Overload Control of a Finite Capacity, Single Server Queue With Vacations

Edwin B. Martin

A Thesis

Submitted to the Faculty of Graduate Studies

in Partial Fulfillment of the Requirements

for the Degree of

MASTER OF SCIENCE

Department of Mechanical and Industrial Engineering

University of Manitoba

Winnipeg, Manitoba

(c) 1993

National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-315-85947-4

Canada

Name _____

Dissertation Abstracts International is arranged by broad, general subject categories. Please select the one subject which most nearly describes the content of your dissertation. Enter the corresponding four-digit code in the spaces provided.

OPERATIONS RESEARCH — SUBJECT TERM

0796 — SUBJECT CODE

U·M·I

Subject Categories

THE HUMANITIES AND SOCIAL SCIENCES

COMMUNICATIONS AND THE ARTS
Architecture 0729
Art History 0377
Cinema 0900
Dance 0378
Fine Arts 0357
Information Science 0723
Journalism 0391
Library Science 0399
Mass Communications 0708
Music 0413
Speech Communication 0459
Theater 0465

EDUCATION
General 0515
Administration 0514
Adult and Continuing 0516
Agricultural 0517
Art 0273
Bilingual and Multicultural 0282
Business 0688
Community College 0275
Curriculum and Instruction 0727
Early Childhood 0518
Elementary 0524
Finance 0277
Guidance and Counseling 0519
Health 0680
Higher 0745
History of 0520
Home Economics 0278
Industrial 0521
Language and Literature 0279
Mathematics 0280
Music 0522
Philosophy of 0998
Physical 0523
Psychology 0525
Reading 0535
Religious 0527
Sciences 0714
Secondary 0533
Social Sciences 0534
Sociology of 0340
Special 0529
Teacher Training 0530
Technology 0710
Tests and Measurements 0288
Vocational 0747

LANGUAGE, LITERATURE AND LINGUISTICS
Language
- General 0679
- Ancient 0289
- Linguistics 0290
- Modern 0291

Literature
- General 0401
- Classical 0294
- Comparative 0295
- Medieval 0297
- Modern 0298
- African 0316
- American 0591
- Asian 0305
- Canadian (English) 0352
- Canadian (French) 0355
- English 0593
- Germanic 0311
- Latin American 0312
- Middle Eastern 0315
- Romance 0313
- Slavic and East European 0314

PHILOSOPHY, RELIGION AND THEOLOGY
Philosophy 0422
Religion
- General 0318
- Biblical Studies 0321
- Clergy 0319
- History of 0320
- Philosophy of 0322

Theology 0469

SOCIAL SCIENCES
American Studies 0323
Anthropology
- Archaeology 0324
- Cultural 0326
- Physical 0327

Business Administration
- General 0310
- Accounting 0272
- Banking 0770
- Management 0454
- Marketing 0338

Canadian Studies 0385
Economics
- General 0501
- Agricultural 0503
- Commerce-Business 0505
- Finance 0508
- History 0509
- Labor 0510
- Theory 0511

Folklore 0358
Geography 0366
Gerontology 0351
History
- General 0578
- Ancient 0579
- Medieval 0581
- Modern 0582
- Black 0328
- African 0331
- Asia, Australia and Oceania 0332
- Canadian 0334
- European 0335
- Latin American 0336
- Middle Eastern 0333
- United States 0337

History of Science 0585
Law 0398
Political Science
- General 0615
- International Law and Relations 0616
- Public Administration 0617

Recreation 0814
Social Work 0452
Sociology
- General 0626
- Criminology and Penology ... 0627
- Demography 0938
- Ethnic and Racial Studies 0631
- Individual and Family Studies 0628
- Industrial and Labor Relations 0629
- Public and Social Welfare 0630
- Social Structure and Development 0700
- Theory and Methods 0344

Transportation 0709
Urban and Regional Planning 0999
Women's Studies 0453

THE SCIENCES AND ENGINEERING

BIOLOGICAL SCIENCES
Agriculture
- General 0473
- Agronomy 0285
- Animal Culture and Nutrition 0475
- Animal Pathology 0476
- Food Science and Technology 0359
- Forestry and Wildlife 0478
- Plant Culture 0479
- Plant Pathology 0480
- Plant Physiology 0817
- Range Management 0777
- Wood Technology 0746

Biology
- General 0306
- Anatomy 0287
- Biostatistics 0308
- Botany 0309
- Cell 0379
- Ecology 0329
- Entomology 0353
- Genetics 0369
- Limnology 0793
- Microbiology 0410
- Molecular 0307
- Neuroscience 0317
- Oceanography 0416
- Physiology 0433
- Radiation 0821
- Veterinary Science 0778
- Zoology 0472

Biophysics
- General 0786
- Medical 0760

EARTH SCIENCES
Biogeochemistry 0425
Geochemistry 0996
Geodesy 0370
Geology 0372
Geophysics 0373
Hydrology 0388
Mineralogy 0411
Paleobotany 0345
Paleoecology 0426
Paleontology 0418
Paleozoology 0985
Palynology 0427
Physical Geography 0368
Physical Oceanography 0415

HEALTH AND ENVIRONMENTAL SCIENCES
Environmental Sciences 0768
Health Sciences
- General 0566
- Audiology 0300
- Chemotherapy 0992
- Dentistry 0567
- Education 0350
- Hospital Management 0769
- Human Development 0758
- Immunology 0982
- Medicine and Surgery 0564
- Mental Health 0347
- Nursing 0569
- Nutrition 0570
- Obstetrics and Gynecology .. 0380
- Occupational Health and Therapy 0354
- Ophthalmology 0381
- Pathology 0571
- Pharmacology 0419
- Pharmacy 0572
- Physical Therapy 0382
- Public Health 0573
- Radiology 0574
- Recreation 0575
- Speech Pathology 0460
- Toxicology 0383

Home Economics 0386

PHYSICAL SCIENCES
Pure Sciences
Chemistry
- General 0485
- Agricultural 0749
- Analytical 0486
- Biochemistry 0487
- Inorganic 0488
- Nuclear 0738
- Organic 0490
- Pharmaceutical 0491
- Physical 0494
- Polymer 0495
- Radiation 0754

Mathematics 0405
Physics
- General 0605
- Acoustics 0986
- Astronomy and Astrophysics 0606
- Atmospheric Science 0608
- Atomic 0748
- Electronics and Electricity 0607
- Elementary Particles and High Energy 0798
- Fluid and Plasma 0759
- Molecular 0609
- Nuclear 0610
- Optics 0752
- Radiation 0756
- Solid State 0611

Statistics 0463

Applied Sciences
Applied Mechanics 0346
Computer Science 0984
Engineering
- General 0537
- Aerospace 0538
- Agricultural 0539
- Automotive 0540
- Biomedical 0541
- Chemical 0542
- Civil 0543
- Electronics and Electrical 0544
- Heat and Thermodynamics ... 0348
- Hydraulic 0545
- Industrial 0546
- Marine 0547
- Materials Science 0794
- Mechanical 0548
- Metallurgy 0743
- Mining 0551
- Nuclear 0552
- Packaging 0549
- Petroleum 0765
- Sanitary and Municipal 0554
- System Science 0790

Geotechnology 0428
Operations Research 0796
Plastics Technology 0795
Textile Technology 0994

PSYCHOLOGY
General 0621
Behavioral 0384
Clinical 0622
Developmental 0620
Experimental 0623
Industrial 0624
Personality 0625
Physiological 0989
Psychobiology 0349
Psychometrics 0632
Social 0451

Nom _____

Dissertation Abstracts International est organisé en catégories de sujets. Veuillez s.v.p. choisir le sujet qui décrit le mieux votre thèse et inscrivez le code numérique approprié dans l'espace réservé ci-dessous.

SUJET	CODE DE SUJET

U·M·I

Catégories par sujets

HUMANITÉS ET SCIENCES SOCIALES

COMMUNICATIONS ET LES ARTS

Architecture 0729
Beaux-arts 0357
Bibliothéconomie 0399
Cinéma 0900
Communication verbale 0459
Communications 0708
Danse 0378
Histoire de l'art 0377
Journalisme 0391
Musique 0413
Sciences de l'information 0723
Théâtre 0465

ÉDUCATION

Généralités 515
Administration 0514
Art 0273
Collèges communautaires 0275
Commerce 0688
Économie domestique 0278
Éducation permanente 0516
Éducation préscolaire 0518
Éducation sanitaire 0680
Enseignement agricole 0517
Enseignement bilingue et multiculturel 0282
Enseignement industriel 0521
Enseignement primaire. 0524
Enseignement professionnel 0747
Enseignement religieux 0527
Enseignement secondaire 0533
Enseignement spécial 0529
Enseignement supérieur 0745
Évaluation 0288
Finances 0277
Formation des enseignants 0530
Histoire de l'éducation 0520
Langues et littérature 0279
Lecture 0535
Mathématiques 0280
Musique 0522
Orientation et consultation 0519
Philosophie de l'éducation 0998
Physique 0523
Programmes d'études et enseignement 0727
Psychologie 0525
Sciences 0714
Sciences sociales 0534
Sociologie de l'éducation 0340
Technologie 0710

LANGUE, LITTÉRATURE ET LINGUISTIQUE

Langues
- Généralités 0679
- Anciennes 0289
- Linguistique 0290
- Modernes 0291

Littérature
- Généralités 0401
- Anciennes 0294
- Comparée 0295
- Médiévale 0297
- Moderne 0298
- Africaine 0316
- Américaine 0591
- Anglaise 0593
- Asiatique 0305
- Canadienne (Anglaise) 0352
- Canadienne (Française) 0355
- Germanique 0311
- Latino-américaine 0312
- Moyen-orientale 0315
- Romane 0313
- Slave et est-européenne 0314

PHILOSOPHIE, RELIGION ET THÉOLOGIE

Philosophie 0422
Religion
- Généralités 0318
- Clergé 0319
- Études bibliques 0321
- Histoire des religions 0320
- Philosophie de la religion 0322

Théologie 0469

SCIENCES SOCIALES

Anthropologie
- Archéologie 0324
- Culturelle 0326
- Physique 0327

Droit 0398
Économie
- Généralités 0501
- Commerce-Affaires 0505
- Économie agricole 0503
- Économie du travail 0510
- Finances 0508
- Histoire 0509
- Théorie 0511

Études américaines 0323
Études canadiennes 0385
Études féministes 0453
Folklore 0358
Géographie 0366
Gérontologie 0351
Gestion des affaires
- Généralités 0310
- Administration 0454
- Banques 0770
- Comptabilité 0272
- Marketing 0338

Histoire
- Histoire générale 0578
- Ancienne 0579
- Médiévale 0581
- Moderne 0582
- Histoire des noirs 0328
- Africaine 0331
- Canadienne 0334
- États-Unis 0337
- Européenne 0335
- Moyen-orientale 0333
- Latino-américaine 0336
- Asie, Australie et Océanie 0332

Histoire des sciences 0585
Loisirs 0814
Planification urbaine et régionale 0999
Science politique
- Généralités 0615
- Administration publique 0617
- Droit et relations internationales 0616

Sociologie
- Généralités 0626
- Aide et bien-âtre social 0630
- Criminologie et établissements pénitentiaires 0627
- Démographie 0938
- Études de l' individu et de la famille 0628
- Études des relations interethniques et des relations raciales 0631
- Structure et développement social 0700
- Théorie et méthodes. 0344
- Travail et relations industrielles 0629

Transports 0709
Travail social 0452

SCIENCES ET INGÉNIERIE

SCIENCES BIOLOGIQUES

Agriculture
- Généralités 0473
- Agronomie. 0285
- Alimentation et technologie alimentaire 0359
- Culture 0479
- Élevage et alimentation 0475
- Exploitation des pêturages ... 0777
- Pathologie animale 0476
- Pathologie végétale 0480
- Physiologie végétale 0817
- Sylviculture et faune 0478
- Technologie du bois 0746

Biologie
- Généralités 0306
- Anatomie 0287
- Biologie (Statistiques) 0308
- Biologie moléculaire 0307
- Botanique 0309
- Cellule 0379
- Écologie 0329
- Entomologie 0353
- Génétique 0369
- Limnologie 0793
- Microbiologie 0410
- Neurologie 0317
- Océanographie 0416
- Physiologie 0433
- Radiation 0821
- Science vétérinaire 0778
- Zoologie 0472

Biophysique
- Généralités 0786
- Medicale 0760

SCIENCES DE LA TERRE

Biogéochimie 0425
Géochimie 0996
Géodésie 0370
Géographie physique 0368
Géologie 0372
Géophysique 0373
Hydrologie 0388
Minéralogie 0411
Océanographie physique 0415
Paléobotanique 0345
Paléoécologie 0426
Paléontologie 0418
Paléozoologie 0985
Palynologie 0427

SCIENCES DE LA SANTÉ ET DE L'ENVIRONNEMENT

Économie domestique 0386
Sciences de l'environnement 0768
Sciences de la santé
- Généralités 0566
- Administration des hipitaux .. 0769
- Alimentation et nutrition 0570
- Audiologie 0300
- Chimiothérapie 0992
- Dentisterie 0567
- Développement humain 0758
- Enseignement 0350
- Immunologie 0982
- Loisirs 0575
- Médecine du travail et thérapie 0354
- Médecine et chirurgie 0564
- Obstétrique et gynécologie ... 0380
- Ophtalmologie 0381
- Orthophonie 0460
- Pathologie 0571
- Pharmacie 0572
- Pharmacologie 0419
- Physiothérapie 0382
- Radiologie 0574
- Santé mentale 0347
- Santé publique 0573
- Soins infirmiers 0569
- Toxicologie 0383

SCIENCES PHYSIQUES

Sciences Pures

Chimie
- Généralités 0485
- Biochimie 487
- Chimie agricole 0749
- Chimie analytique 0486
- Chimie minérale 0488
- Chimie nucléaire 0738
- Chimie organique 0490
- Chimie pharmaceutique 0491
- Physique 0494
- PolymÇres 0495
- Radiation 0754

Mathématiques 0405
Physique
- Généralités 0605
- Acoustique 0986
- Astronomie et astrophysique 0606
- Électronique et électricité 0607
- Fluides et plasma 0759
- Météorologie 0608
- Optique 0752
- Particules (Physique nucléaire) 0798
- Physique atomique 0748
- Physique de l'état solide 0611
- Physique moléculaire 0609
- Physique nucléaire 0610
- Radiation 0756

Statistiques 0463

Sciences Appliqués Et Technologie

Informatique 0984
Ingénierie
- Généralités 0537
- Agricole 0539
- Automobile 0540
- Biomédicale 0541
- Chaleur et ther modynamique 0348
- Conditionnement (Emballage) 0549
- Génie aérospatial 0538
- Génie chimique 0542
- Génie civil 0543
- Génie électronique et électrique 0544
- Génie industriel 0546
- Génie mécanique 0548
- Génie nucléaire 0552
- Ingénierie des systämes 0790
- Mécanique navale 0547
- Métallurgie 0743
- Science des matériaux 0794
- Technique du pétrole 0765
- Technique minière 0551
- Techniques sanitaires et municipales 0554
- Technologie hydraulique 0545

Mécanique appliquée 0346
Géotechnologie 0428
Matières plastiques (Technologie) 0795
Recherche opérationnelle 0796
Textiles et tissus (Technologie) 0794

PSYCHOLOGIE

Généralités 0621
Personnalité 0625
Psychobiologie 0349
Psychologie clinique 0622
Psychologie du comportement 0384
Psychologie du développement .. 0620
Psychologie expérimentale 0623
Psychologie industrielle 0624
Psychologie physiologique 0989
Psychologie sociale 0451
Psychométrie 0632

OVERLOAD CONTROL OF A FINITE CAPACITY, SINGLE SERVER QUEUE WITH VACATIONS

BY

EDWIN B. MARTIN

A Thesis submitted to the Faculty of Graduate Studies of the University of Manitoba in partial fulfillment of the requirements of the degree of

MASTER OF SCIENCE

© 1993

Permission has been granted to the LIBRARY OF THE UNIVERSITY OF MANITOBA to lend or sell copies of this thesis, to the NATIONAL LIBRARY OF CANADA to microfilm this thesis and to lend or sell copies of the film, and LIBRARY MICROFILMS to publish an abstract of this thesis.

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or other-wise reproduced without the author's written permission.

ABSTRACT

This thesis is motivated by a goal of network designers to utilize all available resources as efficiently as possible. A major problem encountered in networks are lost jobs due to finite buffer capacities. To combat this problem, an overload control strategy applied to polling systems is proposed.

The strategy works with finite capacity buffers and utilizes a control limit which acts to call the server to queues having length greater than the limit. A server transfer will be prevented if the queue being served has length greater than the control limit. As the strategy is to be applied to polling systems, the network is approximated by a single server vacation model. In this model, arrival and service rates are exponential, and vacation rates are phase type.

Prior to solving the single server vacation model, information regarding vacation behaviour is required. This information is gained through the use of a computer simulation. A program was written to simulate the operation of a three queue polling system.

The state space of the network is derived and the transition rate matrix for the general case is developed. Then, the resulting linear balance equations are converted to transition probabilities and solved by an iterative algorithm. A **FORTRAN** program was created to solve the steady state probabilities for the general system.

