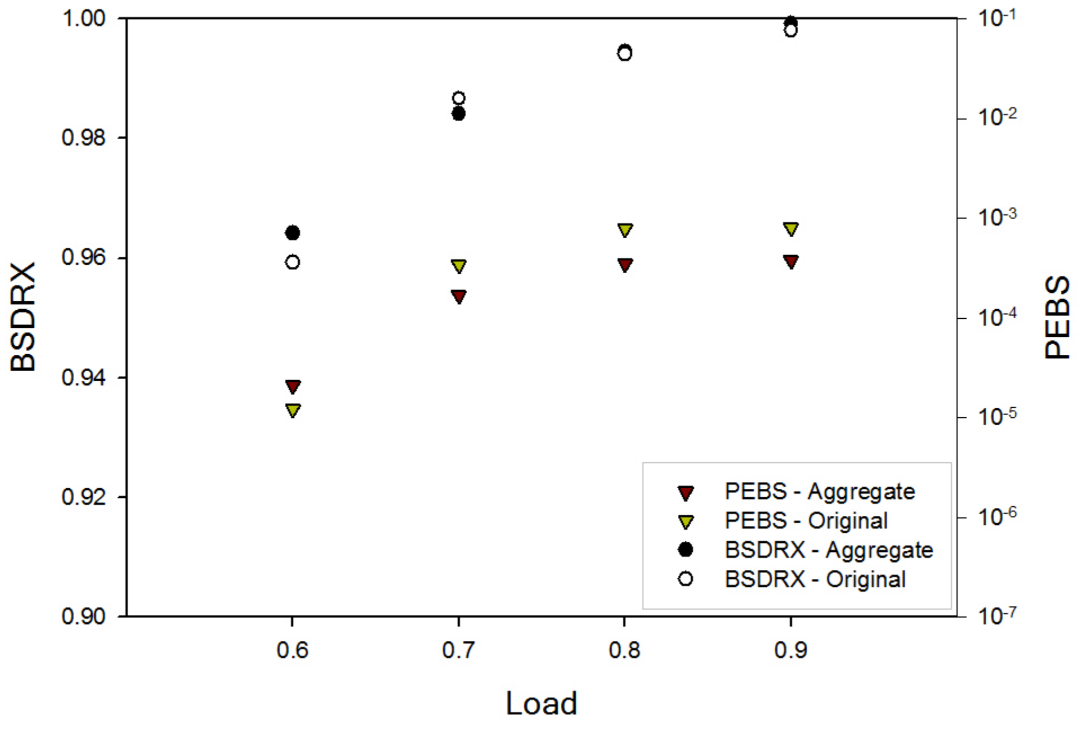


Figure 5-7 Comparison of PEBS and BSDRX: VoIP50

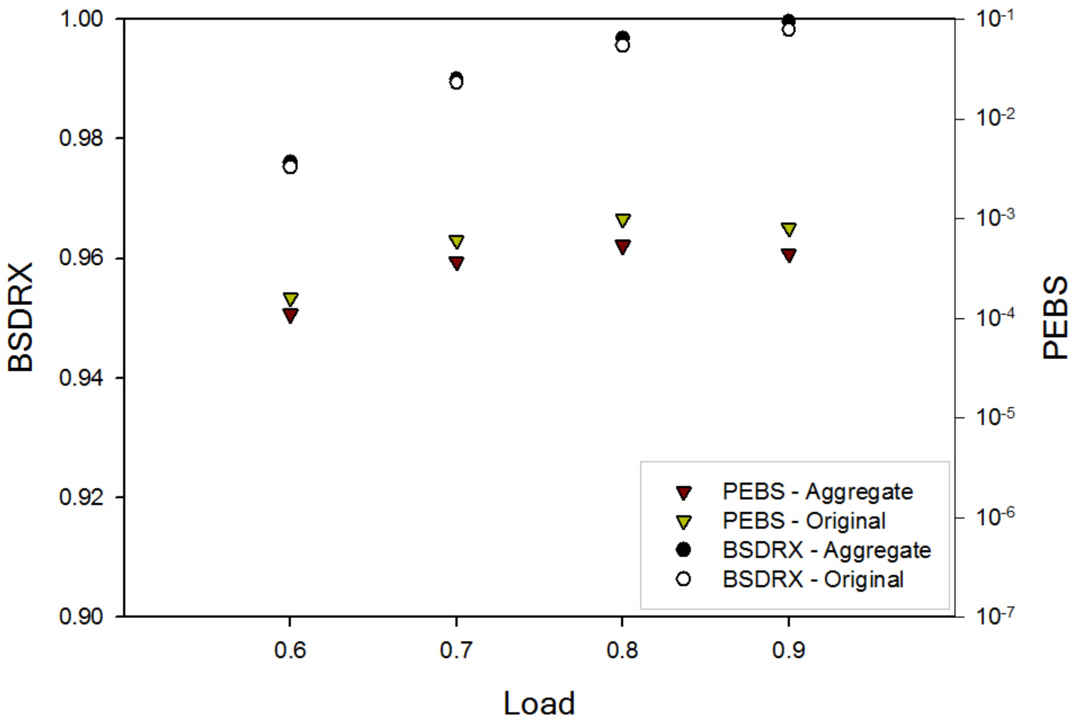


Figure 5-8 Comparison of PEBS and BSDRX: VoIP30

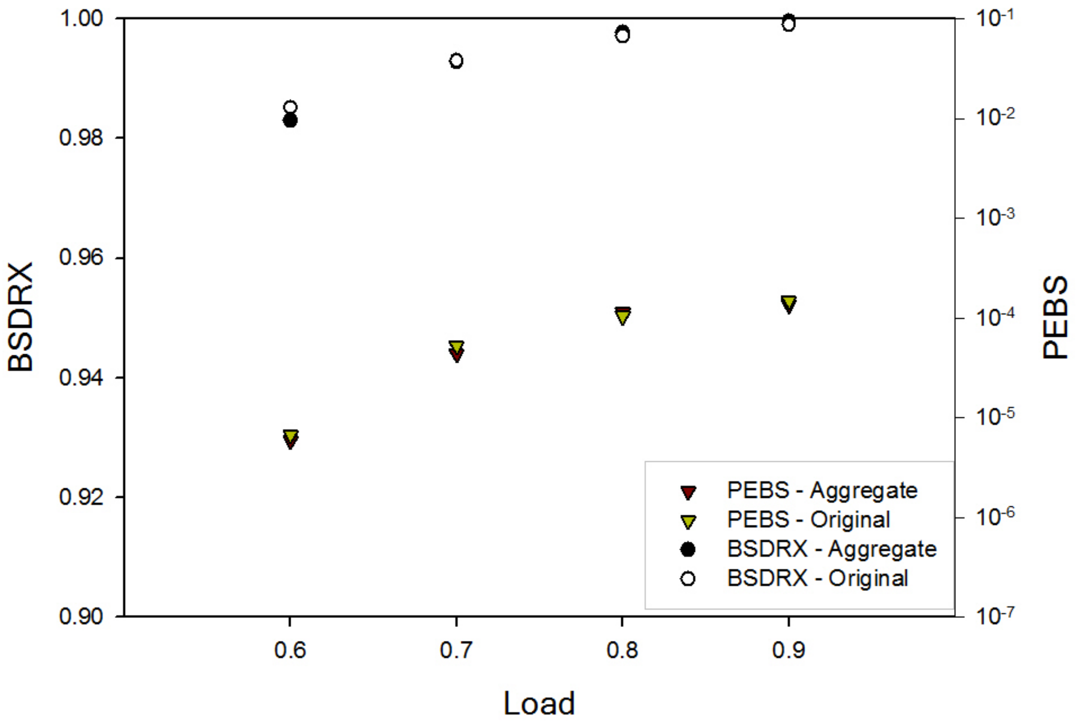


Figure 5-9 Comparison of PEBS and BSDRX: Ton*2.5

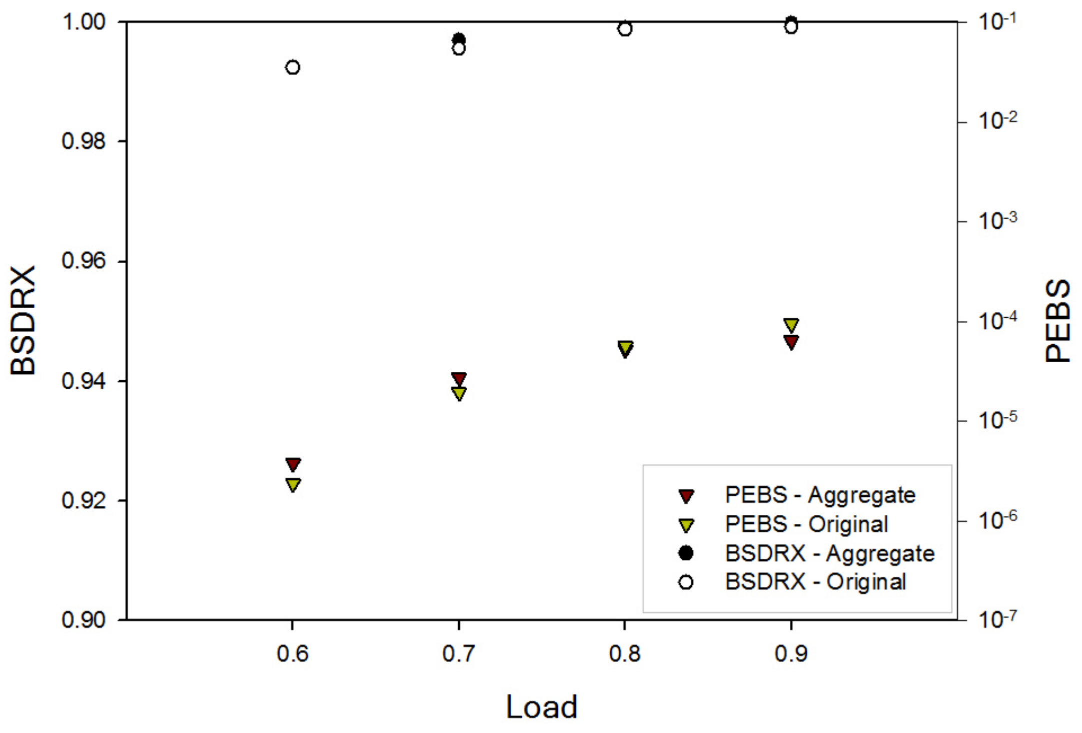


Figure 5-10 Comparison of PEBS and BSDRX: Ton*5

It is apparent that the burst-scale decay rate (BSDRX) in the original model with GPS can be successfully approximated by the aggregation techniques which were partly a result of the scheduler-invariant property of the burst-scale. This aggregate model can be used to develop AS models, enabling fast simulation. See section 5.7 for details.

The error introduced by the aggregation technique (i.e the differences between the original and aggregate BSDR and PEBS) can be considered very small because the practical implications of this error is quite insignificant. This can be quantified using the following buffer dimensioning example.

Service providers typically need to dimension buffers such that the desired buffer overflow probability ($P_{overflow}$) is maintained. From Eq 5.2 and Eq 5.6,

$$P_{overflow} = P_B \cdot R^{x_{overflow}}$$

$$\text{i.e. } x_{overflow} = \frac{\ln(P_{overlfow}) - \ln(P_B)}{\ln(R)}$$

Using VoIP50 (0.8) as an example, the traffic parameters are:

T_{on}=0.96s, T_{off}=1.69s, h=170, N=50, ρ=0.8

The default G.729 codec requires packet loss to be far less than 1% to avoid audible error [79, 80]. Take $P_{overflow} = 0.01$ as an upper bound.

The original simulation produces $R = 0.9941$ and $P_B = 0.1325$, giving $x_{overflow} = 436.73$, while the aggregation technique produces $R = 0.9951$ and $P_B = 0.0710$, giving $x_{overflow} = 399.1$. The difference is $\approx$ 37 packets (or 8.6%).

Figure 5-11 shows the variation of BSDRX across all the scenarios while also comparing it with solutions given in both [31] (which is the one made use of in this thesis) and [76]: the latter is clearly not a good approximation, while the former is excellent, with the accuracy maintained over a wide range of input parameters (varied loads, ON-periods, arrival rates etc.).

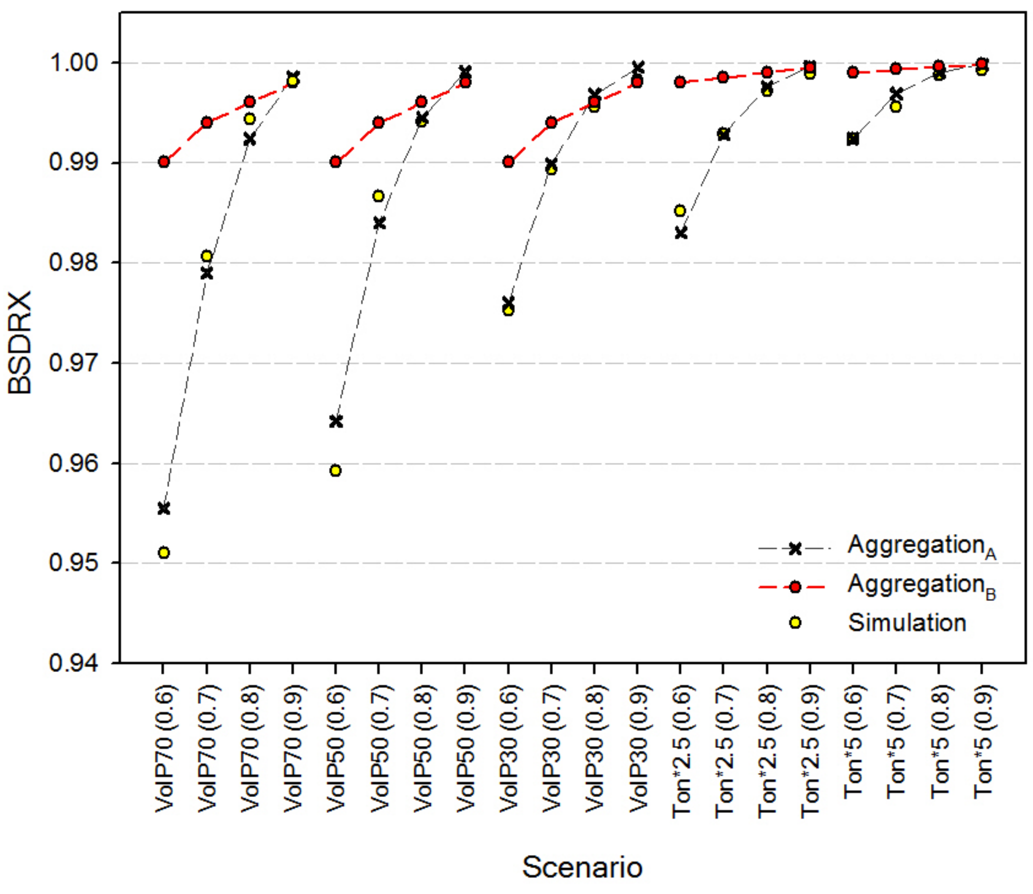


Figure 5-11 Comparison of BSDRX
(Aggregation$_A$: Ref. [31], Aggregation$_B$: Ref. [76])

5.6 *System Time (ST) : bursty traffic scheduled by GPS*

5.6.1 Introduction

In the presence of bursty traffic, the System Time (ST) distribution also consists of the two parts packet-scale and burst-scale [81]. In section 5.6.2, the ST distribution for an arbitrary packet in the GPS model is compared with the FIFO equivalents, demonstrating burst-scale scheduler-invariance (similar to the distribution of X). However, unlike for X, aggregation techniques for the Burst-Scale Decay Rate of ST (BSDRS) of FIFO (or GPS) do not exist in the literature so far. Therefore, a novel technique for the evaluation of BSDRS is derived (in section 5.6.2, which is validated by comparison with simulations of the original model in section 5.6.3. This section also discusses the effects of varying input parameters, on decay rates and PEBS.

5.6.2 Novel Aggregation technique for ST

Statistics of ST were also collected for the simulations listed in Table 5.4.1. The method used to measure ST is the same as the one described in section 4.5.2.

First, the simulated Packet-Scale Decay Rate of ST (PSDRS) values were compared (for validation) with those derived from Eq 4.6 (i.e. ref. [70]) which gives the theoretical ST distribution for an M/D/1 PS system. Figure 5-13 shows that the packet-scale behaviour coincides with M/D/1 PS.

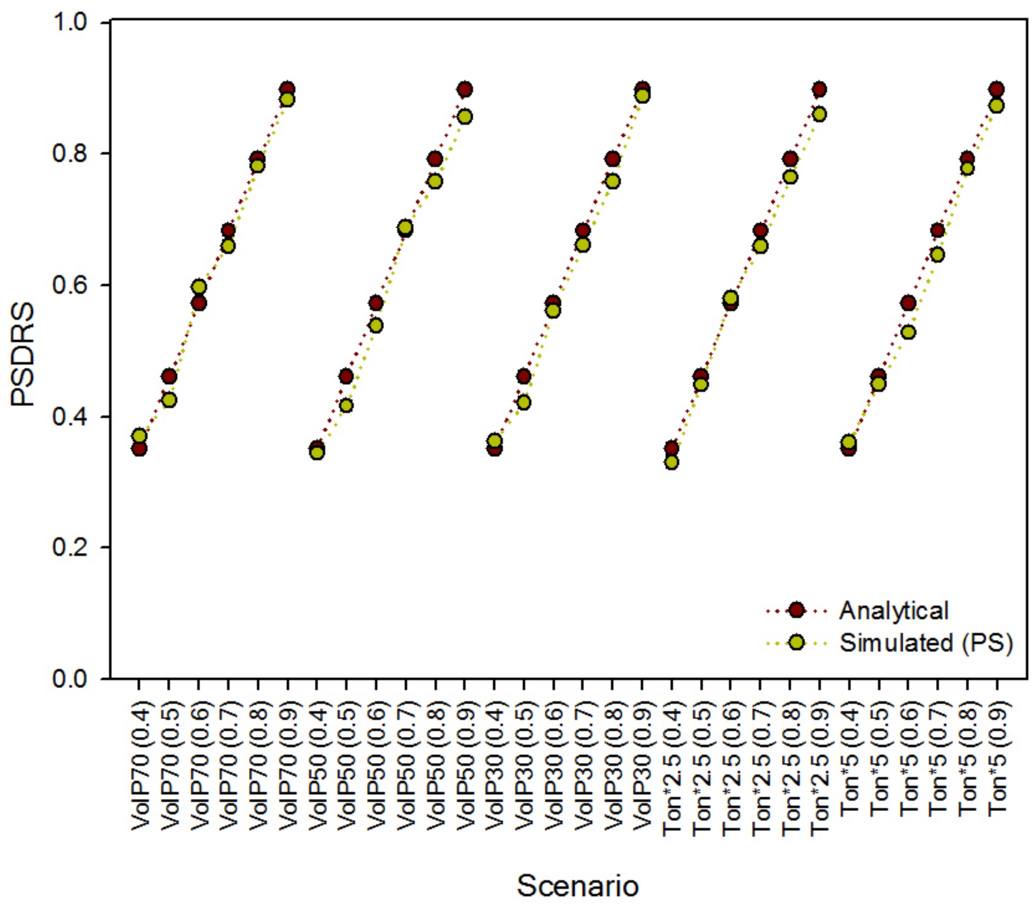


Figure 5-12 Comparison of PSDRS – Simulated and theoretical

Next, the entire simulated ST distribution is considered, again in comparison with the corresponding FIFO cases. The same interesting phenomenon observed with X is seen i.e. the burst-scale behaviour of FIFO and GPS coincide (see Figure 5-13 for the results from the two load values of the VoIP70 scenario[1]). Figure 5-14 illustrates how they start following the same path at the entry point to burst-scale.

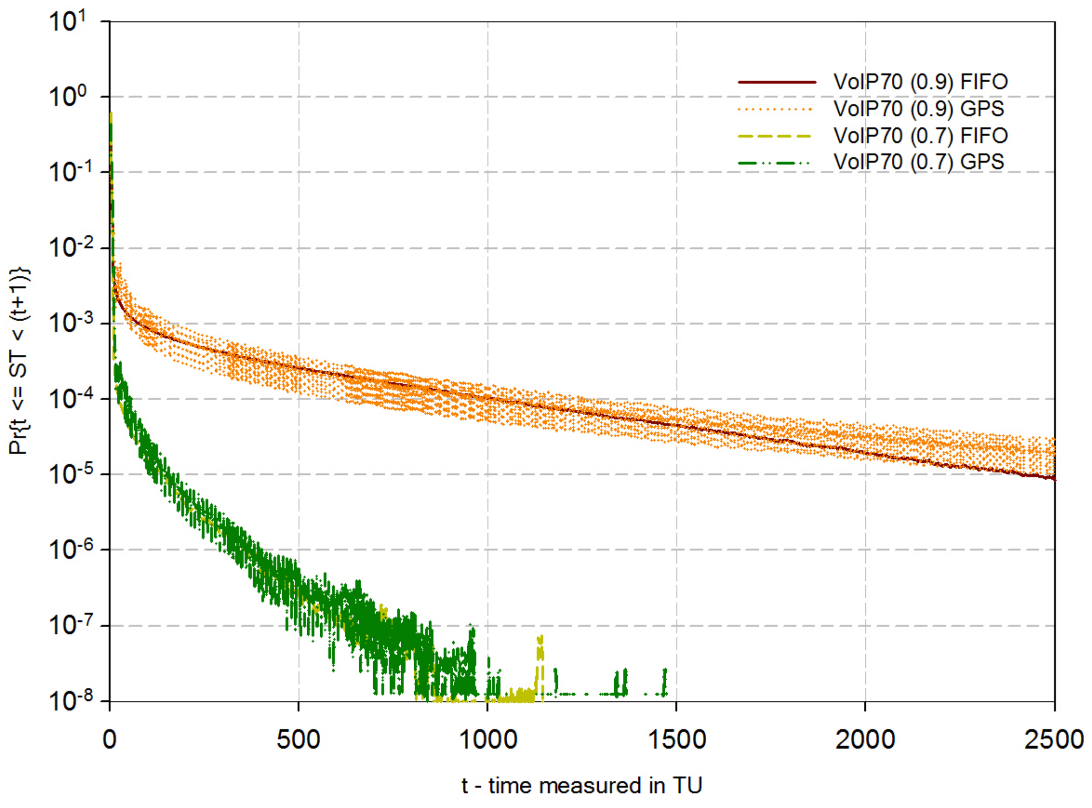


Figure 5-13 Distribution of ST: GPS and FIFO

[1] Others are left out in order to keep the figure simple; however, Table 5.6.1 presents a complete set of results in a processed numerical form.

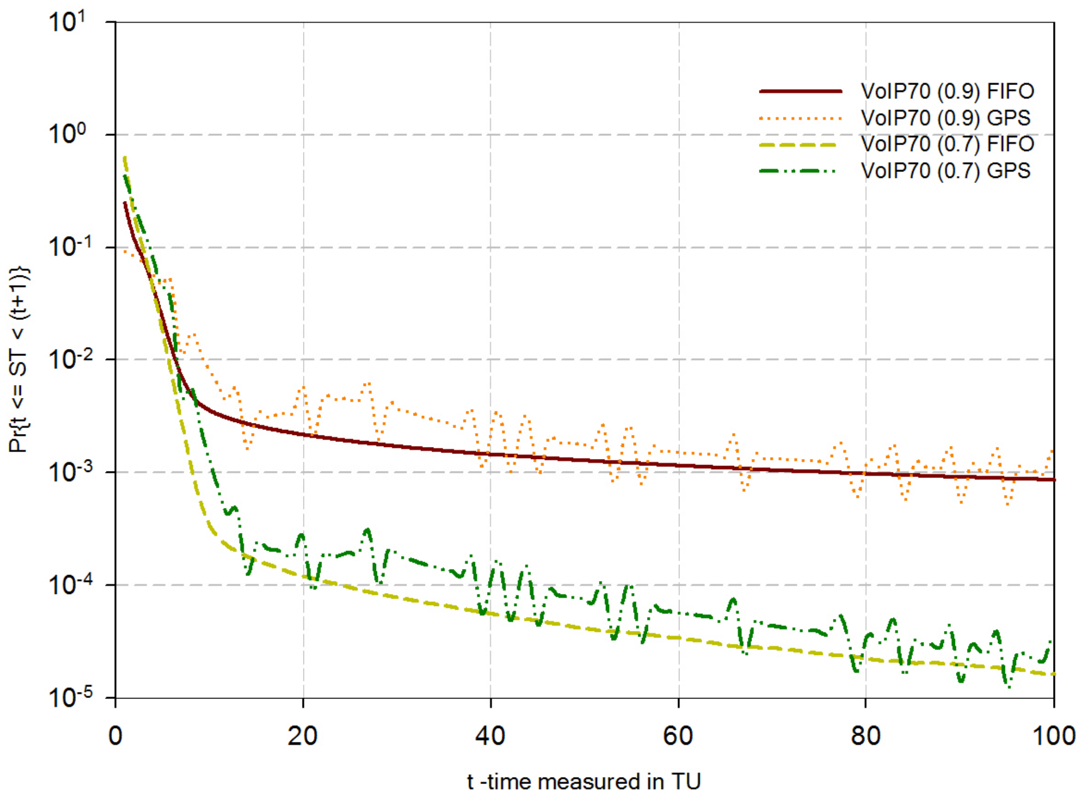


Figure 5-14 Distribution of ST: GPS and FIFO (knee-point region)

Table 5.6.1 which compares BSDRS of GPS and FIFO for all scenarios, justifies the observations made in the graphs.

Table 5.6.1 Comparison of BSDRS - GPS & FIFO (simulated)

Scenario	BSDRS		Scenario	BSDRS	
	GPS	FIFO		GPS	FIFO
VoIP70 (0.6)	0.9608	0.9583	VoIP30 (0.8)	0.9966	0.9966
VoIP70 (0.7)	0.9866	0.9807	VoIP30 (0.9)	0.9988	0.9988
VoIP70 (0.8)	0.9945	0.9943	Ton*2.5 (0.6)	0.9824	0.9768
VoIP70 (0.9)	0.9979	0.9981	Ton*2.5 (0.7)	0.9955	0.9933
VoIP50 (0.6)	0.9619	0.9789	Ton*2.5 (0.8)	0.9972	0.9975
VoIP50 (0.7)	0.9875	0.9875	Ton*2.5 (0.9)	0.9996	0.9989
VoIP50 (0.8)	0.9950	0.9950	Ton*5 (0.6)	0.9938	0.9946
VoIP50 (0.9)	0.9983	0.9983	Ton*5 (0.7)	0.9970	0.9971
VoIP30 (0.6)	0.9808	0.9808	Ton*5 (0.8)	0.9992	0.9988
VoIP30 (0.7)	0.9608	0.9583	Ton*5 (0.9)	0.9994	0.9992

This result is also (similar to X) new to the literature. Further, unlike the observation of X, the scheduler-invariant property of the burst-scale ST would not normally be expected, as the GPS scheduler serves packets in an entirely different manner to that of FIFO. However, as shown by the results, the burst-scale (unlike the packet-scale) ST is not affected by the GPS scheduling mechanism. This result is of significant value as the burst-scale ST distribution of a GPS system could now be obtained by simulating its FIFO equivalent, or an aggregate model. However, aggregate techniques to evaluate the burst-scale distribution of ST, do not exist in the literature so far. A novel approach is developed here.

The following terms are defined:

f(t) – $\Pr\{t \le ST < (t+1)\}$, $1 \le t \le \infty$ [1]

t_k – ST at knee-point

v – PSDRS i.e. $\dfrac{f(t+1)}{f(t)}\ \mathit{for}\ t < t_k$

V – BSDRS i.e. $\dfrac{f(t+1)}{f(t)}\ \mathit{for}\ t \ge t_k$

P_{BS} – Probability of experiencing burst-scale ST

Little's result states the following is true for any work-conserving queueing system [82] :

$$E[X] = \lambda \cdot E[ST] \qquad \text{Eq 5.8}$$

$E[X]$, the average number in the system can be broken down to two parts: the average number in the packet-scale component and the average number in the burst-scale component, denoted by x_{ps} and x_{bs} respectively.

$$E[X] = \sum_{x=0}^{\infty} x \cdot p(x) = \sum_{x=0}^{x_k} x \cdot p(x) + \sum_{x=x_k+1}^{\infty} x \cdot p(x)$$

where $x_{ps} = \sum_{x=0}^{x_k} x \cdot p(x)$ and $x_{bs} = \sum_{x=x_k+1}^{\infty} x \cdot p(x)$.

[1] *t=1* would indicate zero-waiting time.

i.e. $E[X] = x_{ps} + x_{bs}$ Eq 5.9

Similarly, $E[ST] = t_{ps} + t_{bs}$ Eq 5.10

where t_{ps} and t_{bs} are the average ST of packet-scale and burst-scale respectively[1].

Since Little's result is true for any work-conserving queue, it must hold true for the Poisson traffic case, which coincides with the packet-scale component of bursty traffic. Therefore,

$x_{ps} = \lambda \cdot t_{ps}$ Eq 5.11

From Eq 5. to Eq 5.11,

$x_{bs} = \lambda \cdot t_{bs}$ Eq 5.12

Expressed in terms of a geometric progression;

$$x_{ps} = \sum_{x=0}^{x_k} x \cdot p[0] \cdot r^x \approx \sum_{x=0}^{\infty} x \cdot p[0] \cdot r^x \approx \frac{p[0] \cdot r}{(1-r)^2} \qquad \text{Eq 5.13}$$

$\because \sum_{x=x_k+1}^{\infty} x \cdot p[0] \cdot r^x \approx 0$ (see Figure 5-15)

$$x_{bs} = \sum_{x=x_k+1}^{\infty} x \cdot PEBS \cdot R^x \approx \sum_{x=0}^{\infty} x \cdot PEBS \cdot R^x \approx \frac{PEBS \cdot R}{(1-R)^2} \qquad \text{Eq 5.14}$$

$\because \sum_{x=0}^{x_k} x \cdot PEBS \cdot R^x \approx 0$ (see Figure 5-15)

[1] Note that t_{ps} and t_{bs} should ideally be calculated based on $(t + 0.5) \cdot f(t)$ rather than $t \cdot f(t)$ (since $f(t)$ represents $t \le ST < (t+1)$); however, using $t \cdot f(t)$ greatly simplifies the calculations and the difference is negligible.

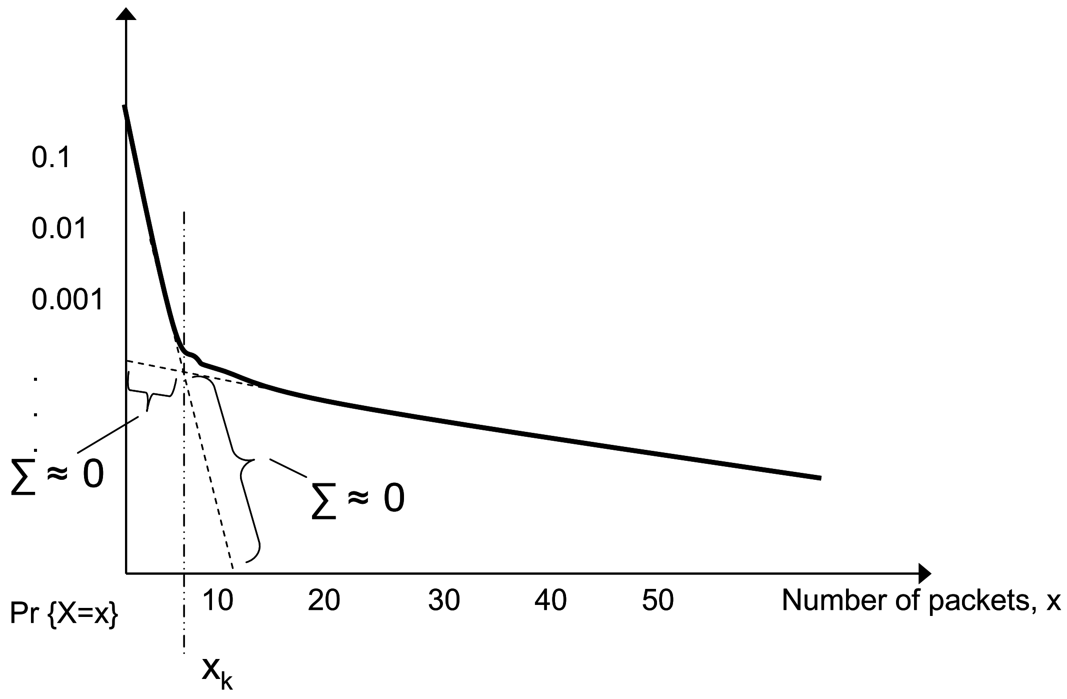


Figure 5-15 Distribution of X for a queue fed by a bursty source

For both X and ST, the probability of entering the burst-scale would be approximately the same. Therefore, similar to Eq 5.14,

$$t_{bs} \approx \frac{PEBS \cdot V}{(1-V)^2}$$

Therefore from Eq 5.12;

$$\frac{V}{(1-V)^2} \cdot \lambda = \frac{R}{(1-R)^2}$$

$$V = \frac{[2 + \lambda \cdot (1/R - 2 + R)] - \sqrt{[2 + \lambda .(1/R - 2 + R)]^2 - 4}}{2} \qquad \text{Eq 5.15}$$

Further, the knee point in the distribution of ST can be derived by following similar reasoning as with X (i.e. Eq 5.2),

$$f(t_k) = P_{BS} \cdot (1-V)$$

$$t_k = \frac{\ln(P_{BS} \cdot (1-V) / f(1))}{\ln(v)} \qquad \text{Eq 5.16}$$

Note that the distribution starts from *f(1)* as ST must be at least 1.

f(1) and *v* could be derived from Eq 4.6.