This program also calculates performance measures related to vacation behaviour, system occupancy and loss frequency. It also provides results for the "no control" case which is used to determine the effectiveness of the overload control strategy.

The results of the analysis confirm that this overload control strategy is an effective way of improving network performance. Loss frequency and overall system occupancy were seen to decrease in most instances. Areas for further research in this area are also provided.

ACKNOWLEDGEMENTS

I would like to thank Dr. A.S. Alfa for his support and assistance while I was performing this research.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

1.1 Problem to be Investigated

This thesis is motivated by a goal of network designers to utilize all available resources as efficiently as possible. A major problem encountered in networks are lost jobs during periods of heavy traffic arrivals. It is important to realize that arriving jobs line up in a buffer of finite capacity. When any of these buffers become full, further arrivals are lost. Inevitably, these lost jobs must be retransmitted causing delays.

Another concern of designers is the method in which the server is allocated work. Quite often, the network is configured as a polling system. Polling systems consist of a series of finite capacity queues that are tended to, in sequence, by a single server. If the server is assigned more work than it can handle for any length of time, the potential for lost jobs is increased. These times are called overload periods and when they occur it is beneficial to distribute the server among all queues in such a way as to minimize the number of incurred losses as previously described.

In most polling systems, the server works on one queue for a predetermined length of time according to its service discipline. More common service methods are *exhaustive, gated* or *limited*. In an exhaustive discipline, the server will not leave a queue until it becomes empty. The gated rule has the server placing a gate behind the last job present upon initial arrival. These jobs are served but any new arrivals must wait for service on the next cycle. A server working under the limited discipline will

remain in a queue until it is emptied or a preset time limit is exceeded. Once either of these conditions are met, the server moves on.

Thus, the server continually cycles through the queues, performing service on those that contain jobs. As such, polling systems are used in communication networks to provide a 'fair' service schedule to systems with many queues but just one server. Unfortunately, these three service disciplines only take into consideration the status of the queue presently being served and neglect the other queues within the polling system. As such, even if the length of one of the other buffers is nearing capacity, it must wait its turn for service. In this light, if an overload control mechanism is implemented, the problem of job loss may be reduced. An illustration of a finite capacity polling system is provided in Figure 1.1-1.

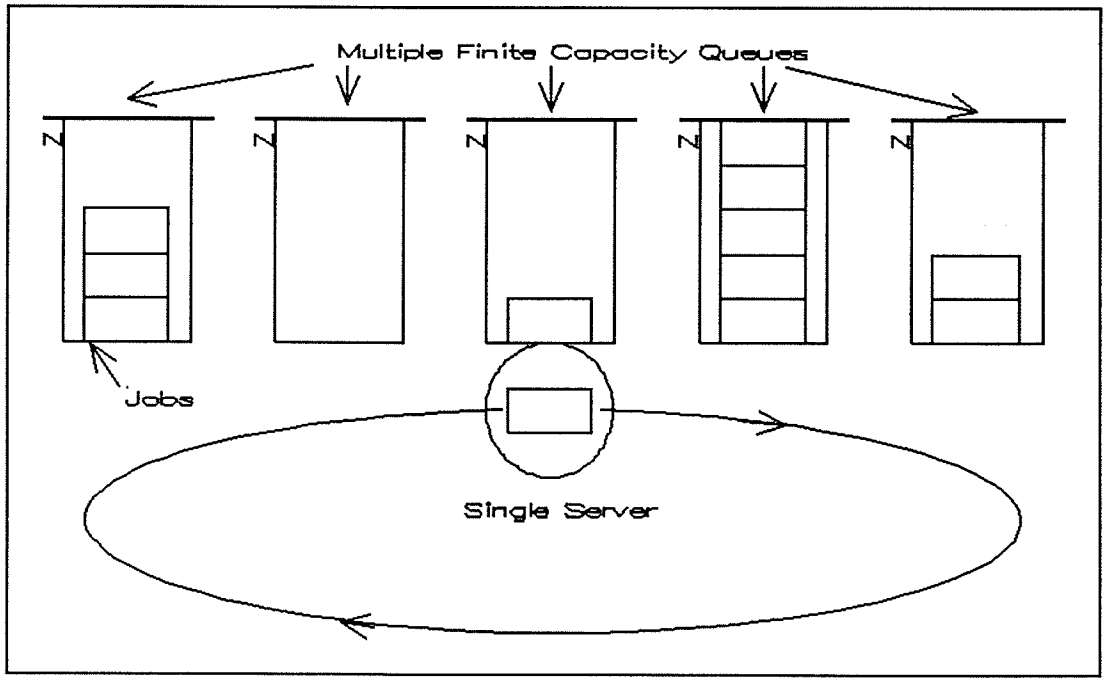


Figure 1.1-1: Finite Capacity Polling System

The purpose of this thesis is to evaluate an overload control mechanism for use in such polling systems. This mechanism is a control limit, K, which is less than the buffer capacity, N. It acts to call the server to any queue where the number of waiting customers exceeds K. Similarly, it prevents the server from being called away if the queue being served presently contains more than K jobs.

1.2 Scope of Analysis

The intent of this research is to determine whether this overload control mechanism has merit for implementation in polling systems. To accomplish this objective, the system is modelled as a finite capacity, single server queue (See Figure 1.2-1) with vacations. The reasons for modelling the control strategy in this manner are described in a later chapter.

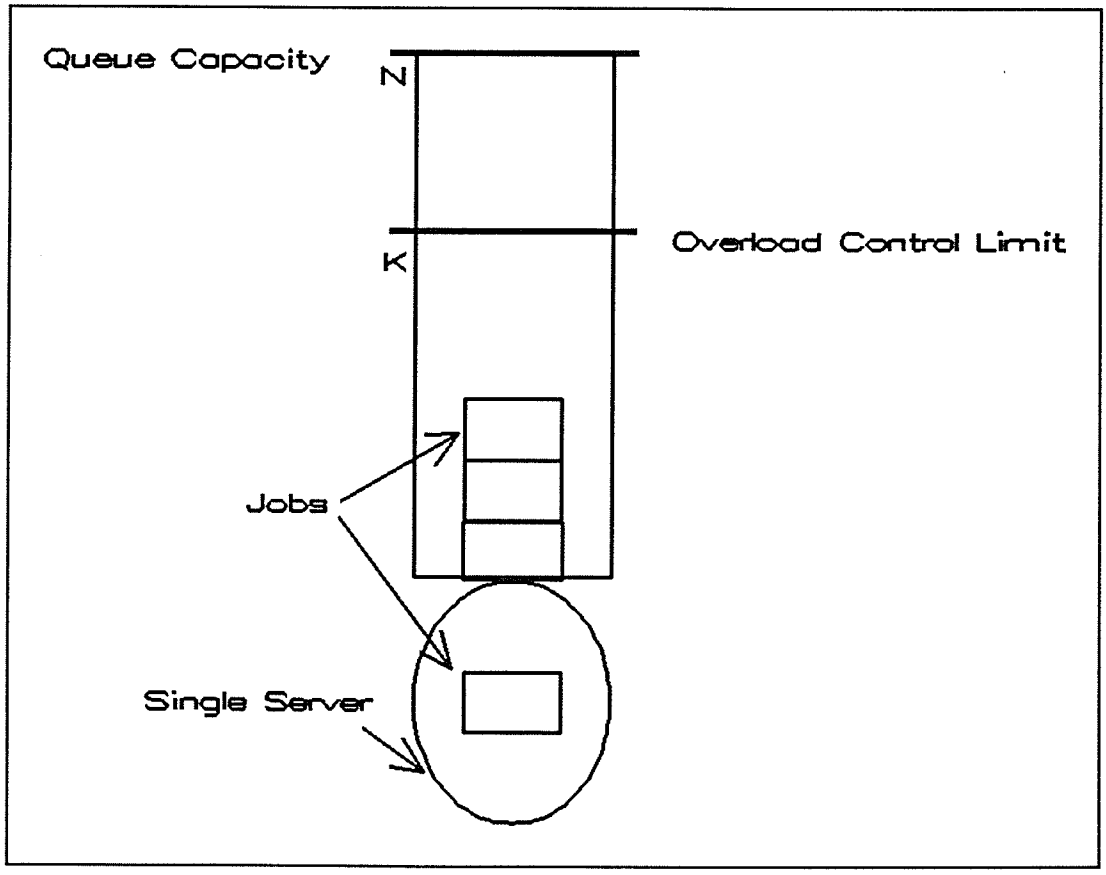


Figure 1.2-1: Single Server Queue with Overload Control Limit

A symmetric system (ie. All queues in the polling system exhibit identical arrival and service characteristics) is considered and the model is solved for the steady state. Upon solution, a number of performance measures are calculated. These relate to system occupancy, vacation behaviour and loss frequency. They provide the basis for evaluating the control scheme.

The remainder of this thesis will be presented as follows. In the next chapter, a review of the relevant literature will be performed. Situations that have been modelled by polling systems and their analysis will be provided. Also, the review will describe the approximation of polling systems by single server vacation models. A variety of previously studied overload control mechanisms will be mentioned. The conclusion of this chapter indicates the contribution of this thesis to the field of congestion control.

Chapter 3 provides a detailed explanation as to the inner workings of the proposed overload control strategy. In this chapter, the reaction of the controlled polling system to all situations are described with a corresponding explanation for the vacation model. By doing so, the validity of the vacation model approximation will be apparent.

Chapter 4 will deal with the actual model construction. It begins with a brief description of Markov processes. Then, the method used in deriving the possible system states is presented. This is followed by a discussion on how the transition rate matrix and the balance equations that approximate the network are determined. Finally, the actual method used in finding the steady state of the process is explained.

Before the model can be solved for the steady state, a general understanding of vacation behaviour must be gained. This is necessary to be able to provide appropriate values for vacation parameters as input into the single server model. This knowledge is acquired by simulating the polling system on a small scale and observing its behaviour. This simulation analysis and its results are the topic of Chapter 5.

In Chapter 6, the analysis of the controlled single server vacation model is performed and performance measures calculated. The measures for the overload control strategy are compared to results obtained for a 'no control' case to provide a rational basis for determining the effectiveness of the mechanism.

As this thesis performs new research in its area, further research is both desirable and recommended. A number of directions that further research can take are proposed. Finally, the conclusions of this thesis are presented.

CHAPTER 2
LITERATURE REVIEW

2.1 Analysis of Polling Systems

This section makes reference to research that deals with the analysis of polling systems. Comprehensive work in this area were carried out by *Takagi [1], [2].* In his works, Takagi describes a number of different types of polling systems and provides procedures for solving them under exhaustive, limited and gated service disciplines. The system modelled in this thesis differs from Takagi's work in that his requires the assumption of infinite queues. While this assumption is valid in systems where delays are the primary concern, it is of great benefit to perform analysis on polling systems with finite capacity queues to gain understanding of network losses.

A description of the capabilities and limitations of different polling models is performed in a paper by *Levy [3].* In the paper, polling applications in many fields, including token ring networks, random-access protocols, robotics and manufacturing are considered. This work emphasizes the wide variety of situations where polling models are applicable.

One example of how polling systems can represent communication networks is in the work by *Lee* and *Sengupta [4].* In their paper, a cyclic server queue is analyzed with limited service. It is motivated by the pipeline polling protocol in satellite communications. The solution method that they propose is an approximation for the queue length and sojourn-time distributions. Due to the long distances that signals have

to travel, the authors concentrate on a service strategy that efficiently handles the propagation delays that these distances can cause. In these networks, job losses resulting from full buffers are not as important an issue and the approximation method of *Lee* and *Sengupta* assumes infinite capacity queues.

In a paper by *Levy* ***[5]***, a binomial-gated service discipline is proposed for use in cyclic polling systems. The work allows for assigning priorities to each queue within the system. He performs an analysis of this discipline to predict customer delay and to effectively optimize the system performance according to the customer needs. His infinite capacity model is for implementation in token ring networks and other communication systems.

A study of finite capacity polling systems can be found in *Tran-Gia* ***[6]***. In this work, general input processes are considered with a limited service discipline of one message per cycle. The importance of job losses is noted and blocking probability is calculated based on the use of efficient discrete convolution operations using fast convolution algorithms.

2.2 Vacation Models for Analysing Polling Systems

Vacation models have been used in the relevant literature to solve many networks, including polling systems. This section will explain the thinking behind solving polling systems through vacation models and will describe some of the research in this area.

A major discussion of vacation models can be found in *Doshi [7]*. Consider a polling system as illustrated in Figure 2.2-1. This system can be divided into two groups. One consists of any single queue (referred to as the target queue), while the other contains all other queues. Then, one of two situations can occur; Either the server is performing work on the target queue or it is not. If it is not serving the target queue, as is the case in Figure 2.2-1, then for jobs waiting in this queue the server seems to be on vacation. When the server next returns to the target queue the vacation is terminated and a busy period commences. By analysing the behaviour of the target queue, the overall behaviour of the polling system can be approximated.

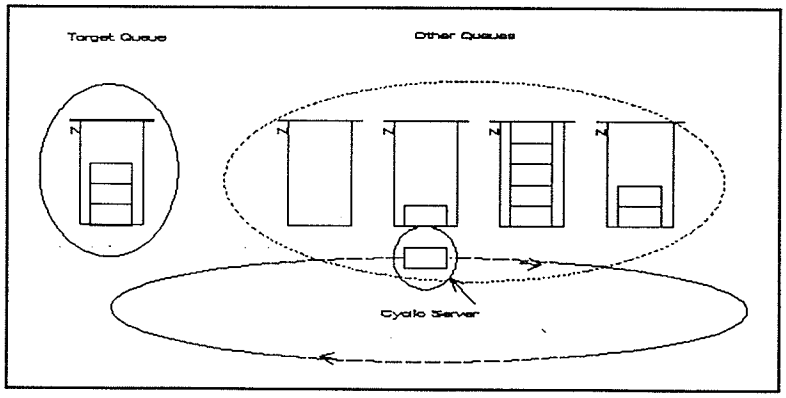


Figure 2.2-1: Grouping For Vacation Models

In the paper by *Doshi*, a number of situations are described where the server works on primary and secondary customers in the context of computer, communication and production systems. The research contains a decomposition technique that can be used to solve a number of related problems. Among these problems are *M/G/1* queuing systems with finite buffers. This section of the paper states that the decomposition technique can be extended for finite systems. It does not, however, contain reference to overload control strategies.

The above *M/G/1* terminology follows the standard *Kendall-Lee* ***[8]*** notation for queuing systems. In it, queuing systems are described in the following way:

A/B/C/D/E/F

where,

A - Nature of the arrival process
B - Nature of the service process
C - Number of parallel servers
D - Queue discipline (Usually omitted if using General queue discipline)
E - Maximum allowable queue length (Usually omitted if infinite)
F - Size of population from which jobs are drawn (Usually omitted if infinite)

Although analysis of infinite capacity queues dominate the literature, there has been a certain amount of research on finite capacity queues. In the work of *Lee* ***[9]***, an *M/G/1/N* queue is studied to determine the performance of a processor with a cyclic scheduling algorithm and exhaustive service discipline. The cyclic nature of the server gives rise to a solution method utilizing vacation times. They recognize the importance of job losses and their solution derives both queue length distribution and blocking

probability. Their model is studied using embedded Markov chain and supplementary variable techniques.

Another analysis of finite capacity queues of interest is by *LaMaire **[10],[11]***. LaMaire develops a general service discipline that encompasses several previously analyzed service disciplines. His solution method is similar to that in ***[9]*** and queue length distributions, mean waiting times and blocking probabilities are developed. While the solution method is valid for a variety of service disciplines it does not include any congestion control mechanism.

2.3 Finite Capacity Systems

There has been considerable research done on general finite capacity systems. This section will describe a few of these works. First to be reviewed is a paper by *Guerin* and *Lien* ***[12]***. In it, a system of multiple, finite capacity queues are modelled. If a job arrives to a full queue, it overflows into a secondary finite capacity pool. They recognize the importance of minimizing job losses and analyze this mechanism over a variety of offered loads for blocking probability.

Another finite capacity system where losses are important is studied by *Hebuterne* and *Gravey* ***[13]***. They model an ATM network that allows higher priority jobs to take the place of lower priority jobs when the queue reaches capacity. In this way, although the total number of lost jobs remains the same, higher priority jobs are saved at the expense of others. The model is of an *M/G/1* type and they note the effects of reducing buffer sizes and increasing admissible load.

Towsley, Sparaggis and *Cassandras* ***[14]***, consider the problem of routing jobs to parallel queues with identical exponential servers and unequal finite buffer capacities. They prove that by joining the shortest queue, the system is optimized when taking into account holding and blocking costs. This paper, again, recognizes the importance of minimizing job losses in finite capacity systems.

2.3 Optimization of Vacation Models

A limited amount of research has been performed in the area of optimization of vacation models. This section describes a couple of published papers in this area. The first work considered is by *Harris* and *Marchal* ***[15]***. In their work, an *M/G/1* queue is used to model a manufacturing situation where a single server (robot) services different types of customers (operations). A specific customer is defined as the target queue and a vacation model results. They utilize linear programming to compute the optimal server-vacation policy. The formulation considers both holding and switchover costs.