Further, similar to Eq 5.7,

$$P_{BS} \approx \frac{PEBS}{(1 - V)} \qquad \text{Eq 5.17}$$

5.6.3 Results and Discussion

Figure 5-16 illustrates examples of the comparison between burst-scale ST of the original and aggregate models where the latter is derived based on Eq 5.6 and Eq 5.15 (PEBS and BSDRS respectively). Figure 5-17 to Figure 5-21 compare BSDRS and PEBS for all scenarios which demonstrate excellent agreement between the original and aggregate models.

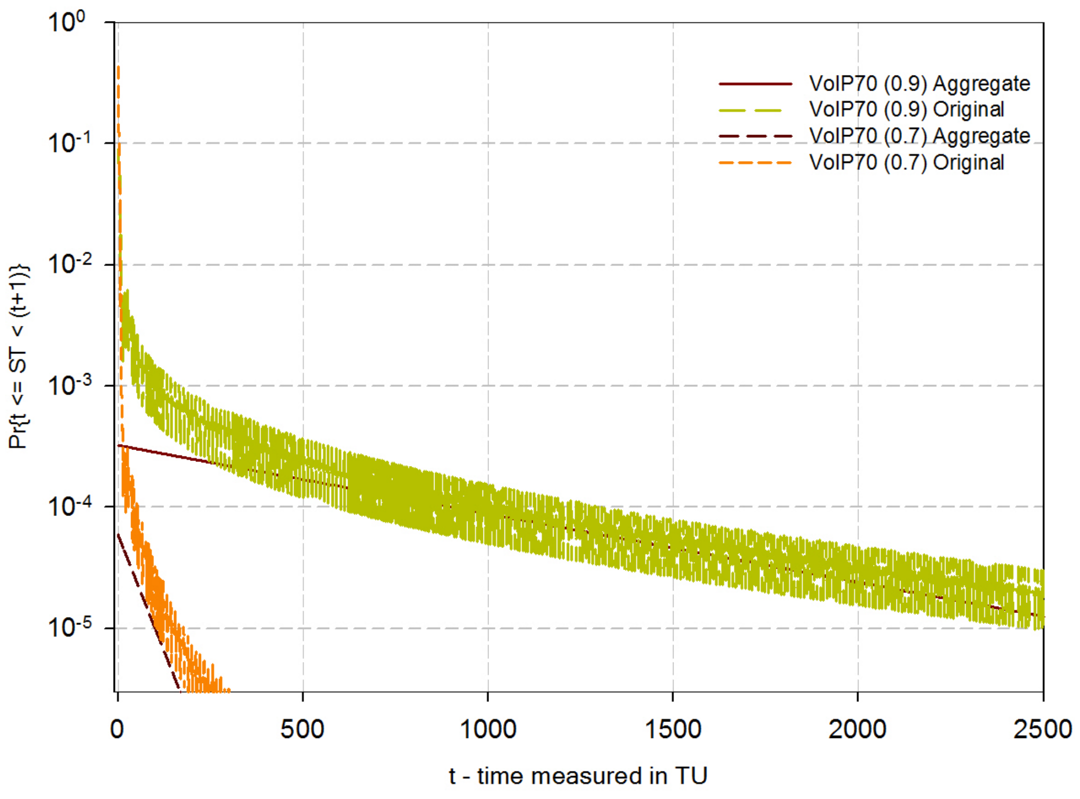


Figure 5-16 Validation of aggregate model: System Time (ST)

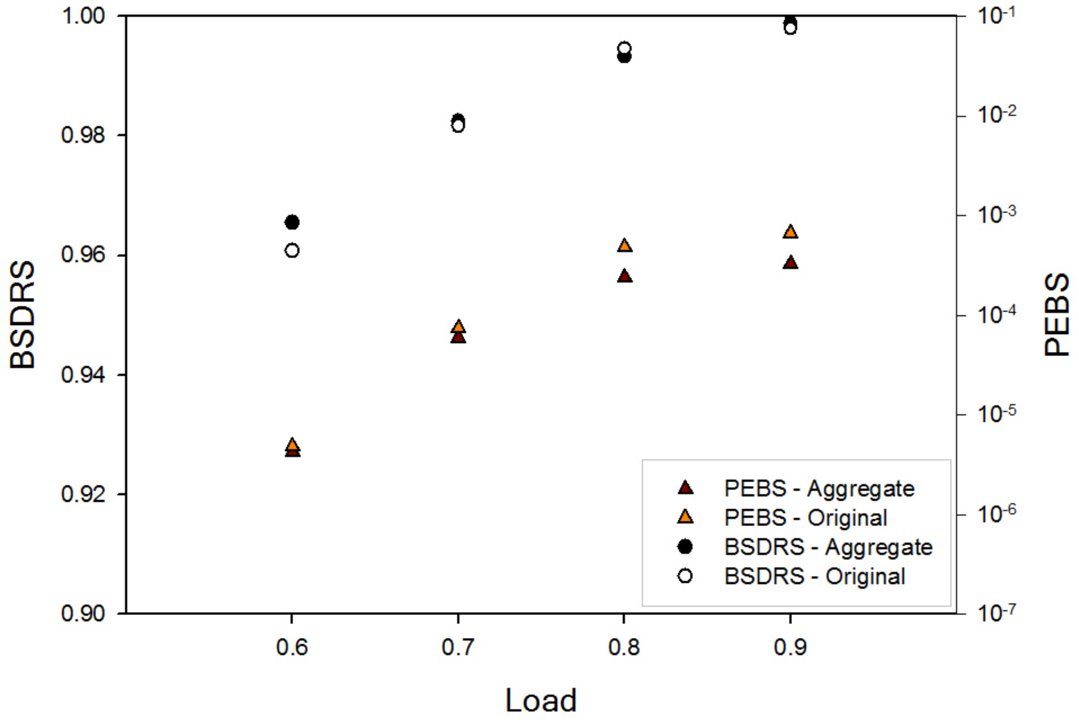


Figure 5-17 Comparison of PEBS and BSDRS: VoIP70

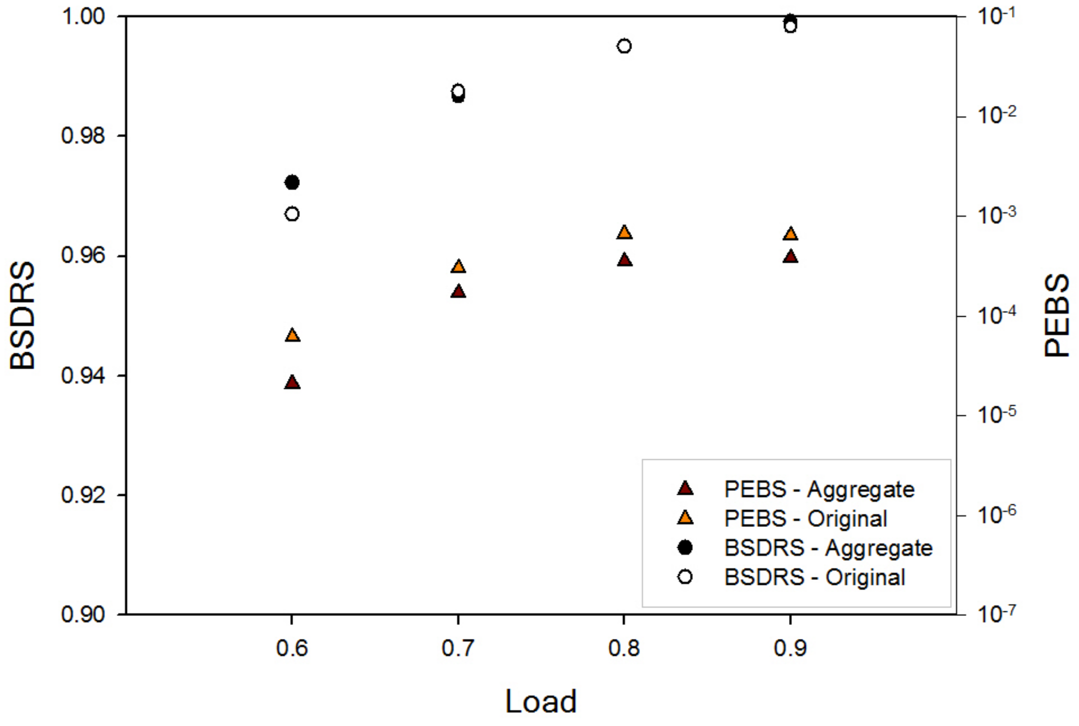


Figure 5-18 Comparison of PEBS and BSDRS: VoIP50

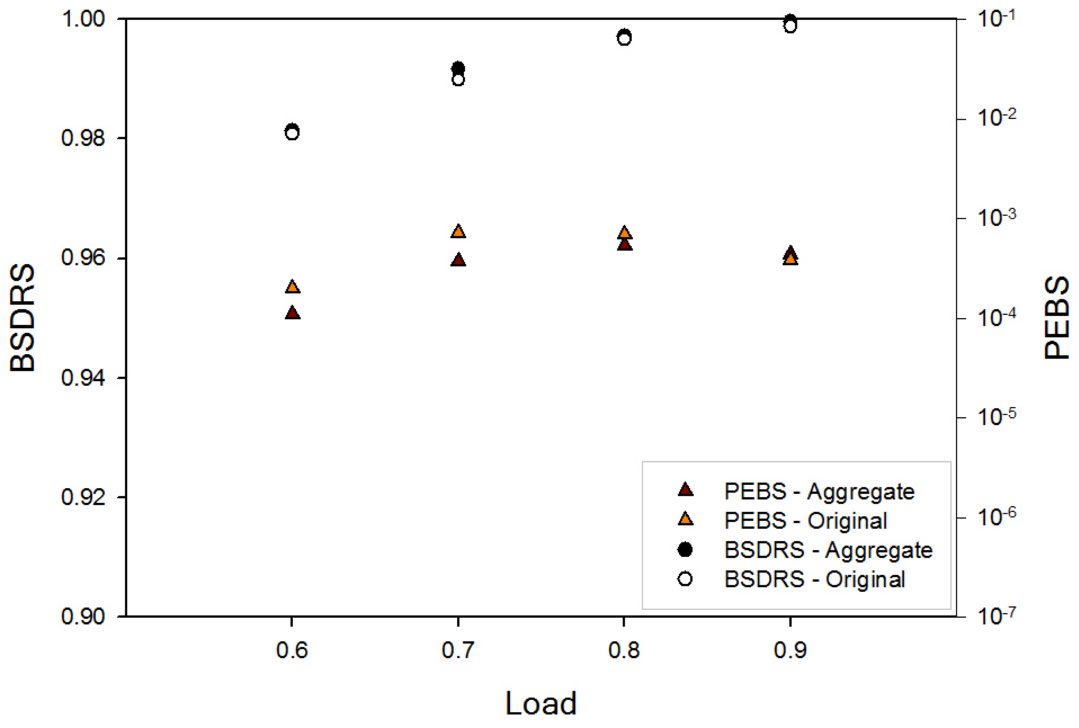


Figure 5-19 Comparison of PEBS and BSDRS: VoIP30

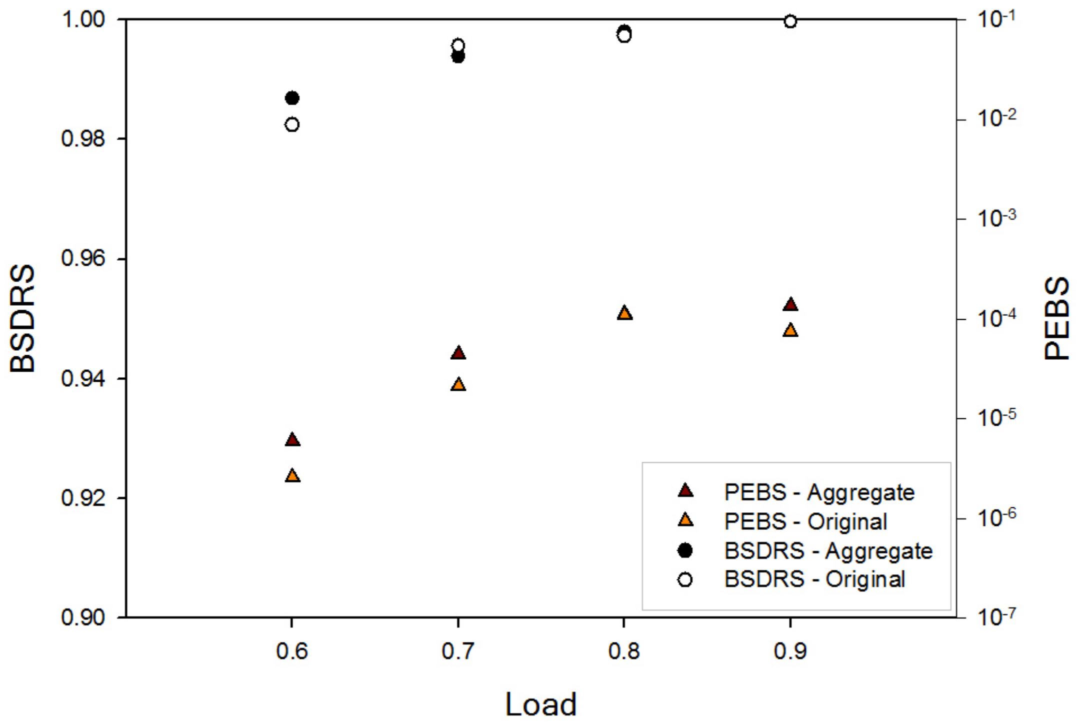


Figure 5-20 Comparison of PEBS and BSDRS: Ton*2.5

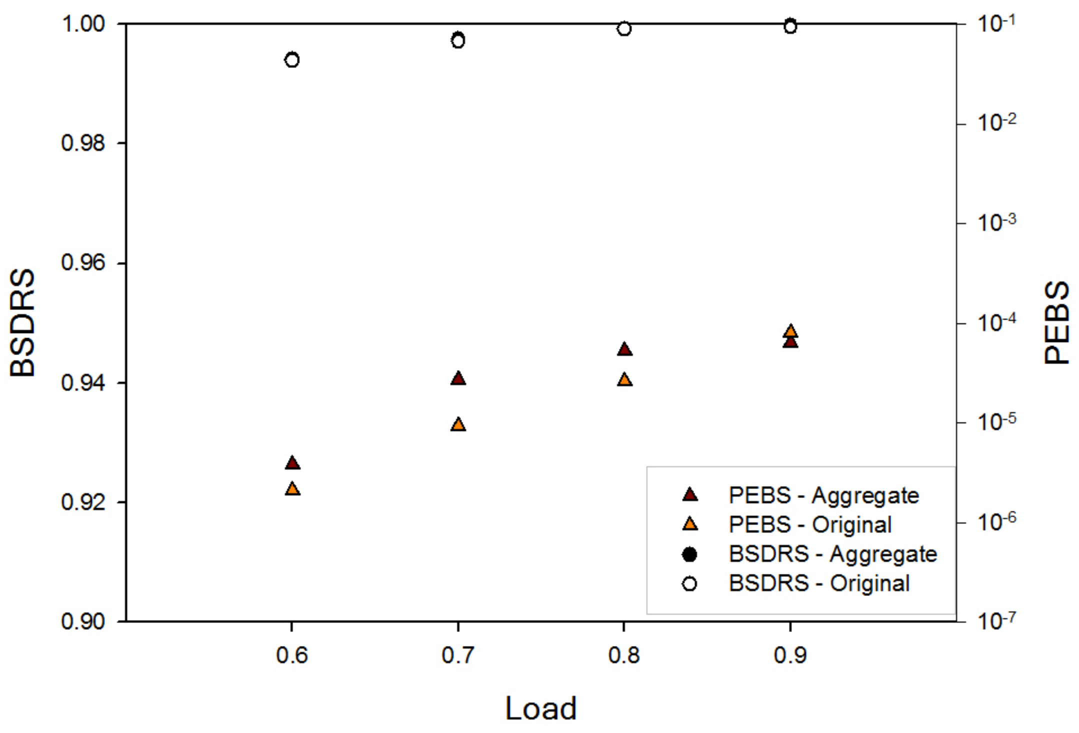


Figure 5-21 Comparison of PEBS and BSDRS: Ton*5

The results show that the burst scale ST distribution can be accurately approximated by the novel aggregation technique, which was derived based on the scheduler-invariant property of the burst-scale and the techniques developed for the burst scale X in section 5.5.2. This is a novel and accurate approximation with minimal complexity. Section 5.7 describes the significance of applying these result to develop AS models, for the performance evaluation of packet networks.

Figure 5-22 shows the variation of BSDRS across all the scenarios, which is similar to the variation of BSDRX.