Another paper in this area was written by *Kella* ***[16]***. He considers a model where the decision whether or not to take a new vacation, when the system is empty, depends on the number of vacations already taken through a random outcome. An infinite capacity *M/G/1* queue is optimized under expected long-run average cost criterion. Holding and setup costs as well as a reward for being on vacation are considered. The optimization problem results in an infinite dimensional fractional program which, when solved, yields a deterministic control policy.

2.4 Research on Overload Control

Overload or congestion control of networks has been the subject of considerable research. A multitude of strategies have been developed and analyzed for an equally diverse number of situations. This section describes a few of the works in this area that have interest to this thesis.

Discussion of overload control research begins with a paper by *Neuts **[17]*** that analyzes an *M/G/1* queue with infinite capacity. In this work, an overload control limit, similar to that used in this thesis, is defined. It acts to reduce the arrival rate of customers when the queue length exceeds the limit and return to normal when the level drops back below the control limit. It is realized that the reduction in customer arrival rate will not be instantaneous and a period of random duration elapses before the desired arrival rate is attained. During that period, the arrival rate is only partially reduced or not at all. An embedded Markov chain approach is used to perform an analysis of queue length. Mention is made of analysing finite capacity queues using multiple control levels via Markov renewal theory, but this avenue is not explored further.

Another congestion control mechanism involving restricting network access is studied by *van As **[18]***. This paper deals with packet switching networks and assigns priority to customers depending upon how much service they have received (ie. The more service a packet has received, the higher priority it is assigned). Thus, job losses will be limited to lower priority (just arriving) packets as they will not be allowed access

to the network under conditions of overload. The system is solved for both steady state and transient cases. In the transient case, the system is exposed to a rectangular overload peak and performance of the mechanism is found by observing behaviour over time. Performance measures of interest include loss probability, throughput and mean system occupancy. The relation of *van As* paper to this thesis is its solution method for the steady state case. This method involves describing the system state structure, creating a transition rate matrix and solving for the steady state. This procedure is fully explained in chapter 4.

An overload control mechanism utilizing two separate control limits is studied by *Li [19]*. His research contains analysis of *M/PH/1/N* and *PH/M/1/N* queues with solution for queue length distribution. The control limits L_1 and L_2 are defined such that $L_1 \leq L_2 \leq N$. A buffer overload period is defined as the time period when buffer content exceeds level L_2 until it drops back below level L_1. The time interval between two subsequent overload periods is defined as the buffer underload period. During the overload period, either the service rate is increased or the job arrival rate is decreased. The analytical method divides the controlled process into two alternating transient birth-death and quasi-birth-death subprocesses, by only observing some selected transitions. Such a division enables representation of the steady-state distribution of the controlled process by the results of mean sojourn time in the two transient processes. While the strategy is similar to this thesis in that control limits are defined, his work does not extend to polling systems.

2.5 Conclusions on the Literature Review

As this chapter has indicated, substantial research has been performed in the areas of polling systems and congestion control. It has also pointed out two major shortcomings that this thesis will attend to. They are:

1) A lack of research on finite capacity queues.

2) Analysing overload control in a polling system environment.

In the first point, it has been shown that most literature to date analyzes infinite capacity queues. While queues of this type are suitable for modelling systems where delays are a primary concern, they ignore the problem of lost customers.

As for the second issue, this chapter has shown that considerable work has been done in both overload control and the analysis of polling systems. Unfortunately, little has been done in the area of controlling polling systems under conditions of overload.

CHAPTER 3
MODEL DESCRIPTION

3.1 The Overload Control Strategy in a Polling System

The new overload control strategy being evaluated in this thesis has application to polling systems. Its major advantage is its ability to use information on the status of other queues in the system to make decisions on which queue the server should be performing work in. This section will describe the overload control mechanism as it is applied to polling systems and will provide examples on how the system will react under various situations.

For descriptive purposes, the following explanation of the overload control mechanism is generalized for an asymmetric case. All actual analysis, however, is limited to a symmetric system. Consider a polling system consisting of a single server performing work on M queues. Each of the queues has a finite capacity N_i $(1 < i \leq M; 0 < N_i < \infty)$. An overload control limit, K_i $(1 < i \leq M; 0 < K \leq N_i)$, is introduced to reduce instances of loss that occur when a job arrives to a full buffer. The control strategy is based on an exhaustive service discipline. That is, barring any overload conditions in the system, if the server is operating on queue i, then it will remain operating on queue i until it becomes empty. When this occurs, it begins servicing queue $i+1$. After completing service on all jobs in queue X, the server moves to queue 1 and begins the cycle again. An example of normal operating procedure is illustrated in Figure 3.1-1.

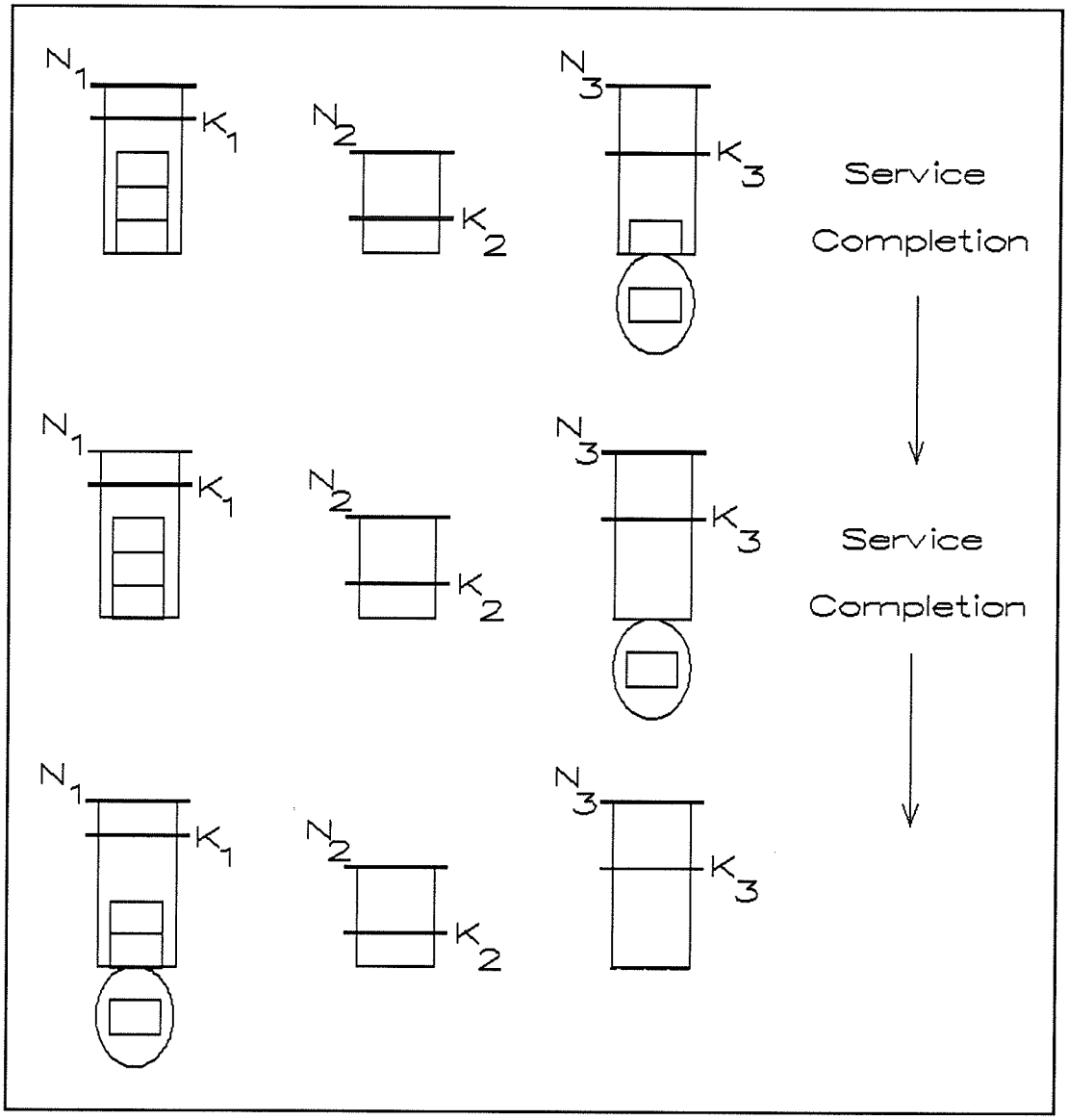


Figure 3.1-1: Normal Polling System Operation

As long as the length of each queue is less than its respective control limit, the polling system behaves in the manner described above. Once the level is exceeded, however, the polling system is in an overloaded state and the control mechanism comes into play. It acts to call the server to the queue exhibiting the overload behaviour as long

as the queue presently being served has length less than the control limit. In Figure 3.1-2, scenario *(a)* illustrates a situation where the server is called away, while *(b)* gives an example of a time when the server will not be called away.

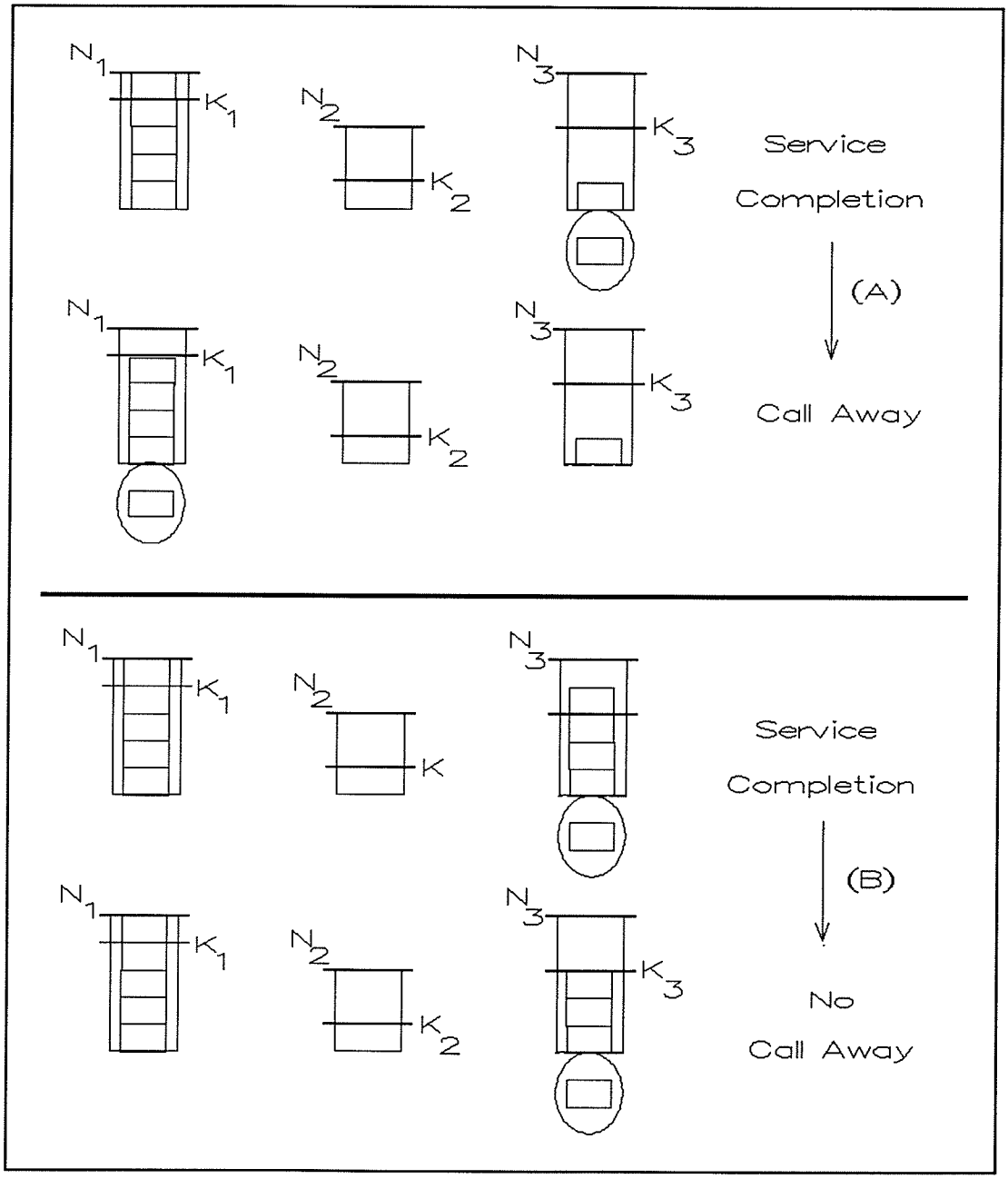


Figure 3.1-2: A Polling System in an Overload State

One other possibility exists. That is, while the queue being served is in overload, more than one other queue exceeds its control limit. In this event, once the present queue length becomes less than the control limit, the server will be transferred to the overloaded queue with length nearest buffer capacity. An example of this situation is provided in Figure 3.1-3. In the event that a tie occurs, the server will be relocated to the nearest queue.

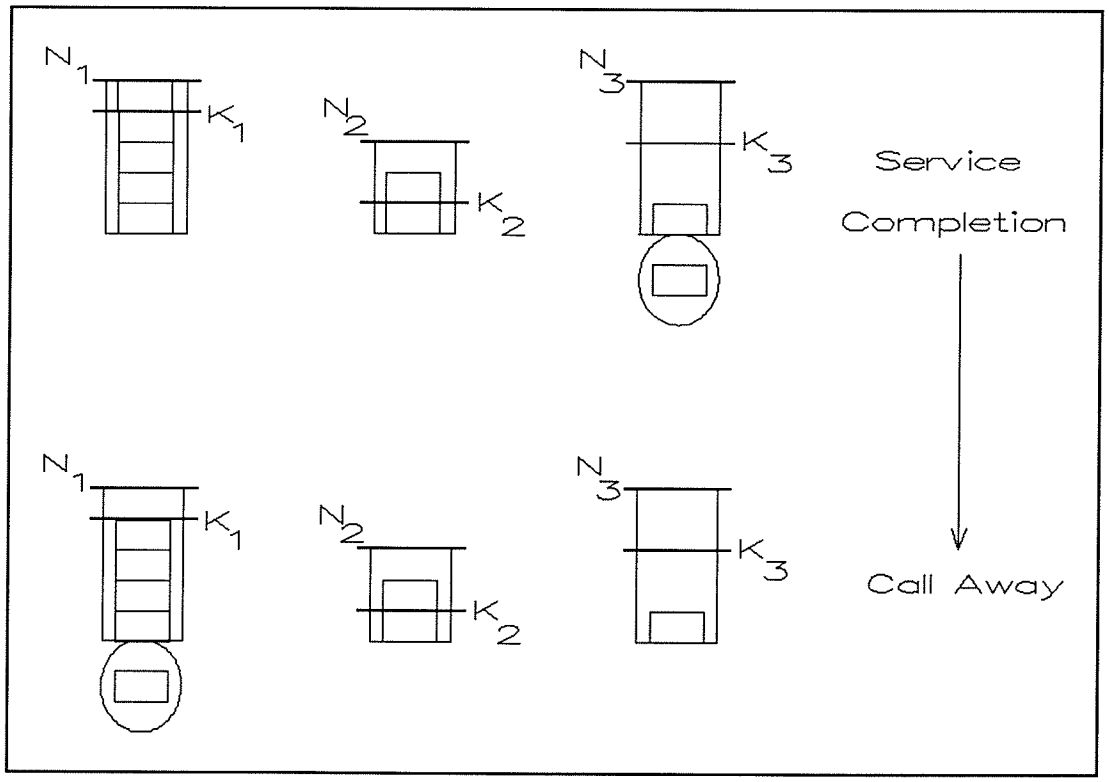


Figure 3.1-3: More Than One Queue in Overload

3.2 The Overload Control Strategy in a Vacation Model

The polling system being analyzed in this research will be solved as a single server vacation model. Thus, the overload control strategy described in section 3.1 must be converted to a form suitable for a vacation system. This section describes the strategy in terms of the vacation model, defining all parameters as required.

To allow for a single server vacation model, the polling system as investigated in this thesis must be divided into two groups. One group consists of a single queue termed the target queue, while the other contains all other queues (See Figure 2.2-1). By observing the behaviour of the target queue, an analysis of the polling system can result. The finite capacity target queue with overload control limit was shown in Figure 1.2-1. Using the aforementioned grouping, a vacation period begins the moment the server leaves the target queue. The vacation is terminated upon the next arrival of the server to the target queue.

When the system is not in an overloaded state, its operation is straightforward. The server remains in the target queue until the last customer is served and it becomes empty. At this time, the server is said to have left on a *normal vacation.* If the system is still empty upon vacation completion, the server takes another normal vacation. Otherwise, it begins service on any waiting jobs. The normal operating procedure is illustrated in Figure 3.2-1.