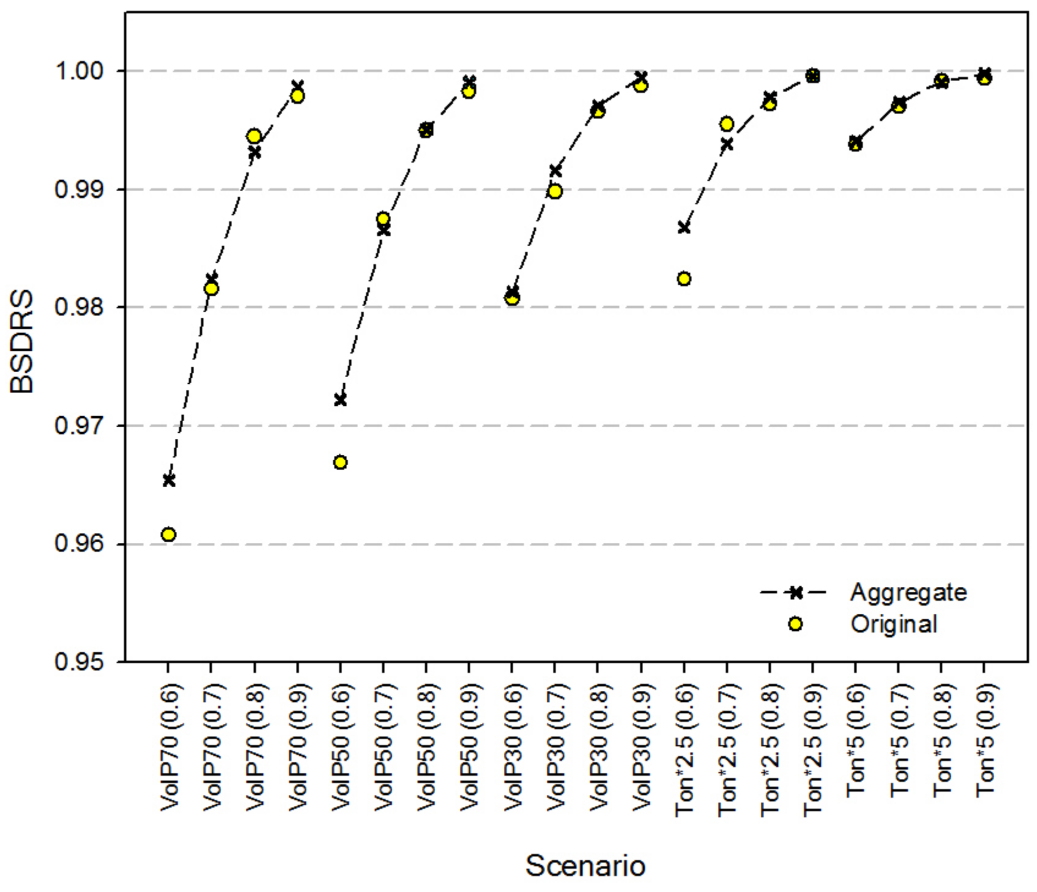


Figure 5-22 Comparison of BSDRS – GPS

Figure 5-11 and Figure 5-22 clearly illustrate the effects on BSDRX and BSDRS respectively (commonly referred to as BSDR), caused by varying the input parameters. For example, increasing *h* and T_{on} (i.e. from VoIP70 to Ton*2.5 to Ton*5) under the same load increases BSDR which means the distributions decay slower. Increasing *N* (i.e. from VoIP30 to VoIP50 to VoIP70) under the same load, on the other hand, decreases BSDR and this is known as multiplexing gain (using more sources makes the distribution decay faster, reducing the tail). Further, decreasing the load decreases BSDR (also PEBS), reducing the burst-scale component (and increasing the packet-scale component). Due to this, the simulated number of events occurring in the burst-scale at low loads is usually too few to be used in any statistical analysis, e.g. PEBS values for the cases with utilisations 0.4 and 0.5 of the scenarios listed in Table 5.4.1 range from 10^{-7} to 10^{-13}; hence they result in very little burst-scale events in a standard (i.e. non-accelerated) simulation that finishes within a practical time limit.

5.7 *Application to Accelerated Simulation: speed-up factor*

Similar to Chapter 4, the aggregate model introduced in this section also consists of a single flow (i.e. equivalent ON-OFF source) instead of *N* flows: this means (similar to Chapter 4), the aggregate model has *N* times less events as in the original model, which will give a speed-up factor equal to *N* in an AS model.

In addition, the discovery of the scheduler invariant property also introduces event reduction. This is because using a FIFO scheduler results in the same burst-scale behaviour as GPS which means a standard ON-OFF source can be used instead of ExpoOn, which will reduce the number of events by (at least – see below) a factor of 10 (i.e. small packets per large packet). In order to demonstrates the significance of this event reduction, consider the scenario 'Ton*2.5' as an example. Here,

Mean sending rate of a single source	*= 2.4 * 200 / (2.4 + 5.39) packets/sec*
	= 61.62 packets/sec
Mean sending rate from all 70 sources	*= 4313.22 packets/sec*

Hence, the number of events in a simulation of length 5000 sec will be $2.16*10^7$, while the total number of events in 100 such (randomised) simulations will be $2.16*10^9$. Results presented in this chapter facilitate the reduction of number of events to $2.16*10^8$ with the use of the standard ON-OFF source which will substantially reduce the wall-clock time taken by an event-driven simulator. Further, simulating with a FIFO scheduler rather than WRR would generally further speed-up the simulation (the exact amount will depend on the particular simulator).

5.8 *Further validation of results*

In addition to the validations presented throughout the chapter, the relationship between the simulated X and ST is validated here by employing Little's result.

The simulated $E[X]$ and $E[ST]$ are determined based on extrapolated best-fit graphs. This is more accurate than using the original graphs for the following reason. It is apparent that the simulated distributions deviate from the expected pattern (sometimes called 'noise') beyond a certain point (referred to as 'cut-off point' from here onwards). This is due to the low probabilities of the events concerned resulting in the number of events being inadequate to give unbiased probabilities: this is normal with stochastic simulations. 'Noisy' values are avoided by taking the extrapolated best-fit graphs rather than the original one (see Figure 5-23). Further, had it been feasible to run the simulations long enough, those simulated distributions would have indeed coincided with the extrapolated best-fit graphs. Moreover, the values prior to the cut-off point are also smoothed out by employing best-fit graphs rather than the original ones. Therefore, the packet-scale average is also determined similarly (i.e. best-fit rather than original). Figure 5-23 shows how the extrapolated best-fit graphs are employed for the calculation of $E[X]$ in the VoIP70 (0.8) scenario as an example.

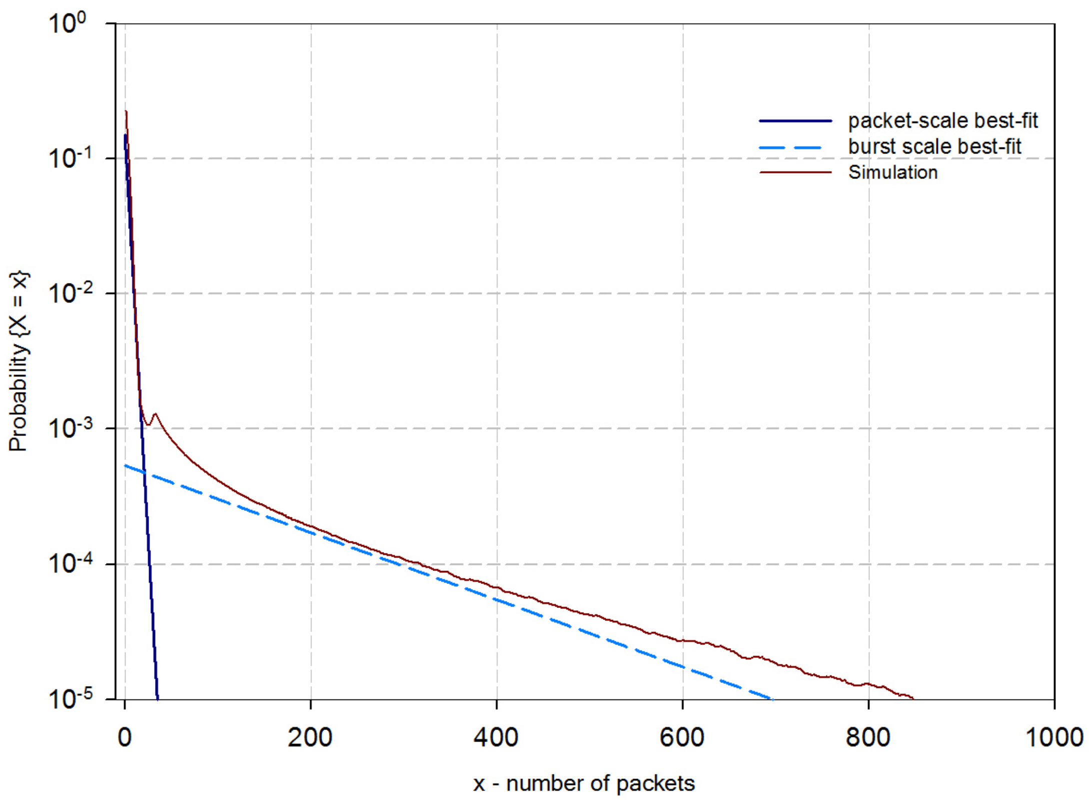


Figure 5-23 Use of best-fit graphs: VoIP70 (0.8)

Once the simulated $p(0)$, PEBS, r and R are determined by the best-fit graphs (of the form $y=a.e^{(b.x)}$), x_{ps} and x_{bs} are calculated using Eq 5.13 and Eq 5.14 respectively. t_{ps} and t_{bs} are calculated similarly (using $f(1)$, PEBS, v and V), which are then used to calculate $E[X]$ and $E[ST]$ (Eq 5.9 and Eq 5.10 respectively) . Finally, simulated $E[X]/E[ST]$ is compared with theoretical λ (see Table 5.8.1). Note that $\lambda = \rho$ as the number of packets served per TU is 1.

Table 5.8.1 Validation of simulated E[X] and E[ST] against Little's result

Scenario	λ		Scenario	λ	
	$\lambda = \frac{E[X]}{E[ST]}$	Theory		$\lambda = \frac{E[X]}{E[ST]}$	Theory
VoIP70 (0.6)	0.6139	0.6	VoIP30 (0.8)	0.7979	0.8
VoIP70 (0.7)	0.7214	0.7	VoIP30 (0.9)	0.8983	0.9
VoIP70 (0.8)	0.7926	0.8	Ton*2.5 (0.6)	0.6100	0.6
VoIP70 (0.9)	0.8730	0.9	Ton*2.5 (0.7)	0.6794	0.7
VoIP50 (0.6)	0.6334	0.6	Ton*2.5 (0.8)	0.8248	0.8
VoIP50 (0.7)	0.7212	0.7	Ton*2.5 (0.9)	0.8993	0.9
VoIP50 (0.8)	0.8298	0.8	Ton*5 (0.6)	0.5825	0.6
VoIP50 (0.9)	0.8966	0.9	Ton*5 (0.7)	0.6801	0.7
VoIP30 (0.6)	0.6263	0.6	Ton*5 (0.8)	0.8121	0.8
VoIP30 (0.7)	0.7029	0.7	Ton*5 (0.9)	0.8800	0.9

Table 5.8.1 shows that the simulations presented in this chapter are in agreement with Little's result.

5.9 Summary

The burst-scale behaviour of multiplexed homogeneous ON-OFF sources is analysed in this chapter, and it is discovered that the burst-scale 'number in the system' (X) and 'system time' (ST) are scheduler-invariant over FIFO and GPS. This is a significant result as this eliminates the need to carry out GPS simulations for bursty sources, which are complex, as the much simpler FIFO equivalent could provide the same results. Section 5.7 gives an example of the acceleration which can be achieved as a result.

The discovery of the scheduler-invariant property also means that for the evaluation of X, aggregation techniques developed for FIFO could be employed to evaluate the performance with a GPS scheduler, as shown in section 5.5.2, e.g. a single equivalent ON-OFF source instead of *N* sources, which as shown in Figure 5-11 is an excellent approximation of the original model. This result leads to event reduction in simulations, i.e. AS models, as N multiplexed ON-Off sources could be accurately replaced by a single equivalent ON-OFF source.

Based on the techniques developed for X, section 5.6.2 has developed a novel formula for the burst-scale decay rate of the ST distribution which is shown to be simple and accurate over a wide range of input parameters (see Figure 5-22), in the evaluation of burst-scale ST of ON-OFF traffic scheduled by GPS.

Finally, in Table 5.8.1, Little's result is used to carry out an overall validation process for this chapter, where the ratio of simulated $E[X]/E[ST]$ is shown to closely approximate (theoretical) λ as stated in Little's result, for all presented scenarios.

CHAPTER 6

WEIGHTED FAIR QUEUEING

6.1 Introduction

The type of non-FIFO scheduler commonly used in practice is Weighted Fair Queueing (WFQ), as described in section 3.4.2.2. Employing WFQ schedulers, this chapter briefly considers in section 6.3, models with homogeneous traffic classes (similar to Chapter 5). Performance evaluation with more realistic models which involve heterogeneous traffic classes are then presented in detail in section 6.4, where a novel aggregation technique to determine the burst-scale System Time (ST) distribution of a class of interest is developed. This section also explains how the developed techniques can be used to develop Accelerated Simulation (AS) models. First, section 6.2 describes with an example how the WFQ scheduler operates, in comparison with GPS and standard WRR.

6.2 Scheduling schemes: GPS, WFQ and WRR

It is important to identify the WFQ scheduling discipline distinctively from GPS and WRR. The detailed descriptions given in Chapter 3 can be summarised as follows. GPS serves an infinitesimally small amount from each class during a cycle, which cannot be achieved practically. WFQ algorithms are packet-based; however, they are able to approximate GPS by serving packets in the same order in which they would finish service in the ideal GPS scheme, e.g. when equal weights are given to all classes, WFQ will approximately serve an equal number of bits from each class irrespective of the packet sizes, constrained only by the fact that packets are not divisible in a practical system. WRR (i.e. standard-WRR – see section 3.4.2.2.2), on the other hand, also packet-based, is not specifically designed to approximate GPS; however, it could act as a rough approximation if the weights are assigned carefully (which only works when the packet sizes of each class are known and non-variable). The following example clearly illustrates the difference between the three schemes, by comparison of their order of service.

Consider a queueing system with separate buffers for two heterogeneous traffic classes, Class #1 and Class #2 (Figure 6-1). The traffic fed into Class #1 and Class #2 have packet sizes 3 bytes (packets denoted by A, B, C,.. etc.) and 2 bytes (packets denoted by α, β, γ,.. etc.) respectively. For the purpose of this analysis, assume that packets are broken down into 'slices' of one byte each, e.g. packet A is broken down into A1, A2 and A3, which are used by the GPS server[1]. Assume that the two classes are backlogged as shown.

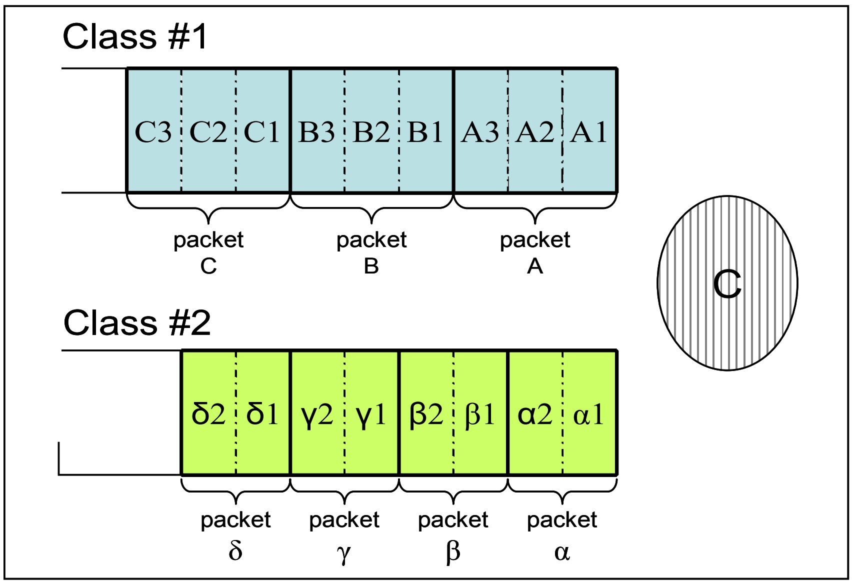


Figure 6-1 Queueing model with two classes of service

Table 6.2.1 gives the order of service if these two buffers were scheduled by each of the three schemes GPS, WFQ and WRR. In GPS and WFQ, equal weights are

[1] For the sake of simplicity of this example, slices can be assumed as one-byte in size without loss of generality. Note that in the ideal GPS scheme, slices would be infinitesimally small in size.

given to the two classes, while in WRR, weights Class #1:Class #2 are set to 2:3 (i.e. reciprocal of packet sizes) in order to approximate GPS (and WFQ).

Table 6.2.1 Comparison of order of service: GPS, WFQ & WRR

Scheme	Order of service													
GPS (1:1)	A1	α1	A2	α2	A3	β1	B1	β2	B2	γ1	B3	γ2	C1	…
WFQ (1:1)	α1	α2	A1	A2	A3	β1	β2	B1	B2	B3	γ1	γ2	δ1	…
WRR (2:3)	A1	A2	A3	B1	B2	B3	α1	α2	β1	β2	γ1	γ2	C1	…

Notice that GPS continues to take a ‘slice’ from each class at a time, while in both WFQ and WRR, the slices belonging to a single packet are always kept together. WFQ selects packets depending on their service finishing order[1] in GPS, i.e. α, A, β, B ..etc. Note that the difference between WFQ and GPS never exceeds the service time of a large packet (unlike WRR, see e.g. packet B). WRR selects an appropriate number of packets from each class, based on the weights assigned. Further, Figure 6-2 demonstrates the fact that an appropriately weighted WRR can approximate WFQ, while an equally weighted WRR cannot, in the case of heterogeneous classes: as shown, the Waiting Time (WT) of Class #1 when scheduled by an equally weighted WRR, is entirely different to those of the other two. Hence, even though WRR is relatively simple to implement, WFQ is of more practical use (than WRR) as it is immune to the effects of variable packet lengths and does not require prior knowledge of packet sizes in order to approximate GPS.