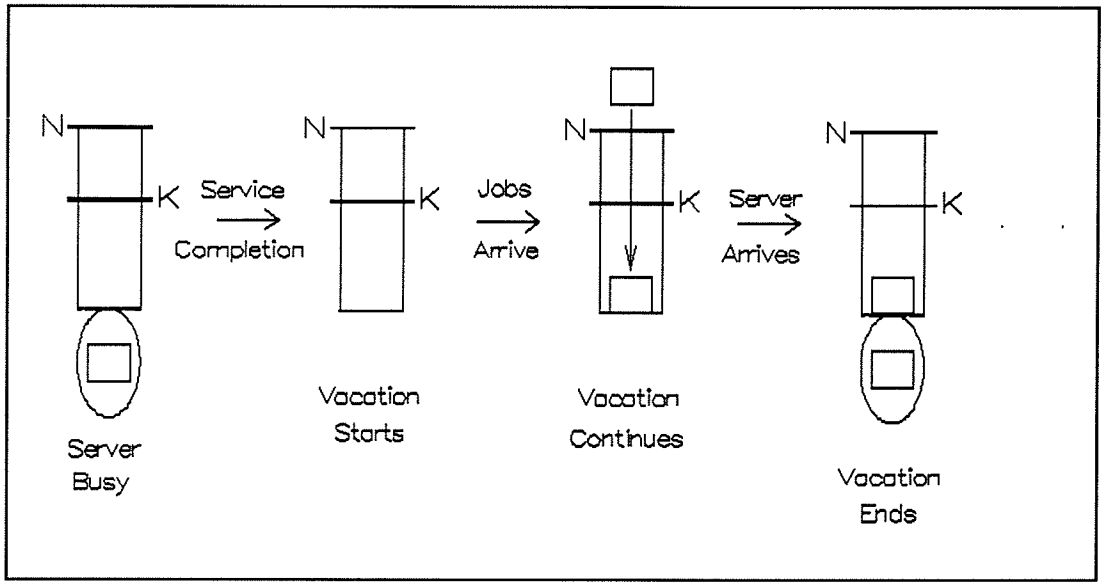


Figure 3.2-1: Normal Operating Procedure

Since this model is to be used in approximating polling systems, one should consider the situation where the length of one of the queues not being served exceeds the control limit, *K*. If this occurs, the server may leave the queue it is presently working on (upon the next service completion) to tend to the calling queue. This situation is incorporated into the single server vacation model by the implementation of an *emergency vacation*. To allow for emergency vacations to begin, a server call away probability, Ω, is defined. That is, upon service completion of any job the server may be called away on an emergency vacation with probability Ω or begin service on the next waiting job with probability *(1-Ω)*. These emergency vacations will not be initiated, however, if the number of customers waiting in the target queue exceeds the control limit value. The operation of the server call away parameter is depicted in Figure 3.2-2.

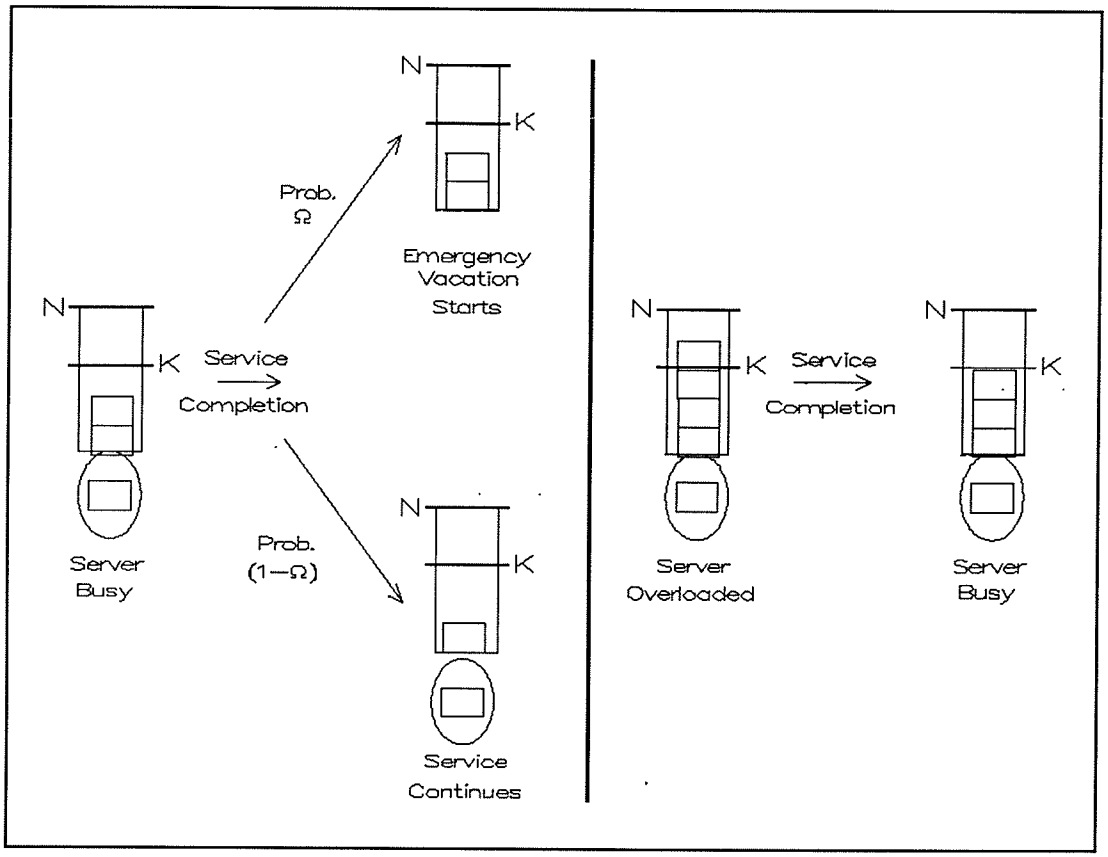


Figure 3.2-2: Server Call Away Operation

Another parameter, the server call back probability, must be defined to complete the conversion of the control strategy to a single server vacation model. This parameter comes into play when the server is on either a normal or emergency vacation and the length of the target queue becomes $\geq K$. If this occurs, the server may be called back to the target queue immediately depending on the state of the other buffers within the polling system. Thus, the server call back probability, θ, is defined as the probability that an arrival results in the server returning to the target queue. An arrival will have no effect on vacation operation with a probability of $(1-\theta)$. The operation of the server call back parameter is illustrated in Figure 3.2-3.

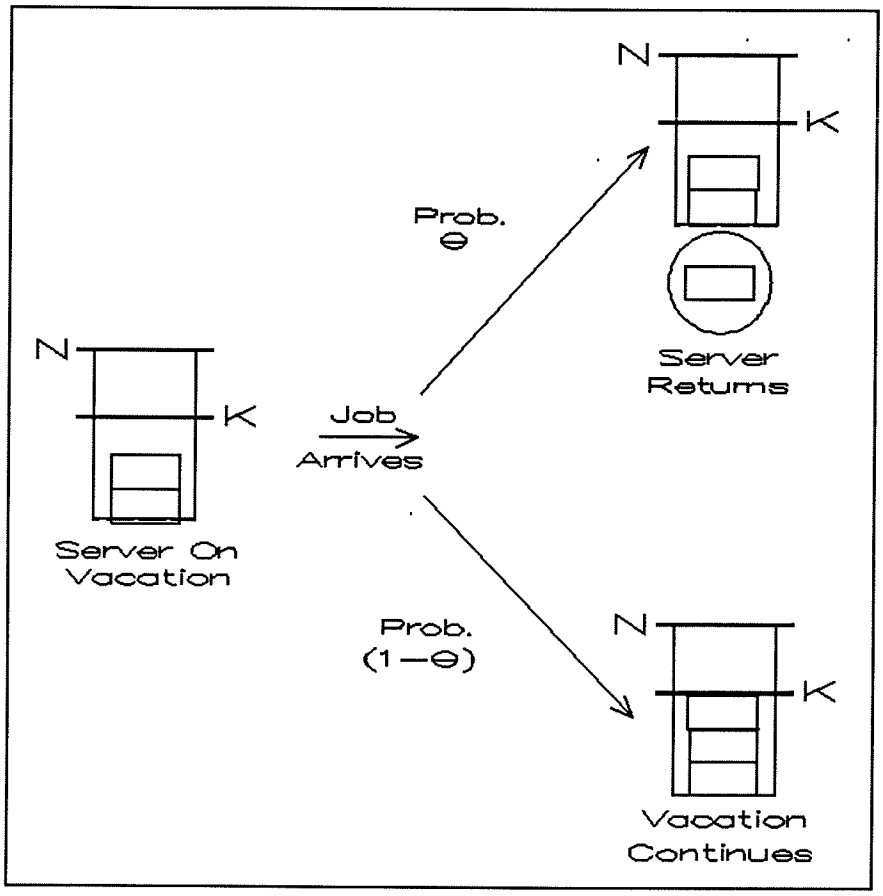


Figure 3.2-3: Server Call Back Operation

In the analysis of the single server vacation model, job arrivals per unit time are assumed to be Poisson with rate λ, and the service is exponential with rate μ_s. The vacation length varies, depending on the specific characteristics of the polling system to be approximated. Therefore, to make the model as general as possible, the vacation lengths are assumed to be phase type (with two phases) to take advantage of this distributions flexibility. For the normal vacation, the rates are denoted by μ_{V1} and μ_{V2} and for emergency vacations, μ_{E1} and μ_{E2}.

Vacations may start in either phase and the probability that it begins in phase one is ϕ. Upon completion of the 1^{st} phase, either the 2^{nd} phase begins or the vacation ends. Similarly, completion of the 2^{nd} phase results in the starting of the 1^{st} phase or vacation termination. These weights are defined as α and β for the 1^{st} and 2^{nd} phases, respectively. For a summary of the single server vacation model parameters, refer to Table 3.2-1.

Table 3.2-1: Single Server Vacation Model Parameters

Parameter Description	**Symbol**
Buffer Capacity	N
Overload Control Limit	K
Arrival Rate	λ
Service Rate	μ
Server Call Away Probability	Ω
Server Call Back Probability	θ
Normal Vacation Rate (Phase 1)	μ_{V1}
Normal Vacation Rate (Phase 2)	μ_{V2}
Emergency Vacation Rate (Phase 1)	μ_{E1}
Emergency Vacation Rate (Phase 2)	μ_{E2}
Prob. Vacation Starts In Phase 1	ϕ
Prob. Transfer From Phase 1 To 2	α
Prob. Transfer From Phase 2 To 1	β

CHAPTER 4
METHOD OF MODEL ANALYSIS

4.1 Markov Processes

The steady state solution to the single server vacation model is found by analysing the network as a Markov Process. This section will provide a brief overview of Markov Processes. It will also describe the steps necessary in solving the model for the steady state.

Markov Processes are those which obey a number of conditions, the most important being that they exhibit the *Markov Property*. The Markov Property simply states that the system is independent of its past. In other words, by knowing the present state of the system at time t, one can predict what the system will look like at time $t+1$. Furthermore, a maximum of one event is allowed to occur between times t and $t+1$.

Another important aspect of stationary Markov Processes is that it does not matter at what state the system begins in, it will eventually end up in the same steady state after a long period of time. It is this steady state probability distribution that needs to be solved for in this thesis. To determine the steady state distribution, the following balance equation must be solved:

$$Q^T * P = 0$$

where,

Q^T - Transition rate matrix
P - State probability vector

The balance equation is set equal to zero since in the steady state, the flow of jobs into a particular system state must equal the flow out of that same state. From this equation it is apparent that the first step in the solution procedure is to determine the state space of the system. Once this is found, it will be necessary to define all valid interstate transitions. This will give rise to a system of simultaneous equations that, when solved, provide the steady state probability distribution. The remaining sections of this chapter will describe the entire solution procedure.

4.2 Derivation of Possible System States

As referred to in Section 4.1, the state space of the system must be derived to allow for steady state solution. To do this, it must be decided what attributes uniquely define what state the system is in. This section deals with the state definition for the single server vacation model.

To be able to gather the relevant performance information, the state space of the single server vacation model must be defined in a way that keeps track of the length of the target queue whether or not the server is on vacation. Also, since normal and emergency vacation types will be of different duration, they must remain separate in the state space definition. Thus, the definition must be such that each state can answer the following three questions:

1) Is the server presently located in the target queue?

2) If it is not in the target queue, is it on a normal vacation or an emergency vacation?

3) How many jobs are currently waiting for service in the target queue?

The state space for the single server vacation model is defined to satisfy all of the above requirements. The state space, Δ, takes the form *(i,j)* or *(i,j,k)* depending on the location of the server:

$$\Delta = [(i,j); i=1; 1 \le j \le N] \bigcup [(i,j,k); i=0; 0 \le j \le N; k=1,2] \bigcup [(i,j,k); i=2; 1 \le j \le N; k=1,2]$$

where,

$i=0$:	Server on normal vacation
$i=1$:	Server in the target queue
$i=2$:	Server on emergency vacation
j:	Number of jobs in the target queue ($j=1,2,\ldots,N$)
k:	Server in k^{th} vacation phase ($k=1,2$)

Using this definition, it can be shown that exactly $5N+2$ states exist. Therefore, polling systems having large buffer capacities can be approximated by this model without having the state space become intractable. The $5N+2$ value is derived in Table 4.2-1.

Table 4.2-1: Breakdown of Possible System States

Location of Server	**Number of States**
In Target Queue	N
Phase 1 of Normal Vacation	N+1
Phase 2 of Normal Vacation	N+1
Phase 1 of Emergency Vacation	N
Phase 2 of Emergency Vacation	N
Total	**5N+2**

When solving the balance equation of Section 4.1, we desire the state space probability distribution. Therefore, in the remainder of the thesis, the probability that the system is in state (i,j,k) will be denoted by $P_{i,j,k}$.

4.3 Determination of Transition Rate Matrix & Balance Equations

Once all of the possible states have been defined, the next step involved in the model construction is to determine the transition rate matrix. This is necessary if movement between states is to be accurately predicted. From this matrix, it is possible to develop the appropriate balance equations. This section clarifies the method in which the transition rate matrix and the balance equations are derived.

In Section 4.1, it was stated that only one event can occur in any time interval and only the present state of the system is important. Therefore, the transition rate matrix is a simple, albeit large, from-to matrix describing valid movement between system states. This matrix is derived by taking each of the $5N+2$ states, and determining to what other state the system may move to over a single time interval. Then, the rate at which this transition can occur is entered into the appropriate element of the matrix. Invalid transitions have a zero entry in their element of the transition matrix.

Therefore, the first step in deriving the transition rate matrix is to define all of the valid movements between states. This is accomplished through the use of a system state diagram. These diagrams consist of a number of boxes, each representing one state in the state space. Then, boxes representing valid state transitions are connected by a line. Finally, the lines are labelled with appropriate transition rates.

To develop the state diagram for the single server vacation model, it is necessary to include the following state transitions (Refer to Table 3.2-1 for parameter summary):

Note: Queue lengths are incremented upon job arrival and decremented upon service completion.