[1] Some WFQ algorithms are based on service starting times, as explained in section 3.4.2.2.1.

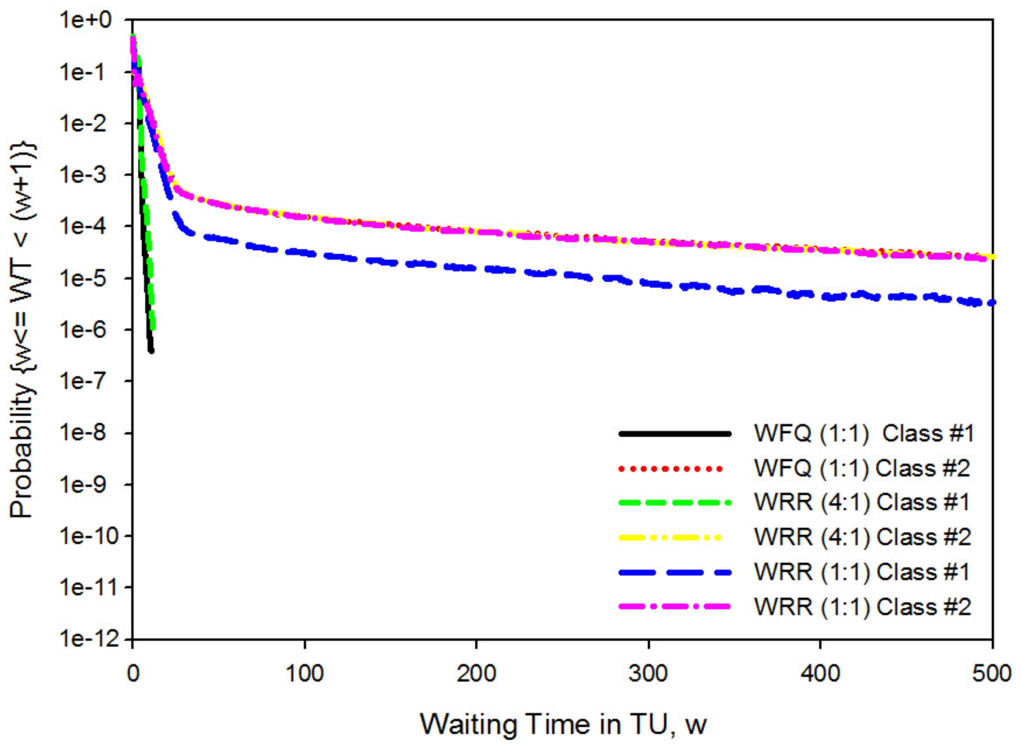


Figure 6-2 Comparison of Waiting Time (TU) with WFQ and WRR (packet-sizes (Class #1):(Class #2) = 1:4)

6.3 Performance evaluatoin for the homogeneous case

The following models were considered:

a) *N* homogeneous classes where each class accepts traffic from a single flow (similar to Chapter 5)

b) 5 homogeneous classes where each class accepts traffic from *N/5* flows.

An equally weighted WFQ scheduler was employed in both cases. The input parameters were the same as those listed in Table 5.4.1.

As would be expected given the scheduler-invariant property seen in Chapter 5 for GPS and FIFO, the burst-scale ST distributions of both models a) and b) coincide with those of GPS, and therefore with FIFO as well (see Chapter 5). Hence, the aggregation techniques developed in Chapter 5 can be used for the performance evaluation in the homogeneous case for WFQ.

6.4 *Performance Evaluation for the heterogeneous case*

6.4.1 Introduction

Evaluating the distributions of ‘Number in the system’ (X) and ‘System Time’ (ST) in heterogeneous classes scheduled by WFQ is challenging. This is because the actual service capacity (or service rate) received by a class cannot be easily determined, due to the following reason. The service rate originally allocated to a class, say Class x is $C \cdot \phi_x$ (where C is the total service capacity and ϕ_x is the weight allocated to Class x), which is indeed the rate received when all classes are backlogged; however, in a work-conserving system, the actual service received could be higher at certain times i.e. when one (or more) of the other classes is (are) empty. If all other classes are empty, then Class x will be served with the full capacity (i.e. *C*) of the server. This is because the (work-conserving) server must always be busy if there are packets in the system. The usage of service capacity (or bandwidth) that was originally allocated to other classes is sometimes referred to as ‘bandwidth stealing’ [78, 83].

As a result of bandwidth stealing, the average service rate received over a long period will not be equal to $C \cdot \phi_x$, but will depend on the amount of bandwidth ‘stolen’ (and ‘lost’, i.e. stolen by other classes), which is a function of the traffic demands of other classes [84]. Ref [78] illustrates with simulation results that predicting the queue-state distribution by assuming the overall per-class service rate to be $C \cdot \phi_x$ is inaccurate. Ref [78] then uses simulation to measure the actual bandwidth received (in certain scenarios), which helps provide a reasonable prediction of the queue-state distribution. Additionally, [84] and [85] have developed the idea of ‘service envelopes’ which basically state that the service rate available to Class x (whenever it is backlogged) is lower bounded by $C \cdot \phi_x$ and upper bounded by C; however they do not give explicit formulae for the exact overall service rate received. This chapter develops a novel and simple aggregation technique (which does not require measurement) to accurately approximate the actual service capacity received by heterogeneous classes (section 6.4.3), which leads to AS models. Section 6.4.4 validates the novel aggregation technique by comparison of the original and aggregate models.

6.4.2 Simulation topology and Model

The topology is the same as in Chapter 5, i.e. incoming traffic flows are routed from the 'Edge' node to the 'Destination' node. The 'original model' consists of *N* heterogeneous traffic classes, Class A, Class B, etc. where Class A is the one which is of interest. As far as Class #A is concerned, the remaining *N-1* classes can be thought of as a single class (e.g. the total excess bandwidth available to Class A could either come from (*N-1*) classes where *N>2*, or a single class). Therefore, for brevity, the original model considered in this chapter consists of two heterogeneous classes Class A and Class B.

The aggregation technique developed in this chapter ensures that the ST experienced by Class A would remain the same even when it is simulated on its own (i.e. without the other classes). Therefore, the 'aggregate model' consists of Class A only.

6.4.3 Novel Aggregation Technique for System Time: heterogeneous classes

The following terms are defined.

λ_x - arrival rate of class x, $x = A, B$

λ - total arrival rate, i.e. $\lambda_A + \lambda_B$

C_x - overall service rate received by class x, $x = A, B$

C - total service rate

ρ_x - utilisation of class x, $x = A, B$, i.e. λ_x / C_x

ρ - total utilisation, i.e. λ / C

ϕ_x - relative weight allocated to class x, $x = A, B$ such that $\sum_x \phi_x = 1$

When both queues are backlogged, the service rate received by each class is equal to $C \cdot \phi_x \quad (x = A, B)$. As explained in section 6.4.1, $C_x \neq C \cdot \phi_x$. Moreover,

C_x depends on the traffic demands of the other class, or more specifically, the probability that the other queue is empty. Define q_x as the probability of queue x being empty, ($x = A, B$). However, finding a solution for q_x is a challenge, because e.g. q_B is a function of ρ_B, therefore C_B, which in turn depends on q_A. Nevertheless, deriving C_x and q_x as described below yields a tidy approximation.

The total of overall service rates received by the two classes is equal to C. i.e.

$$C = C_A + C_B \qquad \text{Eq 6.1}$$

Class A will be served at a rate of C or $C \cdot \phi_A$ depending on whether Class B is empty or not. Therefore,

$$C_A \approx q_B \cdot C + (1 - q_B) \cdot C \cdot \phi_A \qquad \text{Eq 6.2}$$

(C_B is similar.)

An expression for q_x can be derived as follows.

In a work-conserving single class system, the probability of a queue being empty (q) is given by;

$$q = s[o] + s[1]$$

where $s[k], k = 0,1,2,...\infty$ is the probability of k packets in the system, i.e. the queue is empty when the system is empty and when the system has exactly 1 packet (as this packet must be in service).

$s[0] = 1 - \rho$ where ρ is the utilisation of the system.

$s[1] = s[0] \cdot \dfrac{(1 - a[0])}{a[0]}$ where $a[k]$, $k = 0,1,2,...\infty$ is the probability of k arrivals in a Time Unit (TU), where a TU is defined as the service time of a packet.

For a Poisson input distribution;

$a[k] = \dfrac{\lambda^k}{k!} \cdot e^{-\lambda}$ where λ is the number of arrivals per TU [32].

$\therefore a[0] = e^{-\lambda}$

Therefore,

$$q = (1-\rho)\cdot\left(1+\frac{\left(1-e^{-\lambda}\right)}{e^{-\lambda}}\right)$$

$$q = e^{\lambda}\cdot(1-\rho)$$

In a system with two classes A and B, the empty-queue probability is not as simple, e.g. q_A will be a function of $s[0]$, $s[1]$, $s[2]$etc. also of q_B (and vice versa) which makes it impossible to determine either q_A or q_B. However, using the same formula as for the single-class system, with ρ replaced by ρ_A (or ρ_B for class B) provides an excellent approximation (as shown with the comparison at the end of section 6.4.4, in Figure 6-8), i.e.

$$q_x \approx e^{\lambda_x}\cdot(1-\rho_x) \qquad \text{Eq 6.3}$$

Substitution for q_A and q_B in Eq 6.2 and for C_A and C_B in Eq 6.1 results in the following:

$$-\left[e^{\lambda_A}\cdot\phi_A + e^{\lambda_B}\cdot\phi_B\right]\cdot C_A^2 + \left[e^{\lambda_A}\cdot\phi_A\cdot(C+\lambda_A) + e^{\lambda_B}\cdot\phi_A\cdot(C-\lambda_B)\right]\cdot C_A - \left[e^{\lambda_A}\cdot\phi_A\cdot\lambda_A\cdot C\right] = 0 \qquad \text{Eq 6.4}$$

where C_A is the only unknown variable.

The ST distribution of Class A (i.e. aggregate model) can now be evaluated based on the aggregation techniques developed in Chapter 5, i.e. using the same formulae (i.e. Eq 5.15 for the Burst-Scale Decay Rate, BSDRS and Eq 5.6 for the Probability of Entering Burst-Scale, PEBS), with C (total service rate) replaced by C_A.

6.4.4 Results and Discussion

Input parameters used for the simulation experiments are listed in Table 6.4.1. An equal number of traffic sources belong to each class, e.g. in the scenario

‘VoIP70_hetero’, 35 sources belong to each class, giving a total of 70 for the two classes. The sources of both classes generate the same number of packets per TU; however, as shown, the packet sizes are different. The TU is defined as the service time of a Class A packet.

Table 6.4.1 Input parameters

Scenario ID	N	Ton	Toff	h	ψ	Packet size	Packet size
	Class A/B					(Class A)	(Class B)
VoIP100_het	50	0.96	1.69	170	0.3623	20	5
VoIP70_het	35	0.96	1.69	170	0.3623	20	5
VoIP50_het	25	0.96	1.69	170	0.3623	20	5
Ton*2.5_het	35	2.40	5.39	200	0.3079	20	5
Ton*5_het	35	4.80	12.35	220	0.2799	20	5

All scenarios listed in Table 6.4.1 were simulated with utilisations 0.6, 0.7, 0.8 and 0.9, employing a WFQ scheduler with equal weights for the two classes. The results are compared with those of the aggregate model in Figure 6-3 - Figure 6-7. Similar to Chapter 5, all simulated BSDRS and PEBS values shown were obtained by taking the average ST distribution from 100 independent simulation runs and deriving best-fit exponential decay graphs of the form $y=a.e^b.x$ where the BSDRS and PEBS are given by e^b and a respectively.

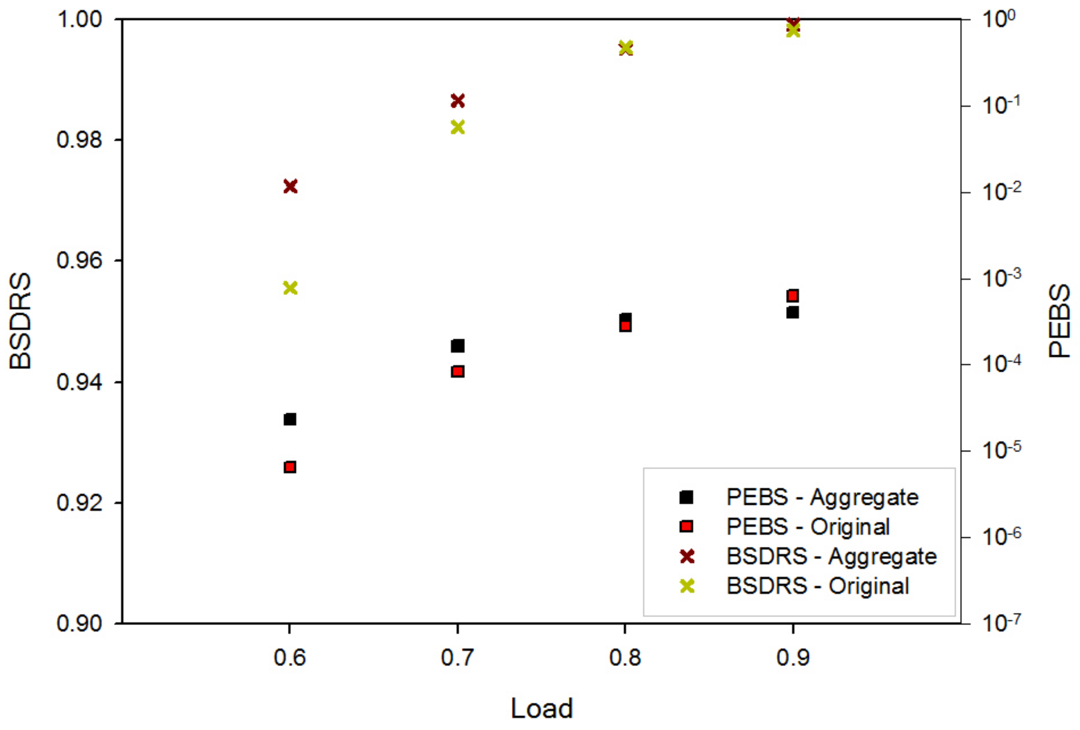


Figure 6-3 Comparison of PEBS and BSDRS of Class A: VoIP100_het

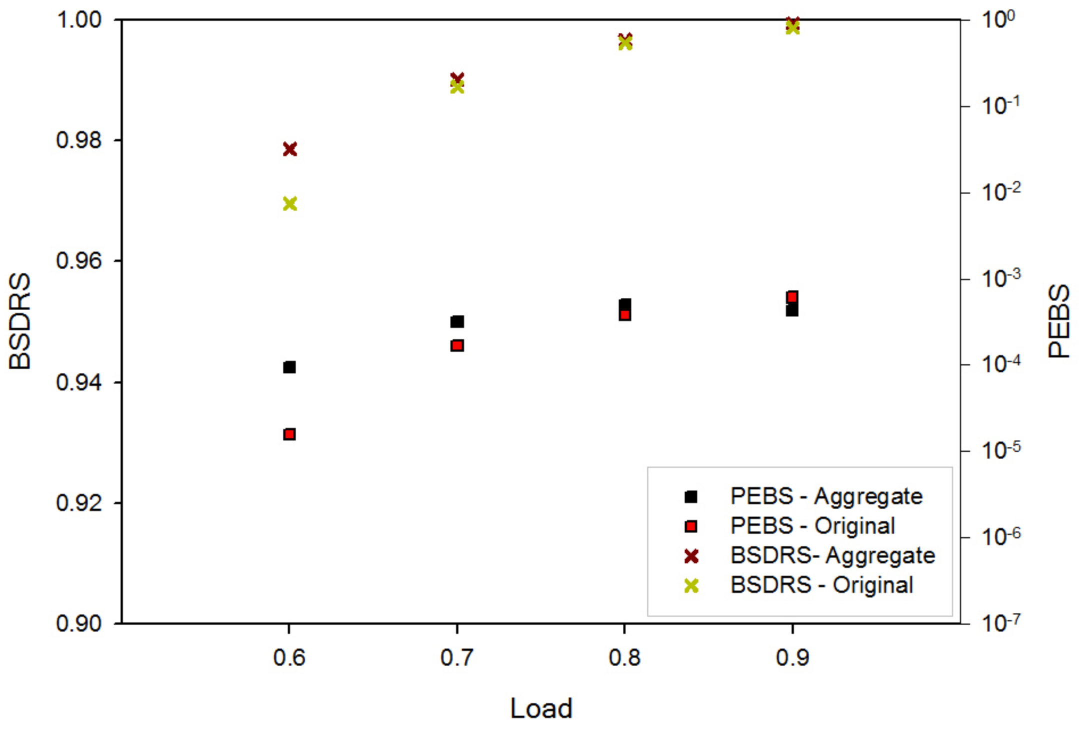


Figure 6-4 Comparison of PEBS and BSDRS of Class A: VoIP70_het

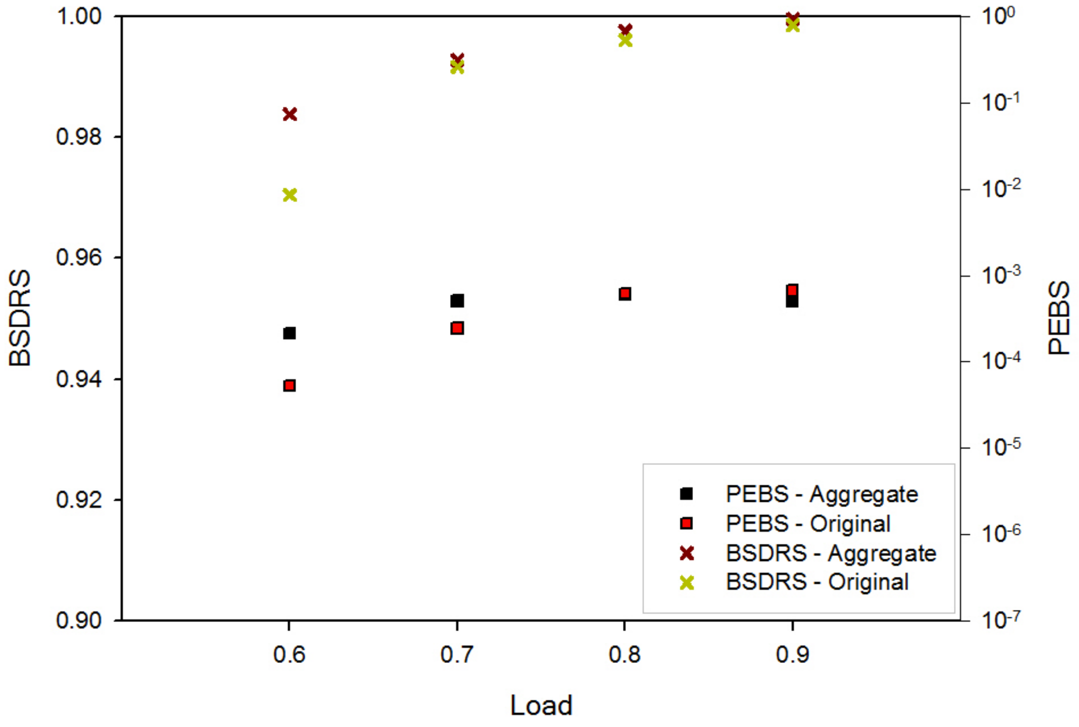


Figure 6-5 Comparison of PEBS and BSDRS of Class A: VoIP50_het

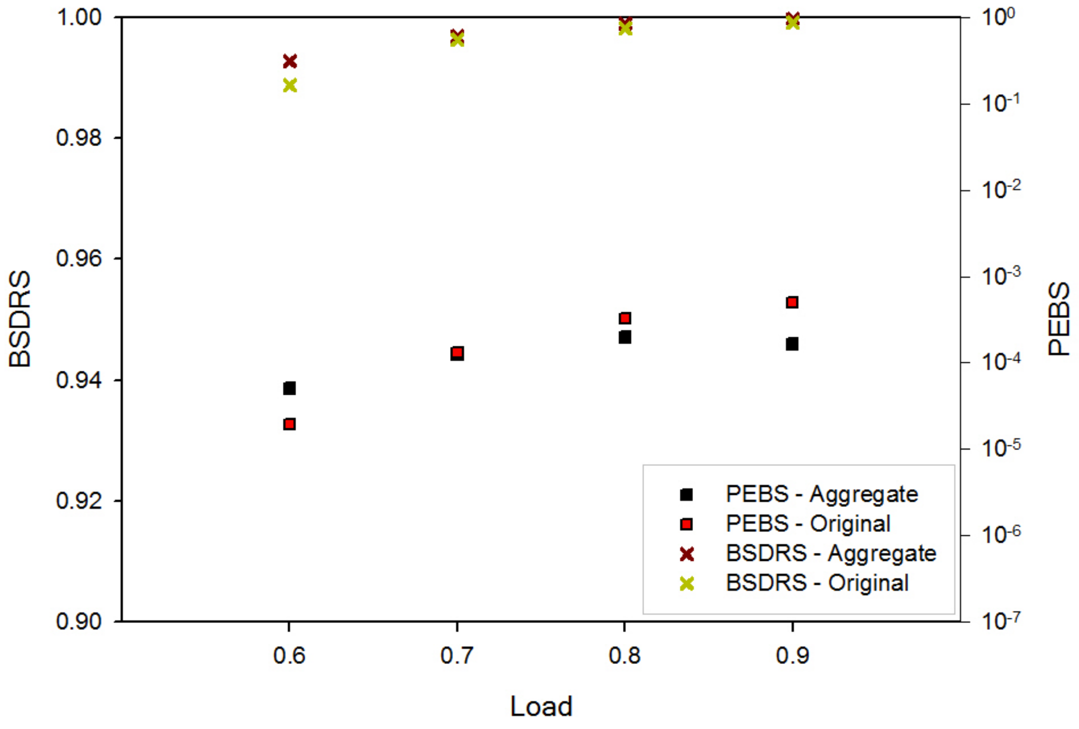


Figure 6-6 Comparison of PEBS and BSDRS of Class A: Ton*2.5_het

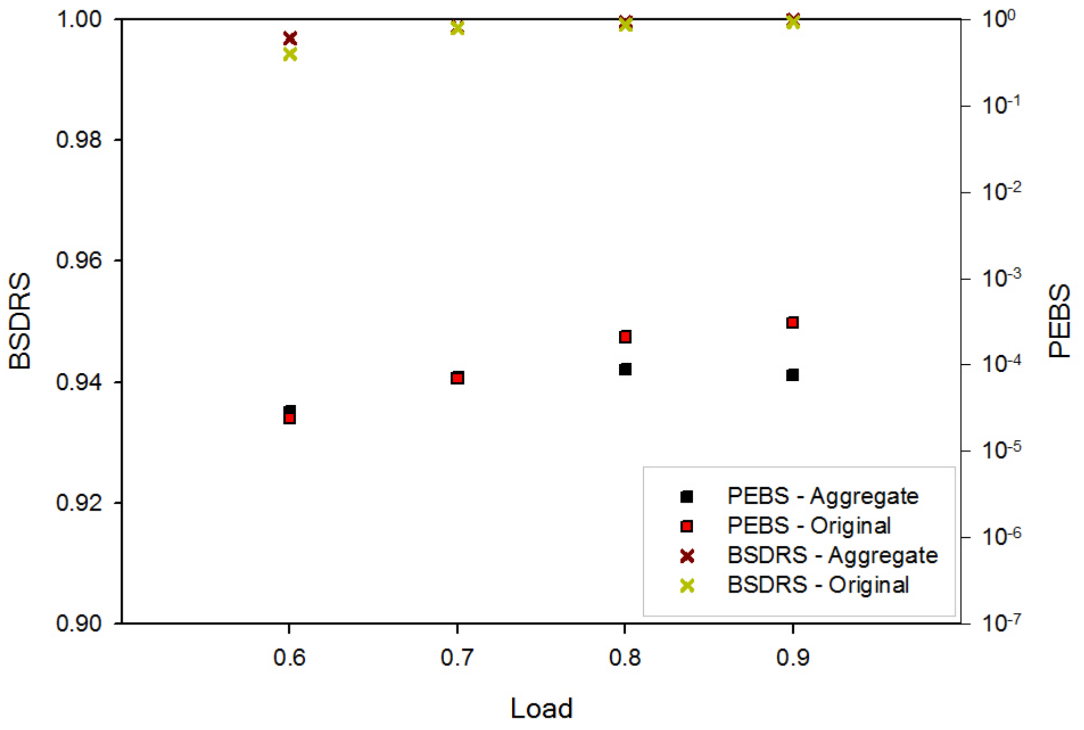


Figure 6-7 Comparison of PEBS and BSDRS of Class A: Ton*5_het

The results show that the aggregation technique introduced in section 6.4.3 is accurate over a range of input parameters, which means the aggregate model could be successfully employed to develop AS models (with minimal complexity), in order to evaluate the burst-scale ST of heterogeneous traffic classes scheduled by WFQ. This can be achieved by simulating Class A on its own and adjusting the ST based on the aggregation technique which ensures that Class A will behave approximately the same as if the other class(es) were present. Section 6.5 shows the speed-up that can be achieved by such AS models.

For further validation, the approximation used for q_A (probability of class A buffer being empty) was compared against the simulated values of the original model (see Figure 6-8), which shows the approximation is excellent.

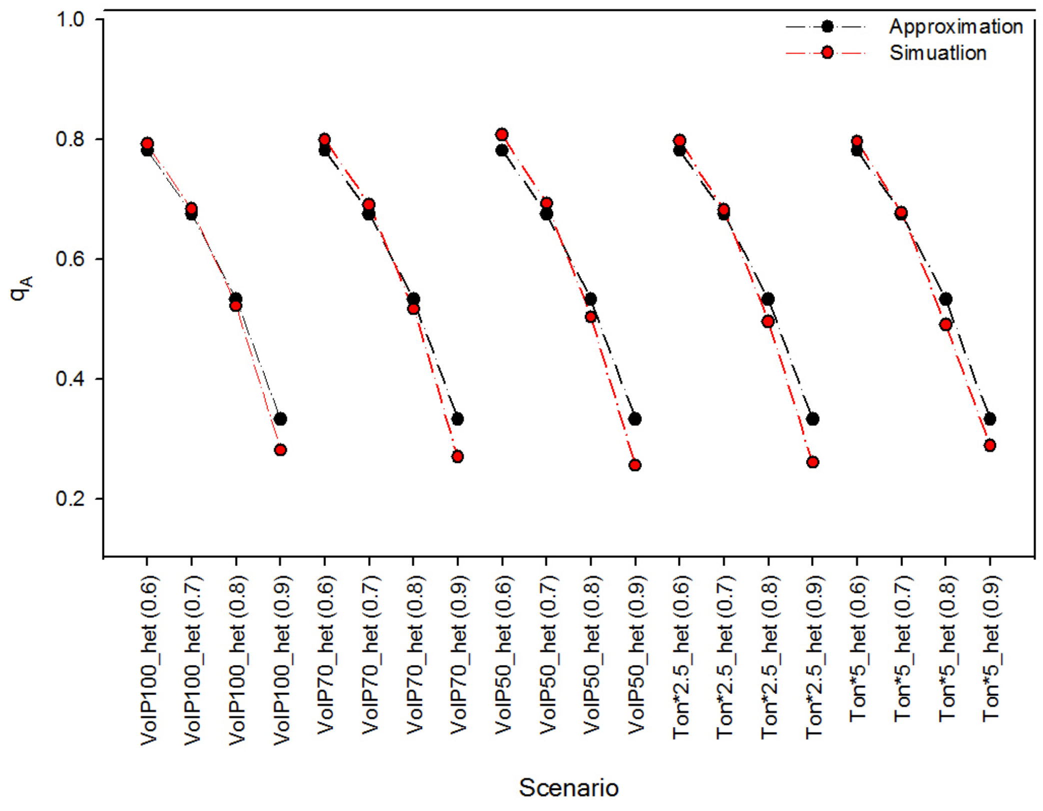


Figure 6-8 Validation of approximation to q_A

The accuracy of the q_A approximation is better at lower loads because at higher loads (e.g. 0.9), the approximation of $s[0]$ (i.e.$1-\rho$) isn't very accurate. This is because at higher loads, the queue keeps building up, introducing correlation among the system state seen by consecutive arrivals, unlike in the case of lower loads.

It is worth noting that the decay rate of the ST distribution of class A is always slightly worse than that of the overall ST distribution in the corresponding homogeneous case (Figure 6-9): this is because the service capacity is now shared.

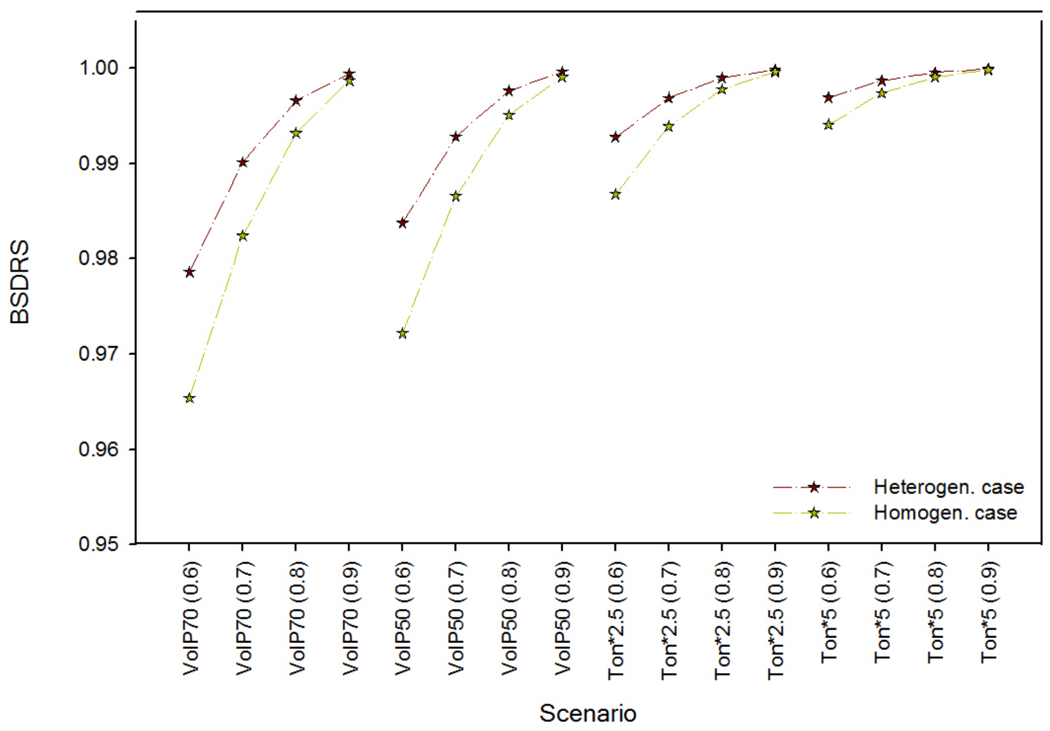


Figure 6-9 Comparison of BSDRS: homogeneous and heterogeneous (Class A)

Further, note that Class B only demonstrates packet-scale queueing as its arrival rate is comparatively small (which means only a small queue will build up during the time the server is attending to class A).

6.5 Application to Accelerated Simulation: speed-up factor

The magnitude of the speed-up factor achieved by applying the aggregation techniques developed in this chapter to develop AS models will depend on the traffic characteristics of Class A and Class B. However, recall that Class B is used to represent a group of classes (see section 6.4.2), which means the acceleration achieved can be quite significant when simulating the model of a real network scenario.

6.6 Summary

This chapter analyses the behaviour of bursty traffic in the presence of a WFQ scheduler, which is the most widely used practical approximation of the ideal GPS scheme. In the case of homogeneous classes, the behaviour is almost the same as in GPS, for which aggregation techniques (to evaluate the burst scale distributions of 'number in the system', X and 'system time', ST) are already presented in Chapter 5. Therefore, this chapter focuses on the evaluation of buffer performance in the presence of heterogeneous classes.

Review of the literature shows that determining the exact burst-scale behaviour of heterogeneous classes is a challenge: the main reason for this is the difficulty in determining the actual service capacity used by each class which has not so far been quantified (except for upper bounds and lower bounds). In this chapter, a novel, simple and accurate aggregation technique to determine the capacity used is presented, based on which the burst-scale ST distribution of the class of interest is derived. The validation results presented in this chapter demonstrate excellent agreement between the original and aggregate models. Consequently, AS models can be developed by removing the classes other than the class of interest from the simulator, which can achieve a substantial reduction of number of events.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

7.1 Introduction

The ongoing rapid increase in demand for improvements in packet networks calls for efficient methods that can evaluate network performance. This research has proposed, implemented and validated novel aggregation techniques that can efficiently evaluate the buffer performance in the presence of bursty traffic scheduled by the ideal GPS discipline. These techniques are also extended so that they are applicable in more practical situations, i.e. in the presence of heterogeneous traffic classes scheduled by the WFQ discipline, which is currently the most widely used practical approximation of GPS. All of the techniques developed in this thesis have demonstrated excellent accuracy and therefore enable the development of Accelerated Simulation (AS) models achieving a substantial reduction of number events, as shown throughout the thesis.

This research can be regarded as an important first step in the development of tools for the performance evaluation of practical networks scheduled by non-FIFO disciplines which are rapidly becoming popular today. Section 7.2 explains how these novel aggregation techniques can be further extended, while section 7.3 presents the final conclusions resulting from the work presented throughout thesis.