Server In The Target Queue

1) If, *Queue Length* $< N$; Then, a job arrival can occur. (rate λ)

2) If, *Queue Length* $= 1$;
Then, upon service completion phase 1 of normal vacation begins with prob. ϕ. (rate $\phi\mu_S$)
Or, phase 2 of normal vacation begins with prob. $(1-\phi)$. (rate $(1-\phi)\mu_S$)

3) If, $1 <$ *Queue Length* $\leq K$;
Then, upon service completion next job is served with prob. $(1-\Omega)$. (rate $\mu_S(1-\Omega)$)
Or, phase 1 of emergency vacation begins with prob. $\Omega\phi$. (rate $\mu_S\Omega\phi$)
Or, phase 2 of emergency vacation begins with prob. $\Omega(1-\phi)$. (rate $\mu_S\Omega(1-\phi)$)

4) If, *Queue Length* $> K$;
Then, the server remains in target queue upon next service completion. (rate μ_S)

Server In Phase 1 Of Normal Vacation

1) If, $0 \leq$ *Queue Length* $< (K-1)$; Then, a job arrival can occur. (rate λ)

2) If, $(K-1) \leq$ *Queue Length* $< N$;
Then, the server returns to the target queue upon next job arrival with prob θ. (rate $\lambda\theta$)
Or, the server remains on vacation with prob. $(1-\theta)$. (rate $\lambda(1-\theta)$)

3) If, *Queue Length* $= 0$;
Then, vacation phase termination results in transfer to phase 2. (rate $\mu_{V1}\Delta t$)

4) If, $1 \leq$ *Queue Length* $\leq N$;
Then, upon service completion vacation continues in phase 2 with prob. α. (rate $\alpha\mu_{V1}\Delta t$)
Or, the server returns to the target queue with prob. $(1-\alpha)$. (rate $(1-\alpha)\mu_{V1}\Delta t$)

Server In Phase 2 Of Normal Vacation

1) If, $0 \leq$ *Queue Length* $< (K\text{-}1)$; Then, a job arrival can occur. (rate λ)

2) If, $(K\text{-}1) \leq$ *Queue Length* $< N$;
Then, the server returns to the target queue upon next job arrival with prob θ. (rate $\lambda\theta$)
Or, the server remains on vacation with prob. $(1\text{-}\theta)$. (rate $\lambda(1\text{-}\theta)$)

3) If, *Queue Length* $= 0$;
Then, vacation phase termination results in transfer to phase 1. (rate $\mu_{V2}\Delta t$)

4) If, $1 \leq$ *Queue Length* $\leq N$;
Then, upon service completion vacation continues in phase 1 with prob. β. (rate $\beta\mu_{V2}\Delta t$)
Or, the server returns to the target queue with prob. $(1\text{-}\beta)$. (rate $(1\text{-}\beta)\mu_{V2}\Delta t$)

Server In Phase 1 Of Emergency Vacation

1) If, $1 \leq$ *Queue Length* $< (K\text{-}1)$; Then, a job arrival can occur. (rate λ)

2) If, $(K\text{-}1) \leq$ *Queue Length* $< N$;
Then, the server returns to the target queue upon next job arrival with prob θ. (rate $\lambda\theta$)
Or, the server remains on vacation with prob. $(1\text{-}\theta)$. (rate $\lambda(1\text{-}\theta)$)

3) If, $1 \leq$ *Queue Length* $\leq N$;
Then, upon service completion vacation continues in phase 2 with prob. α. (rate $\alpha\mu_{EI}\Delta t$)
Or, the server returns to the target queue with prob. $(1\text{-}\alpha)$. (rate $(1\text{-}\alpha)\mu_{EI}\Delta t$)

Server In Phase 2 Of Emergency Vacation

1) If, $1 \leq$ *Queue Length* $<$ *(K-1)*; Then, a job arrival can occur. (rate λ)

2) If, *(K-1)* $\leq$ *Queue Length* $<$ *N*;
Then, the server returns to the target queue upon next job arrival with prob θ. (rate $\lambda\theta$)
Or, the server remains on vacation with prob. *(1-θ)*. (rate $\lambda(1-\theta)$)

3) If, $1 \leq$ *Queue Length* $\leq$ *N*;
Then, upon service completion vacation continues in phase 1 with prob. β. (rate $\beta\mu_{E2}\Delta t$)
Or, the server returns to the target queue with prob. *(1-β)*. (rate $(1-\beta)\mu_{E2}\Delta t$)

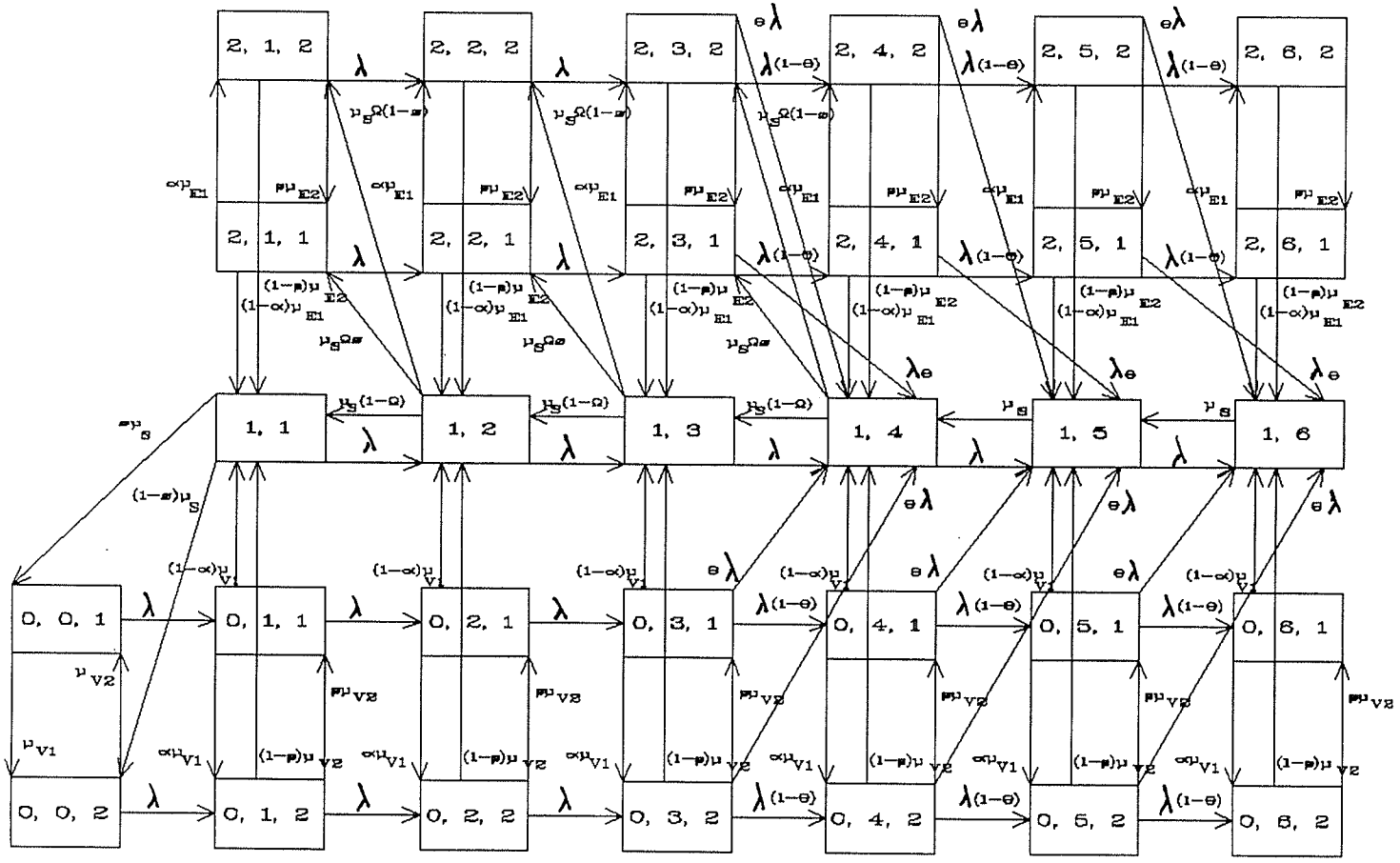
These feasible state transitions, along with their rates, are listed in Appendix A. Given this information, it is possible to draw a state space diagram of the single server vacation model. Such a diagram for the specific case where $K=4$ and $N=6$ is provided in Figure 4.3-1.

Given a diagram of this nature, it is a straightforward matter to develop the balance equations referred to in Section 4.1. In the steady state, the flow of jobs into each state must equal the flow out of the same state. Thus, every box in the state diagram will have a corresponding equation. The equations are found by recognizing that the sum of the lines entering the box must equal the sum of the lines out of the box.

For example, consider state *(1,5)* in Figure 4.3-1. For clarity, this and the relevant states in the system diagram are redrawn in Figure 4.3-2. The derivation of the balance equation for this state is as follows:

Figure 4.3–1: System State Diagram

K=4 N=6



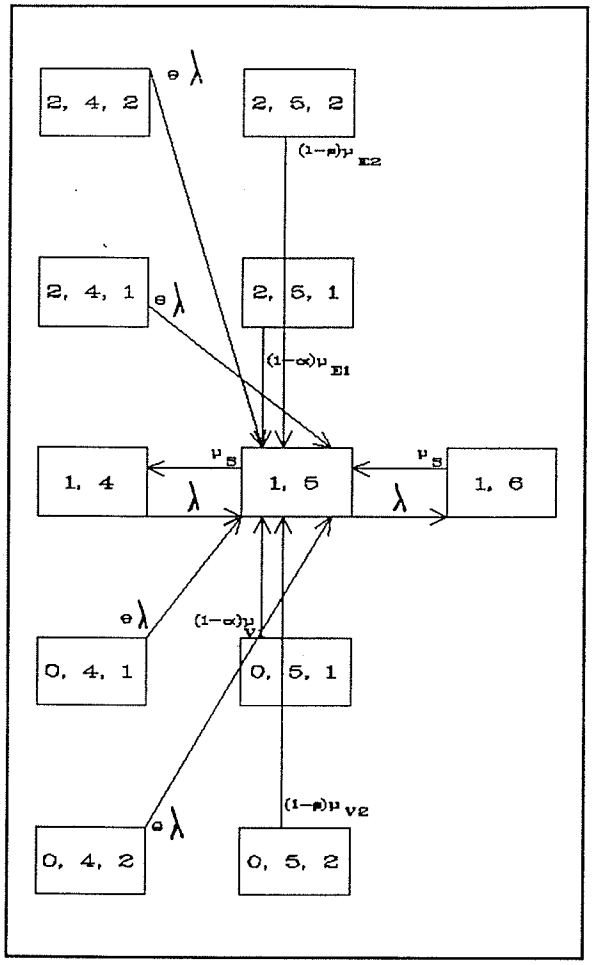


Figure 4.3-2: State (1,5)

Flow Into State

$\lambda P_{1,4} + \mu_S P_{1,6} + \theta\lambda(P_{0,4,1} + P_{0,4,2} + P_{2,4,1} + P_{2,4,2}) + (1-\alpha)(\mu_{V1} P_{0,5,1} + \mu_{E1} P_{2,5,1}) + (1-\beta)(\mu_{V2} P_{0,5,2} + \mu_{E2} P_{2,5,2})$

Flow Out Of State

$\mu_S P_{1,5} + \lambda P_{1,5}$

Balance Equation

$P_{1,5} = 1/(\mu_S+\lambda)\{\lambda P_{1,4} + \mu_S P_{1,6} + \theta\lambda(P_{0,4,1} + P_{0,4,2} + P_{2,4,1} + P_{2,4,2}) + (1-\alpha)(\mu_{V1} P_{0,5,1} + \mu_{E1} P_{2,5,1}) + (1-\beta)(\mu_{V2} P_{0,5,2} + \mu_{E2} P_{2,5,2})\}$

The same method can be used to develop balance equations for the entire system. The set of equations for the general single server vacation model are as follows:

$$P_{1,1}=\frac{1}{\mu_S+\lambda}\,[\mu_S(1-\Omega)\,P_{1,2}+(1-\alpha)\,(\mu_{V1}P_{0,1,1}+(1-\delta_{K1})\,\mu_{E1}P_{2,1,1})+ (1-\beta)\,(\mu_{V2}P_{0,1,2}+(1-\delta_{K1})\,\mu_{E2}P_{2,1,2})]$$

$$P_{1,i}=\frac{1}{\mu_S+\lambda}\,[\lambda P_{1,i-1}+\mu_S(1-\Omega)\,P_{1,i+1}+(1-\alpha)\,(\mu_{V1}P_{0,i,1}+(1-\delta_{K1})\,\mu_{E1}P_{2,i,1})+ (1-\beta)\,(\mu_{V2}P_{0,i,2}+(1-\delta_{K1})\,\mu_{E2}P_{2,i,2})]\,(1-\delta_{K2}) \qquad for\ 2\le i<K$$

$$P_{1,i}=\frac{1}{\mu_S+\lambda}\,[\lambda P_{1,i-1}+\mu_S P_{1,i+1}+\theta\lambda\,(P_{0,i-1,1}+P_{0,i-1,2}+ (1-\delta_{K1})\,(P_{2,i-1,1}+P_{2,i-1,2}))+(1-\alpha)\,(\mu_{V1}P_{0,i,1}+(1-\delta_{K1})\,\mu_{E1}P_{2,i,1})+ (1-\beta)\,(\mu_{V2}P_{0,i,2}+(1-\delta_{K1})\,\mu_{E2}P_{2,i,2})] \qquad for\ K\le i<N$$

$$P_{1,N}=\frac{1}{\mu_S}\,[\lambda P_{1,N-1}+\theta\lambda\,(P_{0,N-1,1}+P_{0,N-1,2}+(1-\delta_{K1})\,(P_{2,N-1,1}+P_{2,N-1,2}))+(1-\alpha)\,(\mu_{V1}P_{0,N,1}+(1-\delta_{K1})\,\mu_{E1}P_{2,N,1})+ (1-\beta)\,(\mu_{V2}P_{0,N,2}+(1-\delta_{K1})\,\mu_{E2}P_{2,N,2})]$$

$$P_{0,0,1}=\frac{1}{\mu_{V1}+\lambda}\,[\phi\mu_S P_{1,1}+\mu_{V2}P_{0,0,2}]$$

$$P_{0,i,1}=\frac{1}{\mu_{V1}+\lambda}\,[\lambda P_{0,i-1,1}+\beta\mu_{V2}P_{0,i,2}]\,(1-\delta_{K1}) \qquad for\ 1\le i<K$$

$$P_{0,i,1}=\frac{1}{\mu_{V1}+\lambda}\left[(1-\theta)\lambda P_{0,i-1,1}+\beta\mu_{V2}P_{0,i,2}\right](1-\delta_{KN}) \qquad for\ K\le i<N$$

$$P_{0,N,1}=\frac{1}{\mu_{V1}}\left[(1-\theta)\lambda P_{0,N-1,1}+\beta\mu_{V2}P_{0,N,2}\right]$$

$$P_{0,0,2}=\frac{1}{\mu_{V2}+\lambda}\left[(1-\phi)\mu_{S}P_{1,1}+\mu_{V1}P_{0,0,1}\right]$$

$$P_{0,i,2}=\frac{1}{\mu_{V2}+\lambda}\left[\lambda P_{0,i-1,2}+\alpha\mu_{V1}P_{0,i,1}\right](1-\delta_{K1}) \qquad for\ 1\le i<K$$

$$P_{0,i,2}=\frac{1}{\mu_{V2}+\lambda}\left[(1-\theta)\lambda P_{0,i-1,2}+\alpha\mu_{V1}P_{0,i,1}\right](1-\delta_{KN}) \qquad for\ K\le i<N$$

$$P_{0,N,2}=\frac{1}{\mu_{V2}}\left[(1-\theta)\lambda P_{0,N-1,2}+\alpha\mu_{V1}P_{0,N,1}\right]$$

$$P_{2,1,1}=\frac{1}{\mu_{E1}+\lambda}\left[\omega\phi\mu_{S}P_{1,2}+\beta\mu_{E2}P_{2,1,2}\right](1-\delta_{K1})$$

$$P_{2,i,1}=\frac{1}{\mu_{E1}+\lambda}\left[\omega\phi\mu_{S}P_{1,i+1}+\lambda P_{2,i-1,1}+\beta\mu_{E2}P_{2,i,2}\right](1-\delta_{K1})(1-\delta_{K2})$$
$$for\ 2\le i<K$$

$$P_{2,i,1}=\frac{1}{\mu_{E1}+\lambda}\left[(1-\theta)\lambda P_{2,i-1,1}+\beta\mu_{E2}P_{2,i,2}\right](1-\delta_{K1})(1-\delta_{KN}) \qquad for\ K\le i<N$$

$$P_{2,N,1}=\frac{1}{\mu_{E1}}\left[(1-\theta)\lambda P_{2,N-1,1}+\beta\mu_{E2}P_{2,N,2}\right](1-\delta_{K1})$$

$$P_{2,1,2}=\frac{1}{\mu_{E2}+\lambda}\left[\omega\phi\mu_{S}P_{1,2}+\alpha\mu_{E1}P_{2,1,1}\right](1-\delta_{K1})$$

$$P_{2,i,2}=\frac{1}{\mu_{E2}+\lambda}\left[\omega\phi\mu_{S}P_{1,i+1}+\lambda P_{2,i-1,2}+\alpha\mu_{E1}P_{2,i,1}\right](1-\delta_{K1})(1-\delta_{K2}) \quad for\ 2\le i<K$$

$$P_{2,i,2}=\frac{1}{\mu_{E2}+\lambda}\left[(1-\theta)\lambda P_{2,i-1,2}+\alpha\mu_{E1}P_{2,i,1}\right](1-\delta_{K1})(1-\delta_{KN}) \quad for\ K\le i<N$$

$$P_{2,N,2}=\frac{1}{\mu_{E2}}\left[(1-\theta)\lambda P_{2,N-1,2}+\alpha\mu_{E1}P_{2,N,1}\right](1-\delta_{K1})$$

where,

$$\delta_{i,j}=\begin{cases}1 & if\ i=j\\ 0 & otherwise\end{cases}$$

The general state equations presented above are valid for all cases where $K \le N$ and $N < \infty$. Application of these equations will result in a set of $5N+2$ linear equations (or $3N+2$ if $K=1$) that are solved simultaneously to determine the system steady state probabilities.

When these balance equations are put in the transition rate matrix form, a square, stable matrix of dimension $5N+2$ results. The structure of the transition rate matrix where $N=4$ and $K=2$ is provided in Figure 4.3-3. Note that in this figure an X in a matrix element indicates a non-zero value (ie. A valid transition). Also, each diagonal element is equal to the negative sum of all other elements in its row. The figure is divided into sections corresponding to the 5 rows of the system state diagram (See Figure 4.3-1). The number along each matrix axis is the number of customers in the target queue (ie. The j value in our state definition). The i and k value over each section also corresponds to the state definition of Section 4.2.

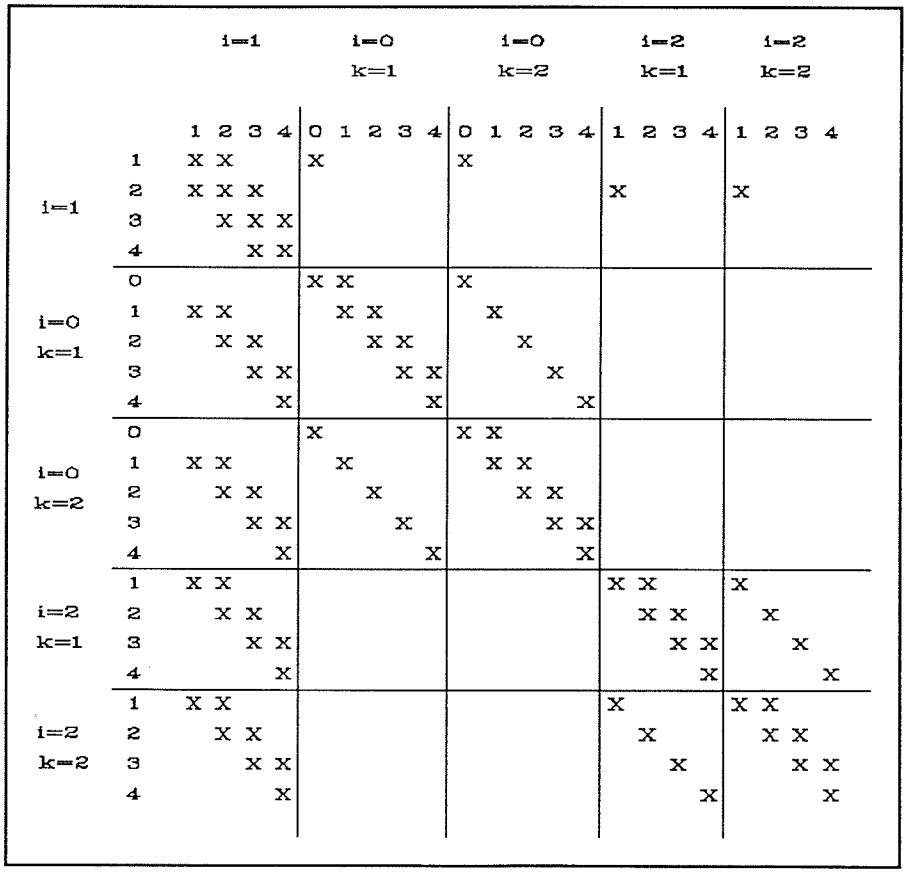


Figure 4.3-3: Transition Rate Matrix (N=4, K=2)

4.4 Solving for the Steady State

Once the set of linear equations describing the single server vacation model have been derived, it is possible to solve for the steady state. The solution method must be able to handle a large number of simultaneous equations with minimal error. The chosen technique for this thesis is an iterative approximation proposed by *Grassmann, et al [20]*.

Briefly, this method uses regenerative theory to modify the standard Gauss-Jordan solution procedure. An elimination algorithm is defined that begins in the last balance equation and works forward. When the first equation has been reached, all state probabilities will have been solved for. The algorithm itself is numerically very stable because it does not contain any subtraction, and if negative numbers are avoided the algorithm is very resistant to rounding errors.

Since the balance equations as defined in section 4.3 are valid for continuous time, they must be discretized prior to using the algorithm to eliminate negativity. This discretization is performed similar to the method described by *Altiok* and *Stidham [21]*. Simply put, the algorithm converts the transition rate matrix into a transition probability matrix. This is accomplished by setting all $a_{i,j}$ in the rate matrix to $a_{i,j}/\eta$ where η is a suitably chosen positive constant.

Once this is completed, one last observation needs to be made. Note that the set of equations is homogenous as they are all equal to zero. Also realize that the sum of all state probabilities must equal to one. Therefore, any one balance equation must be

replaced by:

$$\sum_{\forall i,j,k} P_{i,j,k}=1$$

The entire procedure outlined in this chapter was performed by a **FORTRAN** program. This coding of this program was divided into the following subroutines (A complete program listing is provided in Appendix B):

1) Initialize all model variables.

2) Define all elements of the transition rate matrix.

3) Discretize the transition rate matrix (ie. Convert it into a transition probability matrix).

4) Solve the set of simultaneous equations through *Grassmann's* algorithm.

5) Calculate and output the relevant performance measures.

CHAPTER 5
DETERMINATION OF VACATION PARAMETERS

5.1 Method of Determination

As stated previously in Section 3.2, the length of time a server is on vacation, along with both the server call back and call away probabilities, depend on the polling system under investigation. Thus, a qualitative understanding of these parameters is required before continuing on with the analysis of the single server vacation model. This understanding was gained through the use of a computer simulation, **PCModel**, and is the subject of this chapter. The simulation program listing is provided in Appendix B.

Using this package, a program was written to simulate the behaviour of a symmetric, finite capacity three queue polling system over time. A buffer size of nine for each queue was chosen as this value is large enough to give representative data while remaining tractable. Vacation parameters were derived from the program in the following way:

- The time that the server leaves the target queue, Q1, is noted and serves as the beginning of a vacation interval.
- If the vacation begins because Q1 becomes empty, the vacation is marked as a normal vacation. If it results from another queue exceeding K, it is marked as an emergency vacation.
- When the server next returns to Q1, the time is noted as a vacation termination.
- If this return occurs because both Q2 & Q3 have been served exhaustively,

the vacation is said to terminate regularly. Otherwise, the return results from Q1 exceeding K and the vacation is said to have terminated as a result of a call back.

- The length of the vacation is divided by the current service time to standardize the length in equivalent number of services.

- This value is stored according to type of vacation and mode of termination for later analysis.

The server call probabilities are also derived in this simulation. They are found in the following manner:

- Every time that the server is in Q1 and the length of Q1 is $\leq K$, a server call away *possibility* is said to have occurred. If after the next service completion the server does leave for another queue, an *actual* server call away is said to have occurred.

- At the end of the simulation run, the call away probability, Ω, is found by:

$$\Omega = \frac{No.\ of\ Actual\ Call\ Aways}{No.\ of\ Possible\ Call\ Aways}$$

- Every time that the server is not in Q1 and the length of Q1 is $\geq K$, a server call back *possibility* is said to have occurred. If after the next service completion the server does return to Q1, an *actual* server call back is said to have occurred.

- At the end of the simulation run, the call back probability, θ, is found by:

$$\theta = \frac{No.\ of\ Actual\ Call\ Backs}{No.\ of\ Possible\ Call\ Backs}$$

5.2 Description of Runs

For each simulation run, information on the first *5,000* jobs served is neglected to allow the system to reach the steady state. Then, information is collected until a further *25,000* customers have been processed. This large size allows for more accuracy when assessing the effects of variables on system performance.

Of particular interest are the effects of traffic density, ρ, and control limit, K, on the vacation parameters. This information is obtained by the following simulation runs:

I) Let $K=2$

II) Let $\rho=0.1$

III) Perform Simulation

IV) Let $\rho=\rho+0.1$

V) Repeat 3-4 while $\rho \leq 2.0$

VI) Let $K=K+1$

VII) Repeat **II-VI** while $K \leq 9$

Each simulation run provides a frequency distribution chart for Normal Vacation (Regular and Call Back Termination) and Emergency Vacation durations (Regular and Call Back Termination). They also return probability values for both Ω and θ. By analysing the simulation output, it was apparent that the data could be categorized according to traffic density. Therefore, further analysis can be grouped into low, medium and high values of ρ ($\rho=\lambda/\mu_S$).

5.3 Low Traffic Densities

The analysis discussed in this section are valid for a symmetric n queue polling system with low traffic densities. From the simulation runs, it was determined that the vacation parameters could be grouped together when:

$$\sum_{i=1}^{n} \rho_i \leq 0.5$$

When operating in this state, the system is characterized by low queue lengths. As a result, the server often exhausts one buffer and moves on to the next. In terms of vacation length, these normal type vacations are of short duration. While on a normal vacation, this analysis showed terminations are due to a call back only for low to medium values of K. This is because queue lengths rarely exceed K as it is increased. Frequency distribution graphs for these situations are provided in Figures 5.3-1 and 5.3-2.

Emergency vacations begin with a server call away. For low values of K, call aways occasionally happen, but for higher K values these occurrences are rare and can be neglected. Similar to normal vacations, emergency vacations are of short duration when dealing with low traffic densities. Also, call backs only occur while K is low. Figures 5.3-3 and 5.3-4 illustrate emergency vacation length distributions.