7.2 Future work

This research can be extended in at least three stages which are: employing slightly different scheduling mechanisms (perhaps those most popular in the industry), using other types of traffic (e.g. self-similar traffic) and finally moving from the single queueing system to end-to-end paths.

An important extension of the work presented in Chapter 6 would be to research the applicability of the developed aggregation techniques to a WFQ scheduler with unequal weights across the heterogeneous classes. This is important because, while an equally weighted WFQ scheduler offers equal bandwidth sharing to heterogeneous flows, in practice there is a need to allocate more bandwidth (than their equal share) for traffic belonging to certain classes (and therefore less bandwidth for others) when networks operate under congestion, i.e. differentiated Quality of Service (QoS).

Another interesting extension would be to investigate the behaviour of bursty traffic scheduled by the Deficit Round Robin discipline (also its variations - see section 3.4.2.2.3), which is popular in industry (e.g. Cisco). Despite the popularity, research suggests DRR is not as good as WFQ as an approximation to GPS. It may be worth exploring if the scheduler-invariant property prevails with this discipline, which will indicate if it is possible to directly apply the same techniques. If the burst-scale behaviour is considerably different to GPS, the aggregation techniques developed in this thesis will need to be extended. For heterogeneous classes (with DRR), it may be possible to approximate the service capacity received by each class by using the solution developed for WFQ in Chapter 6.

As Poisson traffic and SRD traffic are already addressed in this thesis, the next most important type of traffic that needs to be investigated is Long Range Dependent (LRD) traffic, as recent advances in network research suggest that effects of LRD traffic in today's networks are non-trivial. LRD traffic, which is self-similar[1], results in queue state distributions that decay slower than exponentially, which results in longer waiting times [33]. This is partly the result of the heavy-tailed nature of data files transferred over IP networks. Therefore, even though LRD traffic is also modelled by ON-OFF sources, at least the ON period should be based on a heavy-tailed distribution, e.g. Pareto[2]. This results in behaviour which makes acceleration a challenge, and at the same time makes AS models all the more necessary in order to efficiently evaluate network performance.

The aggregation techniques developed in this thesis for a single queueing system can be extended to evaluate the end-to-end performance of a traffic flow (or class) of interest. See Figure 7-1 where the interest is in the performance of *flow #1*. The end-to-end delay experienced by *flow #1* will be the summation of the system times experienced at each node, which can be evaluated by applying the novel aggregation techniques developed in Chapter 5 (in the case of homogeneous flows) or Chapter 6 (in the case of heterogeneous flows) at each node.

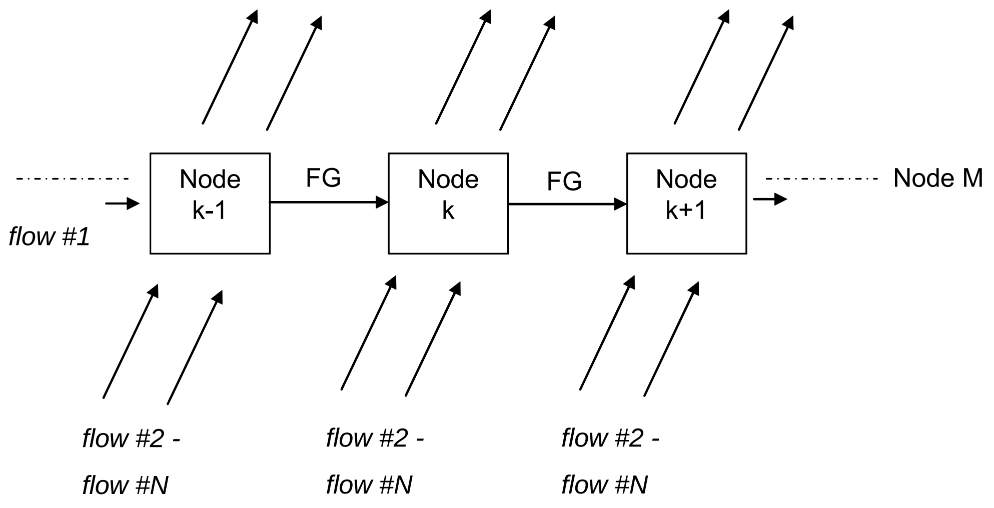


Figure 7-1 End-to-end path

[1] Traffic that is bursty on many or all times scales (unlike SRD traffic) are described as self-similar [86].

[2] A new traffic source with an exponentially-distributed OFF period and Pareto-distributed ON period was developed during this research, which was validated by comparison with the Geo/Pareto/1 theoretical model: therefore, this source is ready be used to model LRD traffic.

Finally, the results of this research can be used to simplify the design of a network emulator, so that the entire network layer does not need to be built. As shown in Figure 7-2, the network layer at each node can be built based on aggregate models developed in this research, which will greatly simply the implementation of the emulator. *Flow #1* here could carry the traffic belonging to a particular application, e.g. VoIP over RTCP.

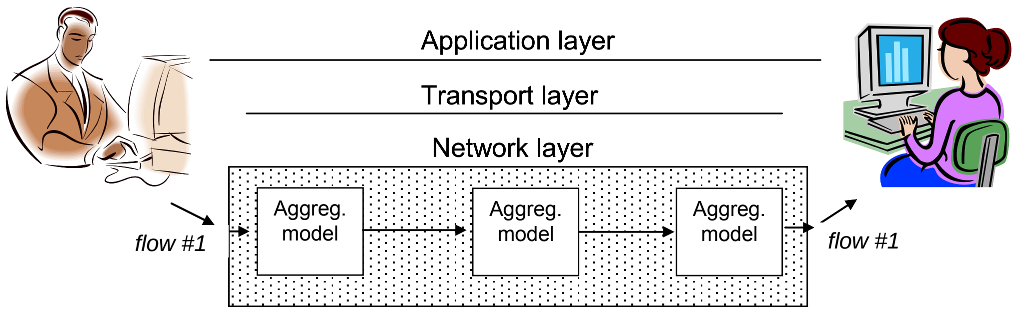


Figure 7-2 Application of aggregate models to a network emulator

7.3 Conclusions

This research has developed novel aggregation techniques that can accurately approximate the distributions of 'number of packets in the system', X and 'system time', ST in the presence of both bursty and non-bursty traffic flows scheduled by GPS and WFQ. The aggregation techniques enable the evaluation of performance via Accelerated Simulation (AS) models (see Chapter 2), which means the amount of resources required for the simulation is substantially reduced (see Chapter 4 – 6 for specific details). This means the desired results can be obtained much faster than in an ordinary simulation, enabling researchers to identify any developments required, in a timely manner. This research is a significant contribution to networking research because AS modelling with processor-sharing schedulers is completely novel, and important as FIFO scheduling is often considered to be inadequate when users require QoS (see Chapter 3).

In addition to speeding up simulations, the novel techniques presented in this thesis can also be used for analytical prediction of buffer performance, which will again speed-up the process because all of these techniques are of minimal complexity (i.e. they are all 'back-of-the-envelope' type of calculations and do not require complex mathematical tools such as Matlab).

Chapter 4 has covered the aggregation of traffic in the presence of homogeneous Poisson traffic flows scheduled by GPS. The original model (with N flows) is replaced by an aggregate model (with a single flow, i.e. flow of interest) where the novel aggregation techniques ensure the aggregate model's behaviour closely approximates that of the original model, in terms of queue state and system time experienced by the flow of interest. The results of this chapter are significant for the overall research because it covers the evaluation of the packet-scale component of bursty traffic (which is the main concern), as the behaviour of Poisson traffic coincides with the packet-scale behaviour of bursty traffic.

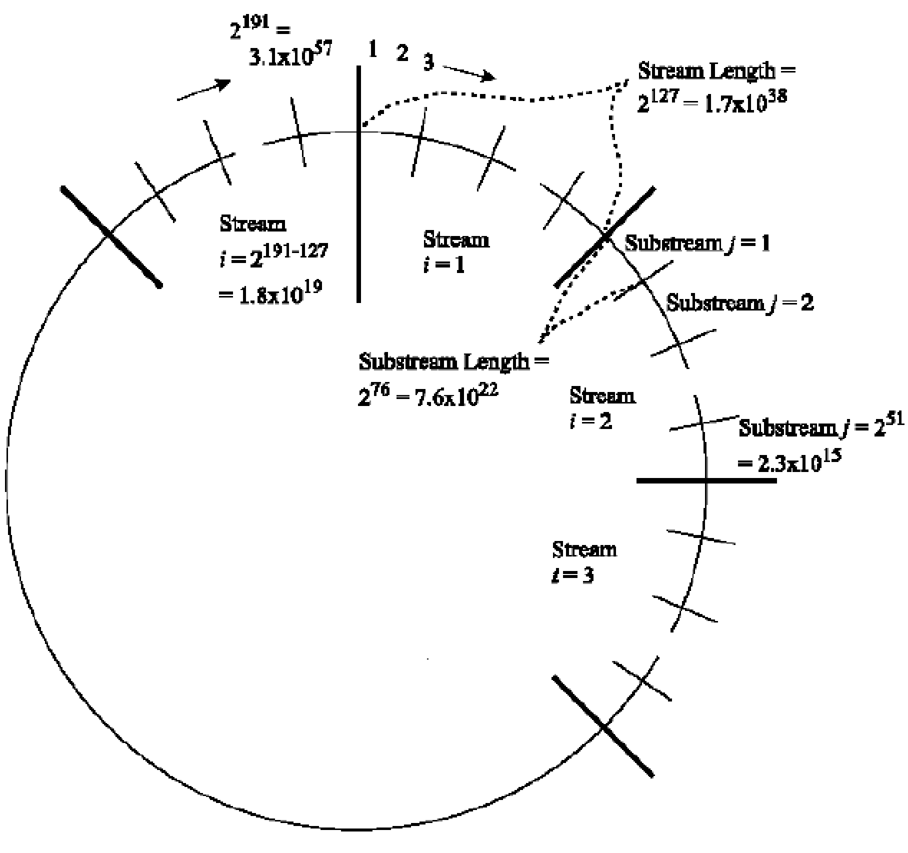
Chapter 5 has investigated bursty traffic, modelled by multiplexed exponential ON-OFF sources. Here, an interesting property is discovered, i.e. scheduler-invariance of the burst-scale behaviour (unlike the packet-scale behaviour), from FIFO to GPS. This discovery greatly simplifies analysis, enabling the evaluation of X based on aggregation techniques developed for FIFO (e.g. ref [31, 32] which replaces *N* ON-OFF flows with a single equivalent flow). Aggregation techniques to evaluate ST, however, do not exist in the literature; therefore, Chapter 5 develops a novel technique to determine the burst-scale ST distribution which has proved to be an excellent approximation, as demonstrated by comparison with (original, non-accelerated) simulation results. The scheduler-invariant property facilitates further acceleration of simulation as the complex GPS scheduler could now be replaced by the simple FIFO (for the purpose of studying burst-scale behaviour). The scheduler-invariance also casts doubt on the value of engaging expensive resources in implementing complex non-FIFO scheduling mechanisms, despite the protection they provide from ill-behaved sources.

Chapter 6 takes the research a step further, by extending the aggregation techniques to be applicable to heterogeneous traffic classes, scheduled by WFQ (as GPS is not implemented in practice). The primary challenge here is to determine the portion of service capacity received by each class, which has previously only been determined in terms of (upper and lower) bounds and via measurements (see section 6.4.1 for details). Chapter 6 presents a simple and novel analytical technique for the approximation of the service capacity received by a class of interest, which (combined with the burst-scale evaluation techniques from Chapter 5) facilitates AS models by removing the other traffic classes from the simulation.

Even though the key performance metric focused in this research is ST (as this is the most important from the perspective of the user), the evaluation of queue state also leads to important implications for the industry, for instance, for the calculation of loss probabilities and more importantly for buffer dimensioning based on a given level of loss probability. For example, the default G.729 codec for VoIP requires packet loss to be far less than 1% to avoid audible errors [79]. The techniques developed in Chapter 5 could be used to easily work out the minimum size of the buffer required for VoIP traffic, in order fulfil this requirement.

APPENDIX A – RANDOM NUMBER GENERATOR (RNG) MRG32K3A

The RNG MRG32k3a, which is implemented in NS2 has a well-structured mechanism for providing multiple independent random processes across multiple runs of a simulation: Figure A-1 depicts how this is achieved. Each random process in the simulator is seeded to the beginning of the next independent stream, and MRG32k3a allows 1.8x10^{19} such independent processes within a simulation. The sub-streams shown in the figure are used for multiple independent replications of a simulation i.e. for each replication, a different sub-stream should be used and in this RNG, a maximum of 2.3x10^{15} such replications are possible. Further, each sub-stream has a period of 7.6x10^{22} which gives the number of random numbers that can be produced for each process in a single replication [15].



2^{191} = 3.1x10^{57}
1 2 3
Stream Length = 2^{127} = 1.7x10^{38}
Stream $i = 2^{191-127}$ = 1.8x10^{19}
Stream $i = 1$
Substream $j = 1$
Substream $j = 2$
Substream Length = 2^{76} = 7.6x10^{22}
Stream $i = 2$
Substream $j = 2^{51}$ = 2.3x10^{15}
Stream $i = 3$

Figure A-1 Overall arrangement of MRG32k3a [15]

APPENDIX B – PGPS ALGORITHM

The three main tasks involved in the PGPS algorithm are;

(i) Updating the virtual time

(ii) Updating the order of service

(iii) Calculating the real times for the departure of packets

Out of these tasks, (i) is the most challenging and also the vital component, since tasks (ii) and (iii) depend on it, as described below.

(i) Updating the virtual time

Let

t_j = time at which the j^{th} event occurs. ('event': arrival or departure of a packet)

t_1 = time of the first arrival of a busy period = 0

B_j = set of sessions that are busy in the interval (t_{j-1}, t_j). This is a fixed value for j = *2,3,….*

$V(t_i)$ = virtual time at time t_j

$V(t_1) = V(0)$ = Virtual time when the server is idle = 0

ϕ_i = weight assigned to sub-queue i (i.e. relative rate of service for sub-queue i)

Then virtual time is updated (whenever there is an event) as follows:

$$V(t_{j-1} + \tau) = V(t_{j-1}) + \frac{\tau}{\sum_{i \in B_j} \phi_i} \quad \text{where } \tau \le t_j - t_{j-1},\ j = 2,3,...$$

Most of the overhead occurs in keeping track of sets B_j, since they need updating at every event (arrival and departure) in order to detect whether each session is active or idle.

(ii) Updating the order of service

This is done at the arrival of a packet.

a_i^k = time at which packet *k* of sub-queue *i* arrives

L_i^k = length of packet *k* of sub-queue *i*

s_i^k = virtual time at which packet k of sub-queue *i* starts service

F_i^k = virtual time at which packet k of sub-queue *i* finishes service

$$s_i^k = \max\left\{F_i^{k-1}, V\left(a_i^k\right)\right\}$$

$$F_i^k = s_i^k + \frac{L_i^k}{\phi_i}$$

The packets are stamped with the value of $F_i(k)$ and inserted into a sorted queue (common for all the sessions), from which the scheduler chooses the next packet to serve.

<u>(iii) Calculating the real times for the departure of packets</u>

F_{min} = smallest virtual time finishing time of a packet in the system at time *t* (the packet with this value will be chosen first by the scheduler)

Next(t) = real-time of the next packet departure after time *t*, if there were no arrivals after time t (so that the next immediate event would be the departure)
Then, *Next(t)* would be the real time corresponding to F_{min}. i.e. in the formula from (i),

$$\tau = Next(t) - t$$

$$\Rightarrow F_{\min} = V(t) + \frac{Next(t) - t}{\sum_{i \in B_j} \phi_i}$$

$$\Rightarrow Next(t) = t + (F_{\min} - V(t)) \sum_{i \in B_j} \phi_i$$

For more details, see ref. [57].

APPENDIX C – EXCESS RATE (ER) ANALYSIS

The ER queueing analysis is based on the identification of an Imbedded Markov Chain (IMC) at ER arrivals. ER packets are those which must be buffered as they represent an ‘excess’ of instantaneous arrival rate over the service rate. Define the fundamental time unit as the time required to serve an average-length packet. Then, if *N* packets arrive in any time unit, then that time unit experiences *N-1* excess packets. The definition of ER packets is important because of its relationship to the change in queue state. i.e. for every ER packet, the queue size increases by one. Accordingly, the arrival of ER packets is connected via balance equations representing adjacent queue states, from which formulae for queue state probability are derived.

The general formula for *p(k)*, the probability that an arriving packet sees *k* already in the queue for an M/G/1 system with utilisation ρ is derived as follows:

$$p(k) = (1-\rho).\left[\frac{q + (1-q)((1-a[0]-a[1])/(1-a[1]))}{(a[0])/(1-a[1])}\right]^{k}$$

where a(k) is the probability of having k arrivals in a packet service time. q is defined as the probability of having another ER packet in a time unit which just had one. $q = a[3]/a[2] = \lambda/3$ where λ is the arrival rate.