Figure 5.3-1: Regular Termination of Normal Vacations - Low ρ

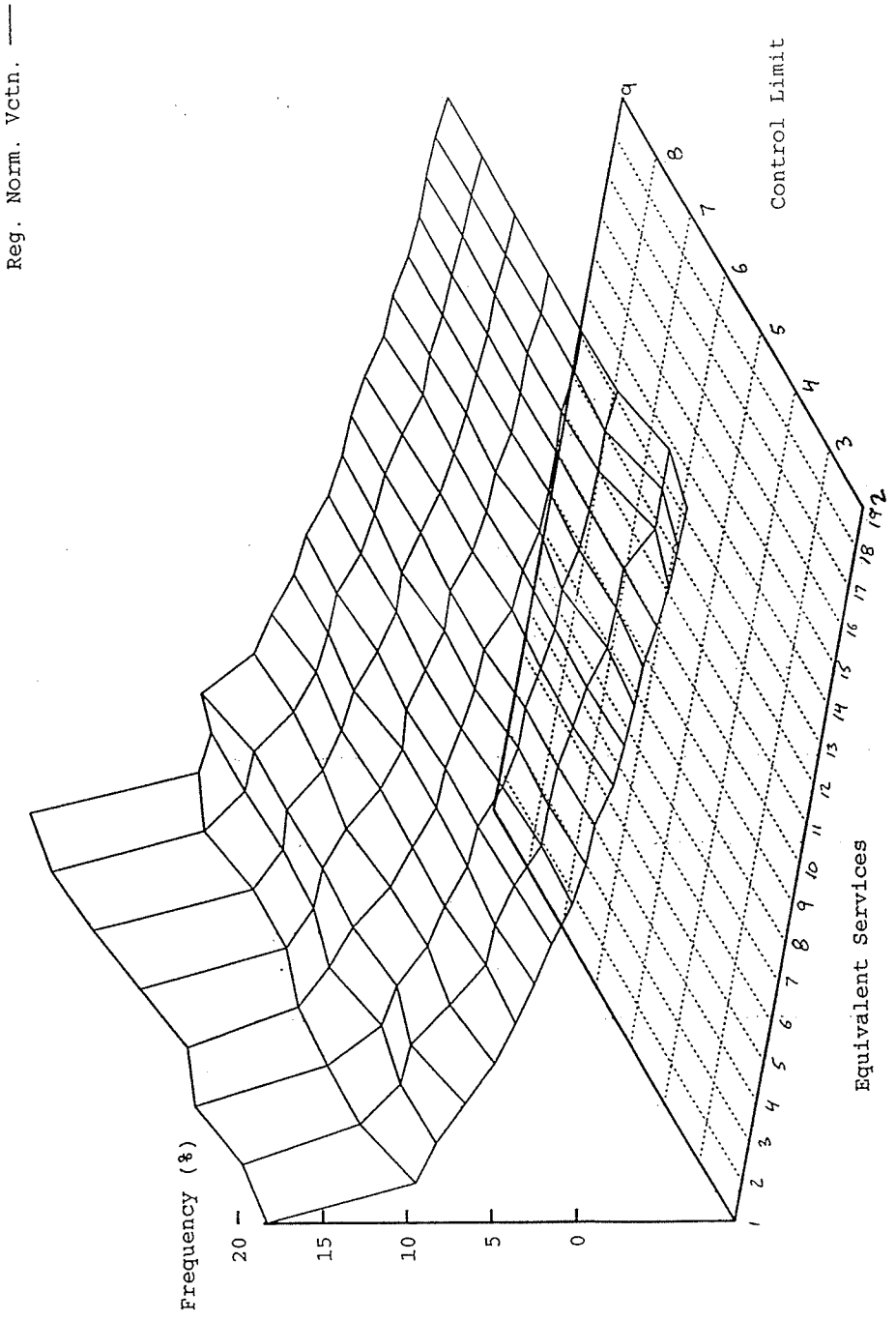


Figure 5.3-2: Call Back Termination of Normal Vacations - Low ρ

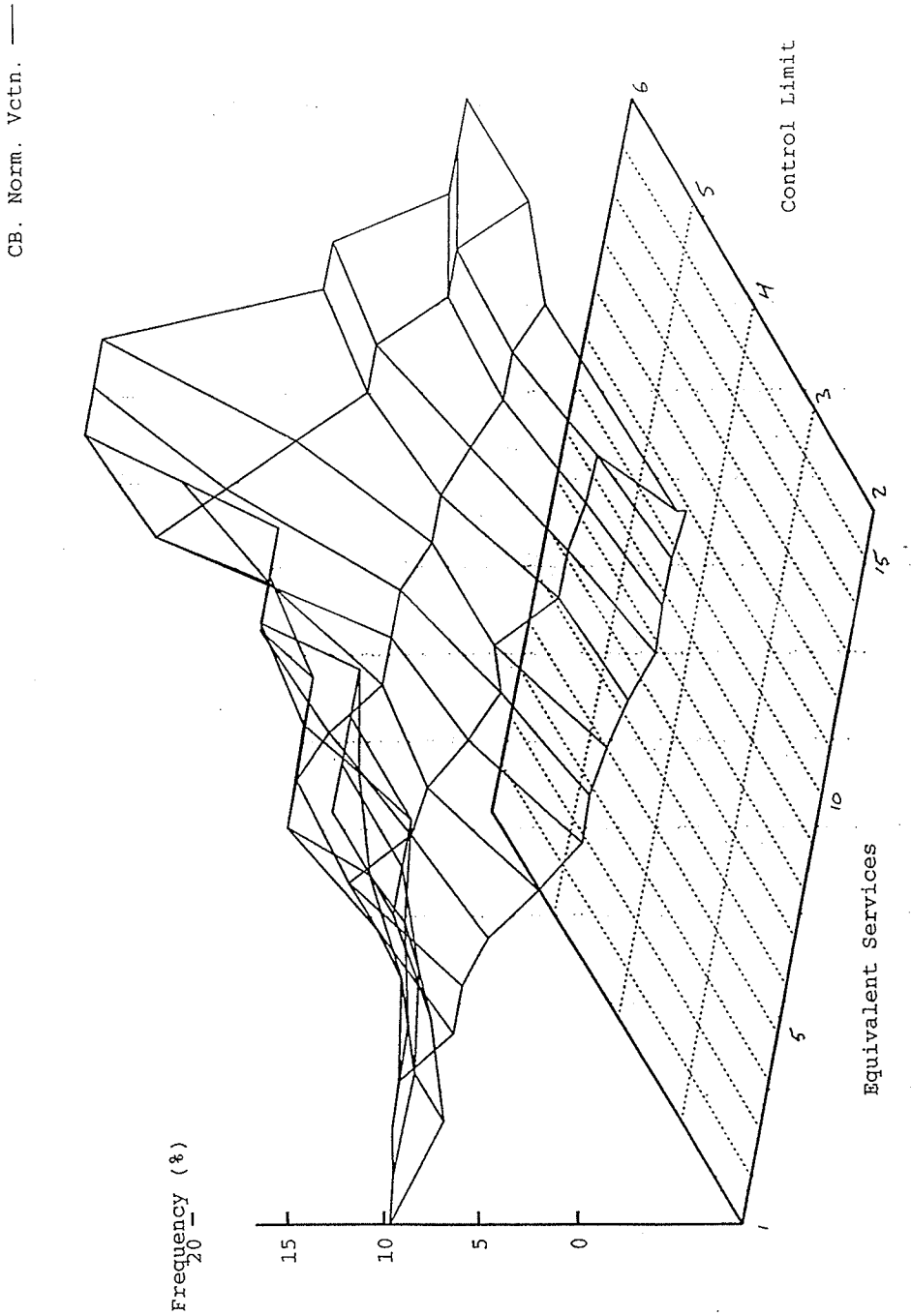


Figure 5.3-3: Regular Termination of Emergency Vacations - Low ρ

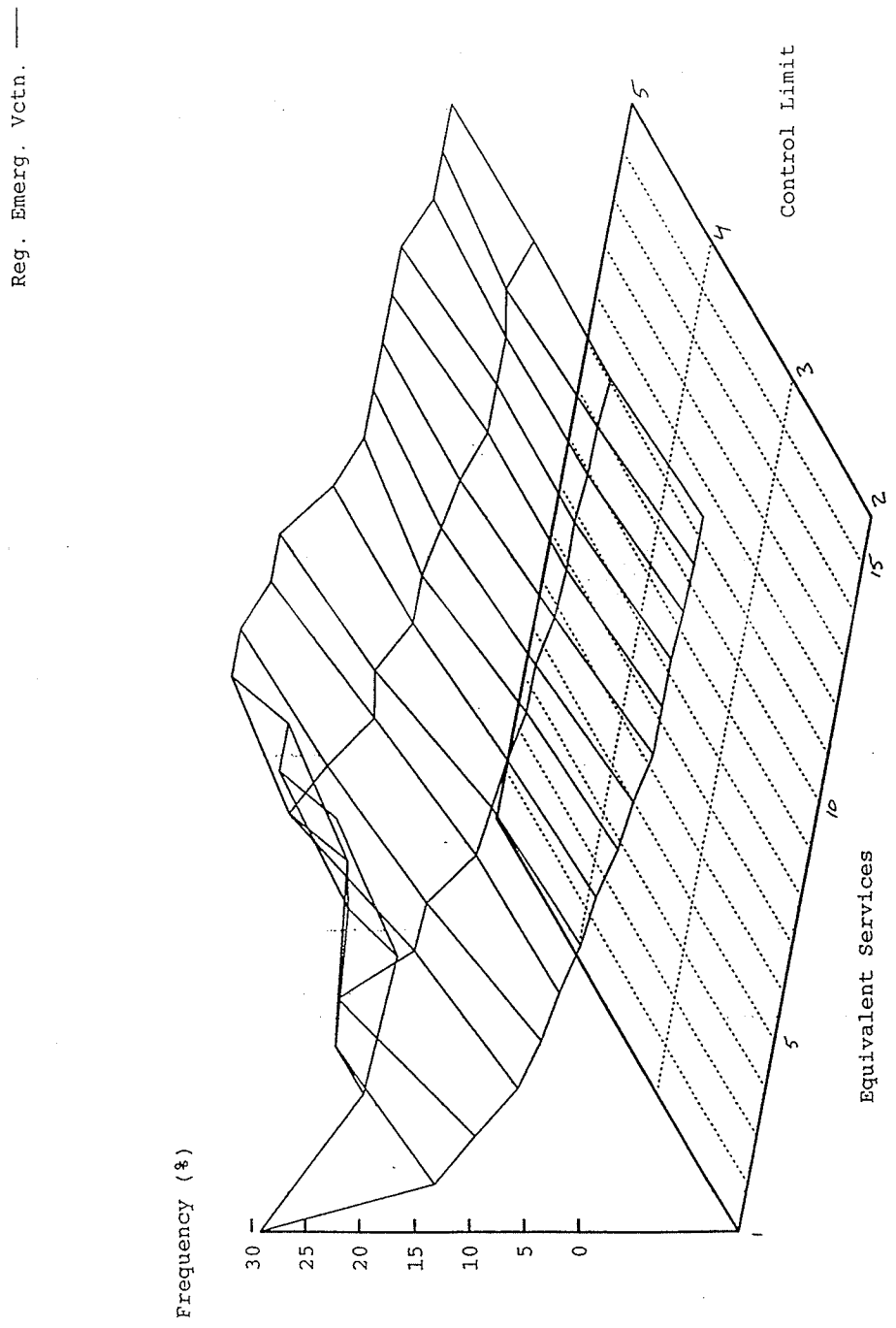
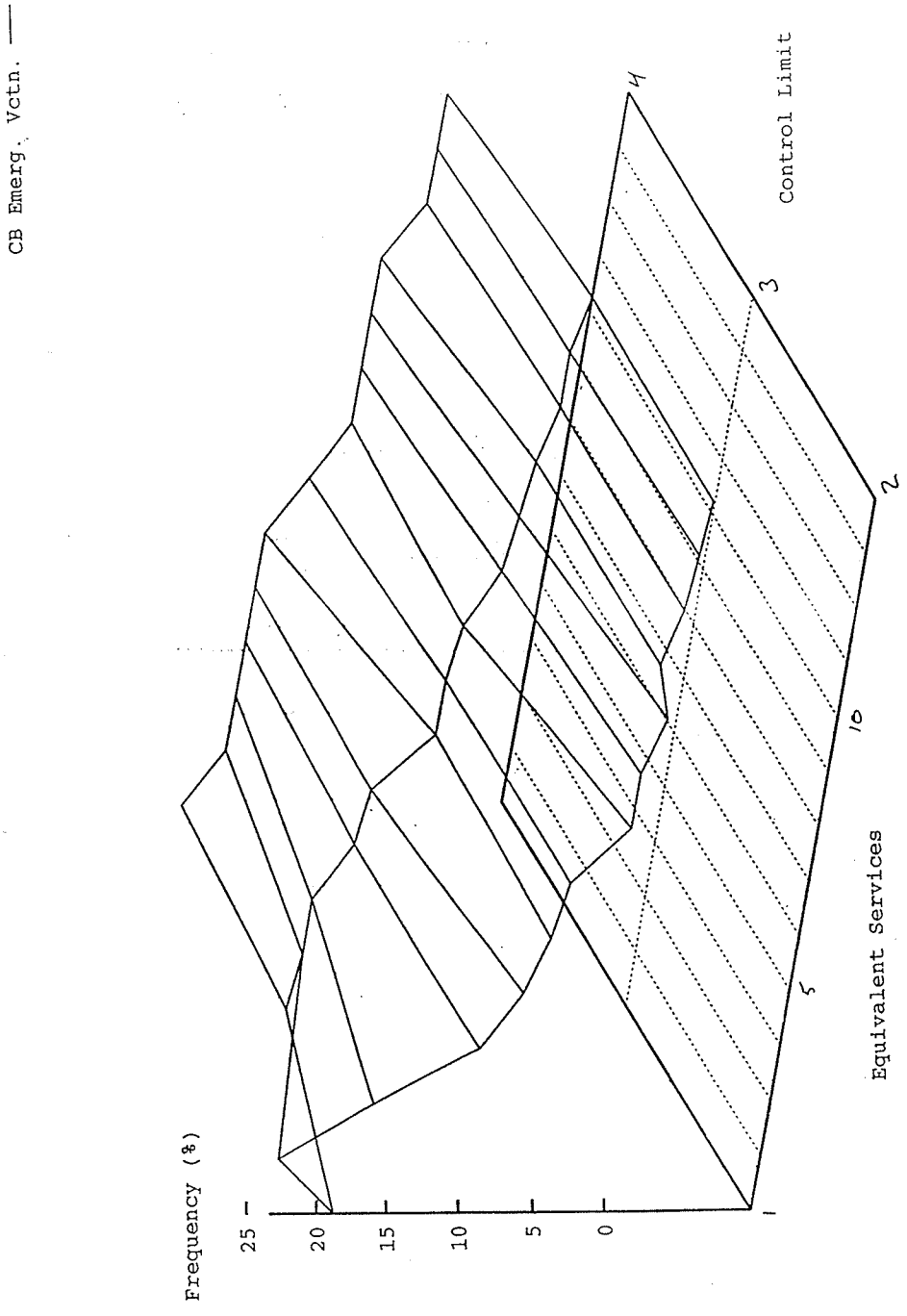


Figure 5.3-4: Call Back Termination of Emergency Vacations - Low ρ



Before analysing this overload control strategy as a single server vacation model, appropriate values for Ω and θ must be found. The simulation runs have shown that values near 0% and 90% were valid for call away and call back percentages, respectively. Appropriate values of Ω and θ for all control levels at low traffic densities are given in Figure 5.3-5. The value of Ω is low because these situations rarely arise. High values for θ occur since queue lengths are generally lower than K. However, when K is exceeded in one of the queues, the others are usually less than K. This results in either a call back or a call away depending on where the server is at the time.

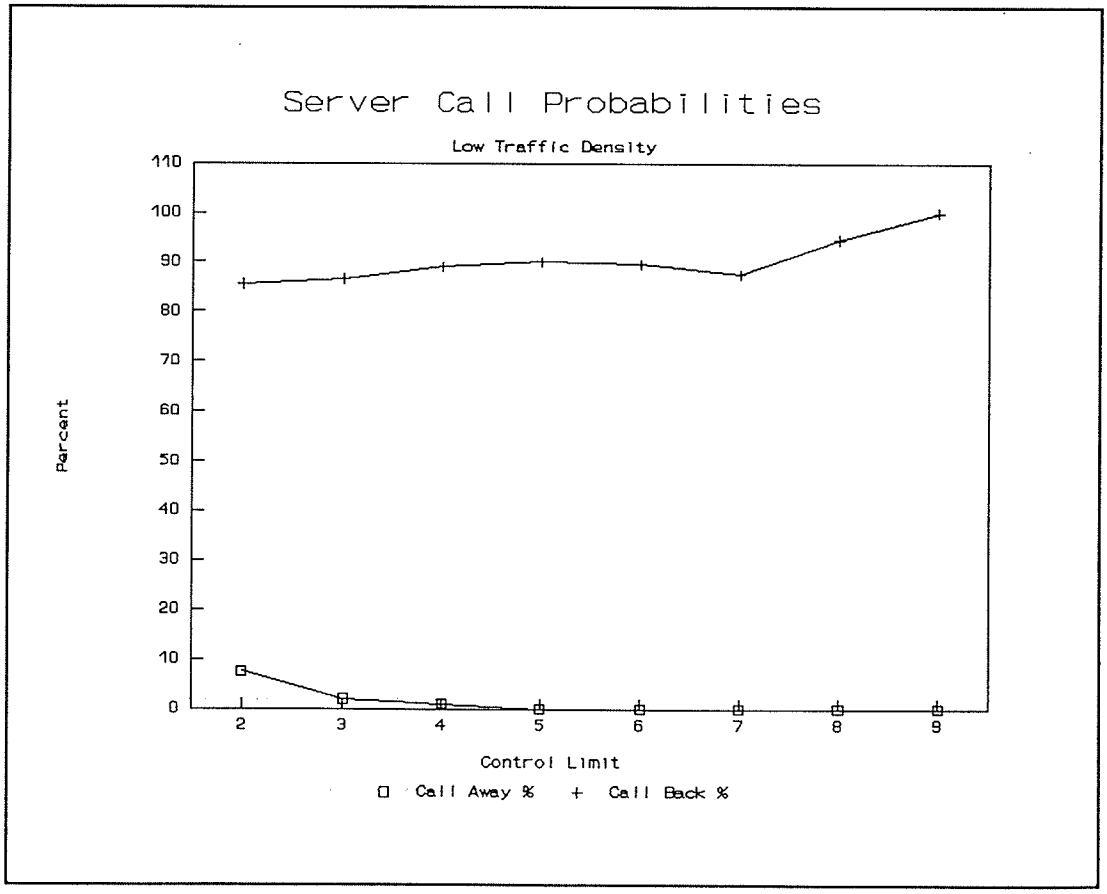


Figure 5.3-5: Server Call Probabilities - Low ρ

Thus, from the graphs, it is seen that when modelling this control strategy for low values of ρ it is appropriate to model both vacations with short average durations. They must be designed so that vacations are generally completed before the target queue reaches K. Call away percentage must be set near 0% and call back percentage slightly lower than 100%. The following information regarding normal vacations are provided in Table 5.3-1. The same information for emergency vacations are given in Table 5.3-2:

1) % of vacations that terminate in the regular manner.

2) The mean & standard deviation for the regular termination distribution.

3) % of vacations that terminate due to a call back.

4) The mean & standard deviation for the call back termination distribution.

Table 5.3-1: Normal Vacation Data For Low Traffic Densities

K	**Regular Termination**	μ	σ	**Call Back Termination**	μ	σ
2	92.6%	7.7	7.7	7.4%	8.1	7.4
3	97.1%	7.5	7.3	2.9%	9.0	6.4
4	99.0%	7.8	7.6	1.0%	10	5.5
5	99.6%	7.8	7.5	0.4%	11	5.3
6	99.8%	8.0	7.6	0.2%	17	7.8
7	100%	7.9	7.5			
8	100%	7.8	7.5			
9	100%	7.7	7.6			

Table 5.3-2: Emergency Vacation Data For Low Traffic Densities

K	**Regular Termination**	μ	σ	**Call Back Termination**	μ	σ
2	65.5%	6.4	7.4	34.4%	5.8	5.5
3	61.2%	6.8	6.7	38.8%	5.9	5.1
4	73.4%	7.5	4.5	26.6%	7.0	5.3
5	72.1%	9.4	5.2	27.9%	9.1	4.4
6	100%	11	4.7			

A summary of the simulation analysis for low traffic densities is provided in Table 5.3-3.

Table 5.3-3: Simulation Analysis Summary - Low Traffic Density

Normal vacations	Short duration, regardless of K
Emergency vacations	Short duration, regardless of K
Server call away probability	Very low, regardless of K
Server call back probability	Very high, regardless of K

5.4 Medium Traffic Densities

The analysis discussed in this section are valid for a symmetric n queue polling system with medium traffic densities. The simulation study found that for a symmetric n queue polling system, a second grouping can be formed when:

$$0.5<\sum_{i=1}^{n}\rho_i<1.0$$

This group interests communication network designers most. It is ideal from a design standpoint to have the server busy 100% of the time yet have a minimum number of job losses. The medium range of ρ resembles this situation most closely. Here, the queues were seen to vary from empty to full throughout the simulation runs.

In this range the server is seen to go on both vacation types regardless of the overload control limit. The value of K does, however, have an impact on the vacation durations as well as the portion of vacations that are terminated due to a call back. In terms of normal vacations, increasing K will result in an increase in the portion of regular vacation terminations along with a corresponding rise in vacation duration for both termination types.

This is explained by the fact that normal vacations begin with an empty target queue. By increasing K, more arrivals have to occur prior to a call back. Thus, increasing K allows the server more time to terminate the vacation regularly. Figures 5.4-1 and 5.4-2 show the normal vacation length distributions for both termination types.

Figure 5.4-1: Regular Termination of Normal Vacations - Med. ρ

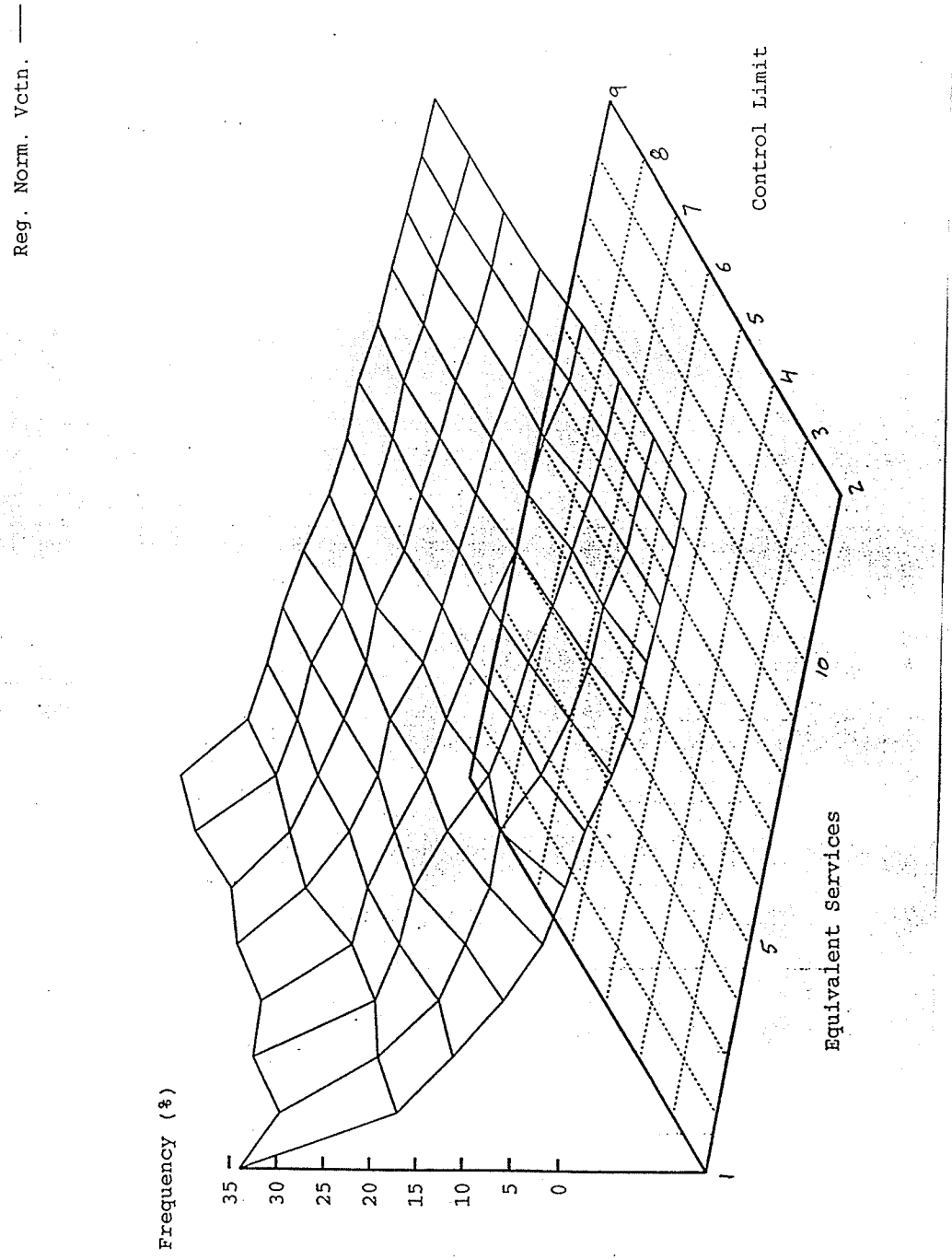
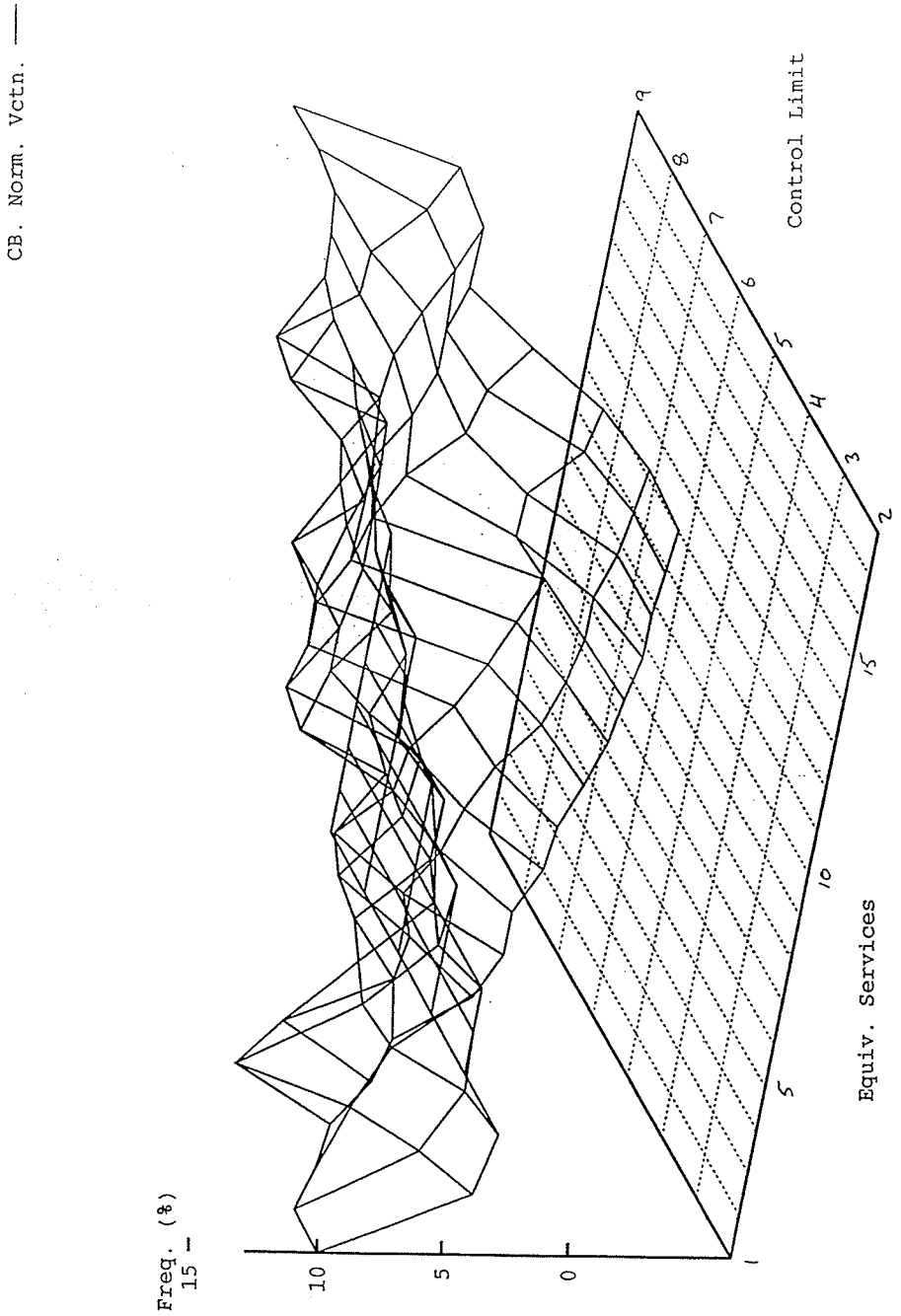