The definitions of $a[0]$ and $a[1]$ are determined by the packet size distribution of the queueing system concerned. For Poisson distributed packet sizes they are defined as;

$$a[0] = (e^{-\rho}/\rho)[\exp(\rho e^{-\lambda}) - 1]$$

$$a[1] = \lambda e^{-\lambda} e^{-\rho}(1 + \rho e^{-\lambda})\exp(\rho e^{-\lambda})$$

For more details, see ref. [32].

AUTHOR PUBLICATIONS

(i) V. Amaradasa and J. Schormans, "Future Simulation Acceleration of Packet Networks Using GPS Scheduling" , *Sixth Annual PostGraduate Symposium on the Convergence of Telecommunications, Networking & Broadcasting (PGNET)*, Liverpool, UK – 2005

(ii) V. Amaradasa and J. Schormans, "Accelerated Simulation of a GPS Scheduler Using Traffic Aggregation", *Fourth International Working Conference on.* Performance Modelling and Evaluation*.* *of* Heterogeneous Networks *(HETNET)*, Ilkley, West Yorkshire, UK, 2006.

(iii) V. Amaradasa, J. Schormans, J.M. Pitts and C.M. Leung, "Evaluating Overflow Probability For VoIP Buffer Dimensioning", *accepted and awaiting publication in IET Communications*.

REFERENCES

[1] R. K. Jain, *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*: Wiley-Interscience, 1991.

[2] M. Hasib, J. Schormans, and T. Timotijevic, "Accuracy of packet loss monitoring over networked CPE," *IET Communications,* vol. 1, pp. 507-513, 2007.

[3] J. A. Schormans and C. M. Leung, "Measurement for guaranteeing QoS in broadband multiservice networks," in *Mobile And Wireless Systems Beyond 3g: Managing New Business Opportunities*: IRM Press, 2005.

[4] T. Timotijevic, C. M. Leung, and J. Schormans, "Accuracy of measurement techniques supporting QoS in packet-based intranet and extranet VPNs," *IEE Proceedings-Communications,* vol. 151, pp. 89-94, 2004.

[5] M. Roughan, "Fundamental bounds on the accuracy of network performance measurements," in *Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems* Banff, Alberta, Canada: ACM, 2005.

[6] J. F. Kurose and H. T. Mouftah, "Computer-aided modeling, analysis, and design of communication networks," *IEEE Journal on Selected Areas in Communications,* vol. 6, pp. 130-145, 1988.

[7] A. M. Law and W. D. Kelton, *Simulation modeling and analysis*, 3rd ed. ed. Boston ; London: McGraw-Hill, 2000.

[8] K. Pawlikowski, H. D. J. Jeong, and J. S. R. Lee, "On credibility of simulation studies of telecommunication networks," in *Communications Magazine*. vol. 40: IEEE, 2002, pp. 132-139.

[9] M. D. Fraser, "Simulation," in *Potentials*. vol. 11: IEEE, 1992, pp. 15 - 18.

[10] P. L'Ecuyer, "Good parameters and implementations for combined multiple recursive random number generators," *Operations Research,* vol. 47, pp. 59-164, 1999.

[11] NS-developers, "ns: Change History, http://www.isi.edu/nsnam/ns/CHANGES.html," 2008.

[12] M. C. Weigle, "Improving confidence in network simulations," in *38th Winter Simulation Conference*, Monterey, California, 2006, pp. 2188 - 2194.

[13] Intel, "Intel® Math Kernel Library," 9 ed, 2006.

[14] P. L'Ecuyer, R. Simard, E. J. Chen, and W. D. Kelton, "An object-oriented random number package with many long streams and substreams," *Operations Research,* vol. 50, pp. 1073 - 1075, 2002.

[15] *NS Manual*, 2008.

[16] J. Banks, *Handbook of simulation : principles, methodology, advances, applications and practice*. New York ; Chichester: Wiley, 1998.

[17] "Network Performance Objectives for IP-Based Services," in *Y.1541*, ITU, Ed., 2006.

[18] G. F. Riley and M. H. Ammar, "Simulating large networks: How big is big enough?," in *First International Conference on Grand Challenges for Modeling and Simulation*, 2002.

[19] P. E. Heegard, "Comparison of speed-up techniques for simulation," in *The 12th Nordic Teletraffic Seminar (NTS-12)*, Trondheim, Norway, 1995, pp. 407-420.

[20] C. Huang, M. Devetsikiotis, I. Lambadaris, and A. R. Kaye, "Fast simulation for self-similar traffic in ATM networks," in *Communications, 1995. ICC '95 Seattle, 'Gateway to Globalization', 1995 IEEE International Conference on*, 1995, pp. 438-444 vol.1.

[21] A. R. Dabrowski, G. Lamothe, and D. R. McDonald, "Accelerated simulation of ATM switching fabrics," in *Fields Institute Workshop on Monte Carlo Methods*, New York, 1999.

[22] S. Asmussen, K. Binswanger, and B. Hojgaard, "Rare Events Simulation for Heavy-Tailed Distributions," *Bernoulli,* vol. 6, pp. 303-322, 2000.

[23] J. R. Gallardo, D. Makrakis, and L. Orozco-Barbosa, "Fast simulation of broadband telecommunications networks carrying long-range dependent bursty traffic," in *Simulation Conference Proceedings, 1999 Winter*, 1999, pp. 374-381 vol.1.

[24] A. Karasaridis and D. Hatzinakos, "On the modeling of network traffic and fast simulation of rare events using alpha-stable self-similar processes," in *Higher-Order Statistics, 1997. Proceedings of the IEEE Signal Processing Workshop on*, 1997, pp. 268-272.

[25] M. Villen-Altamirano and J. Villen-Altamirano, "On the efficiency of RESTART for multidimensional state systems," *Acm Transactions on Modeling and Computer Simulation,* vol. 16, pp. 251-279, Jul 2006.

[26] C. Kelling and G. Hommel, "Rare event simulation with an adaptive "RESTART" method in a Petri net modeling environment," in *Parallel and Distributed Real-Time Systems, 1996. Proceedings of the 4th International Workshop on*, 1996, pp. 229-234.

[27] P. E. Heegard, "A survey of speedup simulation techniques," in *Workshop tutorial on rare event simulation* Aachen, Germany, 1997.

[28] J. M. Pitts, "Cell-rate modelling for accelerated simulation of ATM at the burst level," *IEE Proceedings-Communications,* vol. 142, pp. 379-385, 1995.

[29] J. A. Schormans, E. Liu, L. Cuthbert, and J. Pitts, "A hybrid technique for accelerated simulation of ATM networks and network elements," *ACM Trans. Model. Comput. Simul.,* vol. 11, pp. 182-205, 2001.

[30] E. Liu, "A Hybrid Queueing Model for Fast Broadband Networking Simulation," in *Department of Electronic Engineering*: Queen Mary, University of London, 2002.

[31] C. M. Leung, J. A. Schormans, J. M. Pitts, and A. Woolf, "Accurate decay rate prediction for burst-scale queueing in packet buffers," *Electronics Letters,* vol. 39, pp. 253-254, 2003.

[32] J. M. Pitts and J. A. Schormans, *Introduction to IP and ATM Design and Performance: With Applications Analysis Software*, 2nd ed., 2000.

[33] A. H. I. Ma, "Accelerated Simulation of Power-Law Traffic in Packet Networks," in *PhD, Department of Electronic Engineering*: Queen Mary, University of London, 2002.

[34] A. H. I. Ma and J. A. Schormans, "Fast, stable simulation of power-law packet traffic using concatenated acceleration techniques," *IEE Proceedings-Communications,* vol. 152, pp. 420-426, Aug 2005.

[35] S. H. S. Ariffin and J. A. Schormans, "Efficient accelerated simulation technique for packet switched networks: a buffer with two priority inputs,"

in *IEEE International Conference on Communications*, 2004, pp. 2246-2250 Vol.4.

[36] T. Konstantopoulos and S. J. Lin, "Fractional Brownian approximations of stochastic networks," in *Lecture Notes in Statistics*. vol. 117, P. Glasserman, K. Sigman, and D. Yao, Eds. New York: Springer, 1996, pp. 257-274.

[37] S. Lu and J. A. Schormans, "SSIM: A Multi-Resolution Fluid Traffic Simulator for MANETs," in *Third International Conference on Mobile Computing and Ubiquitous Networking (ICMU2006)*, London, 2006.

[38] C. Benveniste and P. Heidelberger, "Parallel simulation of the IBM SP2 interconnection network," in *Simulation Conference Proceedings, 1995. Winter*, 1995, pp. 584-589.

[39] K. S. Perumalla, "'mhu'sik - a micro-kernel for parallel/distributed simulation systems," in *Principles of Advanced and Distributed Simulation, 2005. PADS 2005. Workshop on*, 2005, pp. 59-68.

[40] L. Bononi, M. Bracuto, G. D'Angelo, and L. Donatiello, "Concurrent replication of parallel and distributed simulations," in *Principles of Advanced and Distributed Simulation, 2005. PADS 2005. Workshop on*, 2005, pp. 234-243.

[41] A. Mishra and K. J. Christensen, "Improving the Simulation Process in Model-Based Codesign with Romote Execution with Load Observation and Distribution (RELOAD)," in *International Conference on Parallel and Distributed Processing Techniques and Applications*, 1999, pp. 3009-3015.

[42] "Network Simulator 2, http://www.isi.edu/nsnam/ns/," Computer Systems Engineering Group at Lawrence Berkeley Laboratory, University of California.

[43] J. Chung and M. Claypool, "NS by Example," Worcester Polytechnic Institute, 2004.

[44] J. W. Roberts, "Traffic theory and the Internet," *IEEE Communications Magazine,* vol. 39, pp. 94-99, 2001.

[45] V. RAKOCEVIC, "Dynamic Bandwidth Allocation in Multi-class IP Networks using Utility Functions," in *PhD, Department of Electronic Engineering*: Queen Mary, University of London, 2002.

[46] L. Kleinrock and H. Levy, "The Analysis of Random Polling Systems," *Operations Research,* vol. 36, pp. 716-732, Sep-Oct 1988.

[47] H. Takagi, *Analysis of polling systems*: MIT Press, 1986.

[48] M. Sakata, S. Noguchi, and J. Oizumi, "An Analysis of the M/G/1 Queue under Round-Robin Scheduling," *Operations Research,* vol. 19, pp. 371-385, 1971.

[49] L. Kleinrock, "Time-shared Systems: a theoretical treatment," *J. ACM,* vol. 14, pp. 242-261, 1967.

[50] L. Kleinrock, *Queueing Systems Volume II: Computer Applications*: John Wiley &Sons, 1976.

[51] S. Borst, D. van Ooteghem, and B. Zwart, "Tail asymptotics for discriminatory processor-sharing queues with heavy-tailed service requirements," *Performance Evaluation,* vol. 61, pp. 281-298, Jul 2005.

[52] J. Roberts and L. Massoulié, "Bandwidth Sharing and Admission Control for Elastic Traffic," in *ITC Specialist Seminar*, Yokohama, 1998.

[53] E. Altman, K. Avrachenkov, and U. Ayesta, "A survey on discriminatory processor sharing," *Queueing Systems,* vol. 53, pp. 53-63, Jun 2006.

[54] S. Borst, O. Boxma, and P. Jelenkovic, "Asymptotic behavior of generalized processor sharing with long-tailed traffic sources," in *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 2000, pp. 912-921 vol.2.

[55] S. F. Yashkov, "Processor-sharing queues: some progress in analysis," *Queueing Syst. Theory Appl.,* vol. 2, pp. 1-17, 1987.

[56] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," in *Symposium proceedings on Communications architectures and protocols* Austin, Texas, United States: ACM, 1989.

[57] A. K. Parekh and R. G. Gallager, "A Generalized Processor Sharing Approach to Flow-Control in Integrated Services Networks - the Multiple Node Case," *IEEE-ACM Transactions on Networking,* vol. 2, pp. 137-150, Apr 1994.

[58] J. Roberts, U. Mocci, and J. Virtamo, *Broadband Network Teletraffic, Final Report of Action COST 242*: Springer Verlag, 1996.

[59] G. Lisa, "QoS in Packet-Switching Networks," Politecnico Di Torino, Turin, Italy 1999.

[60] M. Shreedhar and G. Varghese, "Efficient fair queueing using deficit round-robin," *IEEE-ACM Transactions on Networking,* vol. 4, pp. 375-385, Jun 1996.

[61] T. V. Do, L. Jereb, K. Umann, and G. Wolfner, "Simulation Comparison of Link Scheduling Algorithms: Impacts on Cell Delay Variation'," in *Third Workshop on Performance Modelling and Evaluation of ATM Networks* Ilkley UK, 1995.

[62] S. J. Golestani, "A self-clocked fair queueing scheme for broadband applications," in *INFOCOM '94. Networking for Global Communications., 13th Proceedings IEEE*, 1994, pp. 636-646 vol.2.

[63] P. Goyal, H. M. Vin, and P. C. Cheng, "Start-time fair queueing: A scheduling algorithm for integrated services packet switching networks," *IEEE-ACM Transactions on Networking,* vol. 5, pp. 690-704, Oct 1997.

[64] L. X. Zhang, "Virtualclock - a New Traffic Control Algorithm for Packet-Switched Networks," *Acm Transactions on Computer Systems,* vol. 9, pp. 101-124, May 1991.

[65] D.-y. Lee and S.-h. Oh, "A new DBA scheme to improve bandwidth utilization in EPONs," in *Advanced Communication Technology, 2006. ICACT 2006. The 8th International Conference*, 2006, p. 5 pp.

[66] A. Sayenko, "Weighted Fair Queueing, http://www.cc.jyu.fi/~sayenko/src/wfq-1.2.2.tar.gz," 2004.

[67] E. G. Coffman, R. R. Muntz, and H. Trotter, "Waiting Time Distributions for Processor-Sharing Systems," *Journal of the Acm,* vol. 17, pp. 123-&, 1970.

[68] J. A. Morrison, "Response-Time Distribution for a Processor-Sharing System," *SIAM Journal on Applied Mathematics,* vol. 45, pp. 152-167, 1985.

[69] T. J. Ott, "The Sojourn-Time Distribution in the M/G/1 Queue with Processor Sharing," *Journal of Applied Probability,* vol. 21, pp. 360-378, 1984.

[70] R. Egorova, B. Zwart, and O. Boxma, "Sojourn time tails in the M/D/1 processor sharing queue," *Probability in the Engineering and Informational Sciences,* vol. 20, pp. 429-446, 2006.

[71] H. Michiel and K. Laevens, "Teletraffic engineering in a broad-band era," *Proceedings of the Ieee,* vol. 85, pp. 2007-2033, Dec 1997.

[72] V. Paxson and S. Floyd, "Wide Area Traffic - the Failure of Poisson Modeling," *IEEE-ACM Transactions on Networking,* vol. 3, pp. 226-244, Jun 1995.

[73] I. Elhanany and D. Sadot, "Queueing analysis of Markov modulated ON/OFF arrivals with geometric service times," in *The 22nd Convention of Electrical and Electronics Engineers*, in Israel, 2002, pp. 189 - 191.

[74] R. Krishnan and J. A. Silvester, "Resource allocation in broadband networks-cell, burst or connection level?," in *IEEE ICC*, 1994, pp. 86-90.

[75] X. Yang, "Designing traffic profiles for bursty Internet traffic," in *Global Telecommunications Conference, GLOBECOM*, 2002, pp. 2149 - 2154.

[76] D. Anick, D. Mitra, and M. M. Sondhi, "Stochastic theory of a data handling system with multiple sources," *Bell System Technical Journal,* vol. 61, pp. 1871-1894, 1982.

[77] H. Hagirahim, "Expression for the probability of packet loss owing to buffer overflow for multiplexing packetised voice," *IEE Proceedings-Communications,* vol. 153, pp. 238-244, 2006.

[78] C. M. Leung, "Non-Intrusive Measurement in Packet Networks and its Applications," in *PhD, Department of Electronic Engineering*: Queen Mary, University of London, 2004.

[79] "G.729 : Coding of speech at 8 kbit/s using conjugate-structure algebraic-code-excited linear prediction " in *G.729 (01/07)*, ITU, Ed., 2007.

[80] T. Szigeti and C. Hattingh "Quality of Service Design Overview," in *End-to-End QoS Network Design:Quality of Service in LANs, WANs, and VPNs*: Cisco Press, 2004.

[81] R. Stewart, "End-to-End Delay Analysis for Small/Medium Scale IP Networks." vol. PhD, 2002.

[82] L. Kleinrock, *Queueing Systems Volume I: Queueing Theory*: Wiley-Interscience, 1975.

[83] B. Lee, W. K. Edward, S. Scott, S. Ion, and Z. Hui, "Endpoint admission control: architectural issues and performance," in *Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication* Stockholm, Sweden: ACM, 2000.

[84] A. Kuzmanovic and E. W. Knightly, "Measuring service in multi-class networks," in *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 2001, pp. 1281-1289 vol.3.

[85] J. Y. Qiu and E. W. Knightly, "Inter-class resource sharing using statistical service envelopes," in *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE,* 1999, pp. 1404-1411 vol.3.

[86] M. Crovella and B. Bestavros, "Explaining World Wide Web Traffic Self-Similarity," Boston University 1995.