Figure 5.4-2: Call Back Termination of Normal Vacations - Med. ρ



With respect to emergency vacations, increasing K gives the server fewer opportunities to take an emergency vacation and more of a chance to be called back from vacation. This leads to a higher portion of emergency vacations terminating in a server call back.

A higher control limit also means that when an emergency vacation begins, the queue that the server is called to has a longer length. Thus, it takes more time to complete the vacation for higher K values. Similarly, increasing K leads to a decrease in the duration of vacations that terminate in a call back because the call back can occur sooner. Figures 5.4-3 and 5.4-4 graph the emergency vacation length distributions for both termination types.

For Ω, the probability is seen to begin around 70% and decrease almost linearly with increasing control limit. This is because as K rises, the chance that a call away will occur decreases. With respect to θ, a 25% value increases rapidly as $K \rightarrow N$, until it reaches 100% at $K = N$. The simulation results for medium densities are graphed in Figure 5.4-5.

Tables 5.4-1 and 5.4-2 provide the simulation results for both normal and emergency vacations that correspond to medium ρ values.

Figure 5.4-3: Regular Termination of Emergency Vacations - Med. ρ

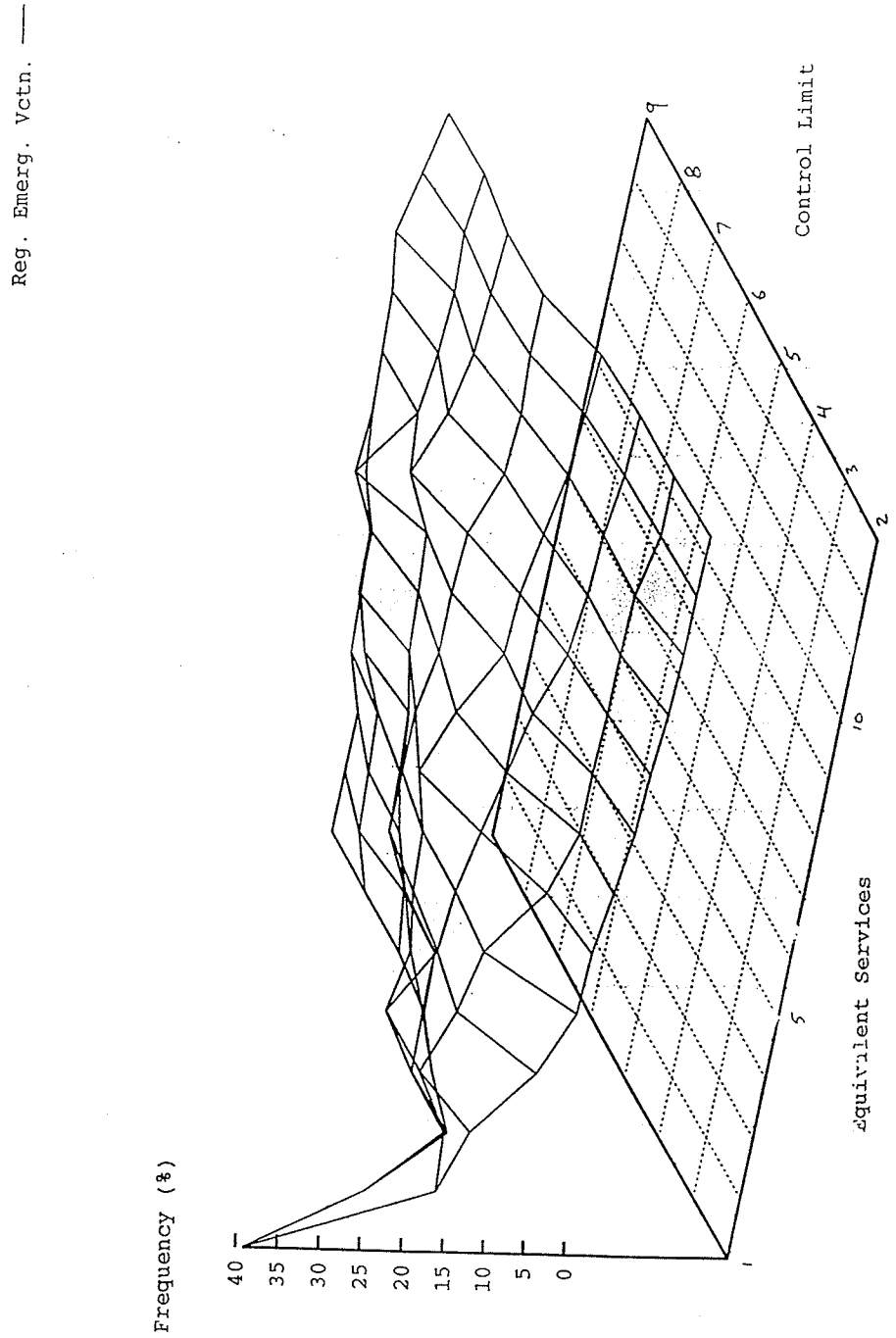
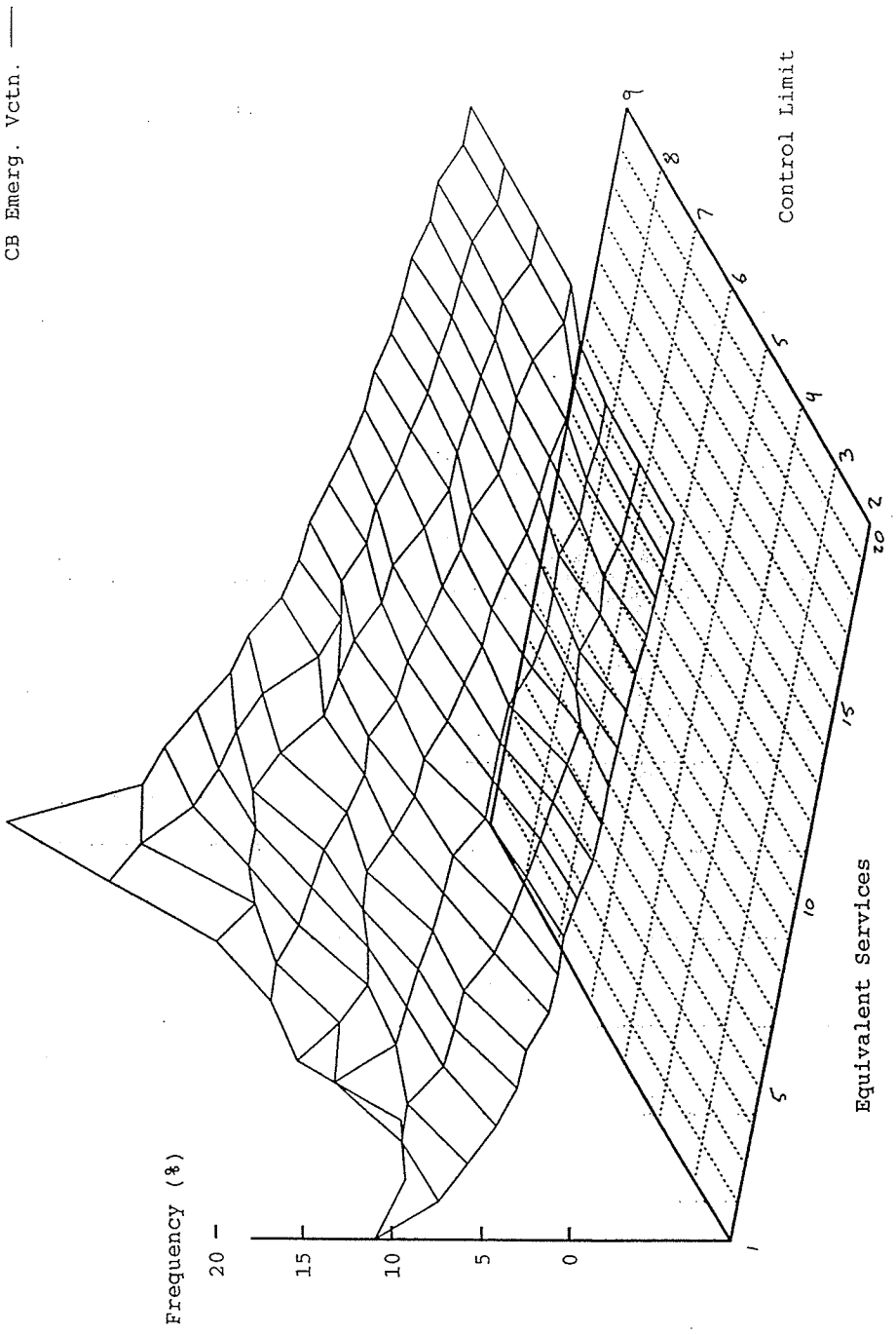


Figure 5.4-4: Call Back Termination of Emergency Vacations - Med. ρ



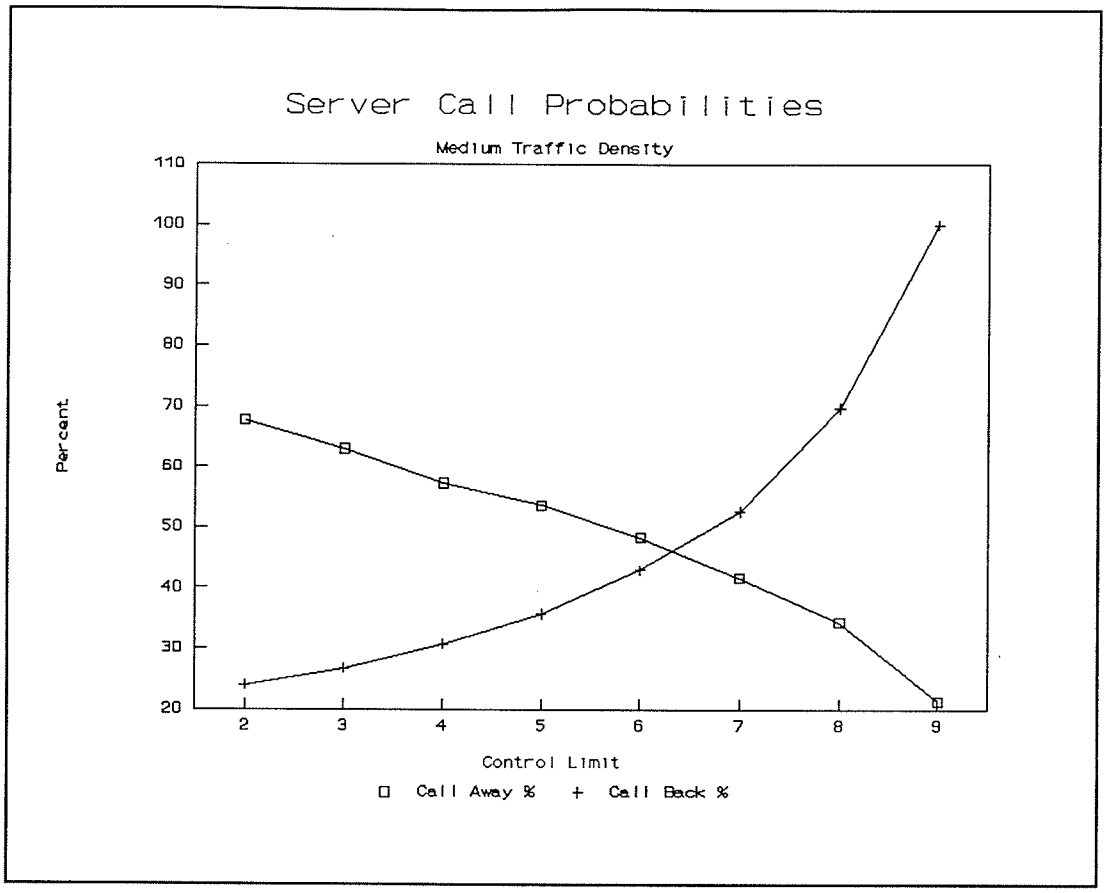


Figure 5.4-5: Server Call Probabilities - Med. ρ

Table 5.4-1: Normal Vacation Data For Medium Traffic Densities

K	Regular Termination	μ	σ	Call Back Termination	μ	σ
2	72.6%	2.9	2.8	27.4%	7.6	8.0
3	81.5%	3.5	3.0	18.5%	8.6	6.9
4	87.6%	4.1	3.4	12.4%	12	9.0
5	89.7%	5.0	4.4	10.3%	13	5.6
6	92.2%	5.5	4.7	7.8%	15	6.2
7	91.4%	6.1	5.0	8.6%	18	6.6
8	93.8%	7.6	6.0	6.2%	19	6.2
9	93.6%	8.4	6.3	6.4%	21	6.5

Table 5.4-2: Emergency Vacation Data For Medium Traffic Densities

K	Regular Termination	μ	σ	Call Back Termination	μ	σ
2	21.1%	2.8	2.8	78.9%	13	10
3	16.1%	4.1	3.1	83.9%	12	8.6
4	12.4%	5.7	3.7	87.6%	9.9	7.6
5	13.0%	6.4	3.8	87.0%	8.4	6.6
6	10.8%	8.5	4.5	89.2%	7.9	6.0
7	9.0%	11	5.9	91.0%	7.2	5.4
8	7.2%	12	6.7	92.8%	7.1	5.9
9	5.0%	16	10	95.0%	9.0	7.0

A summary of the simulation analysis for medium traffic densities is provided in Table 5.4-3.

Table 5.4-3: Simulation Analysis Summary - Medium Traffic Density

Normal vacations	Increase duration as K increases
Emergency vacations	Decrease duration as K increases
Server call away probability	Decrease as K increases
Server call back probability	Increase as K increases

5.5 High Traffic Densities

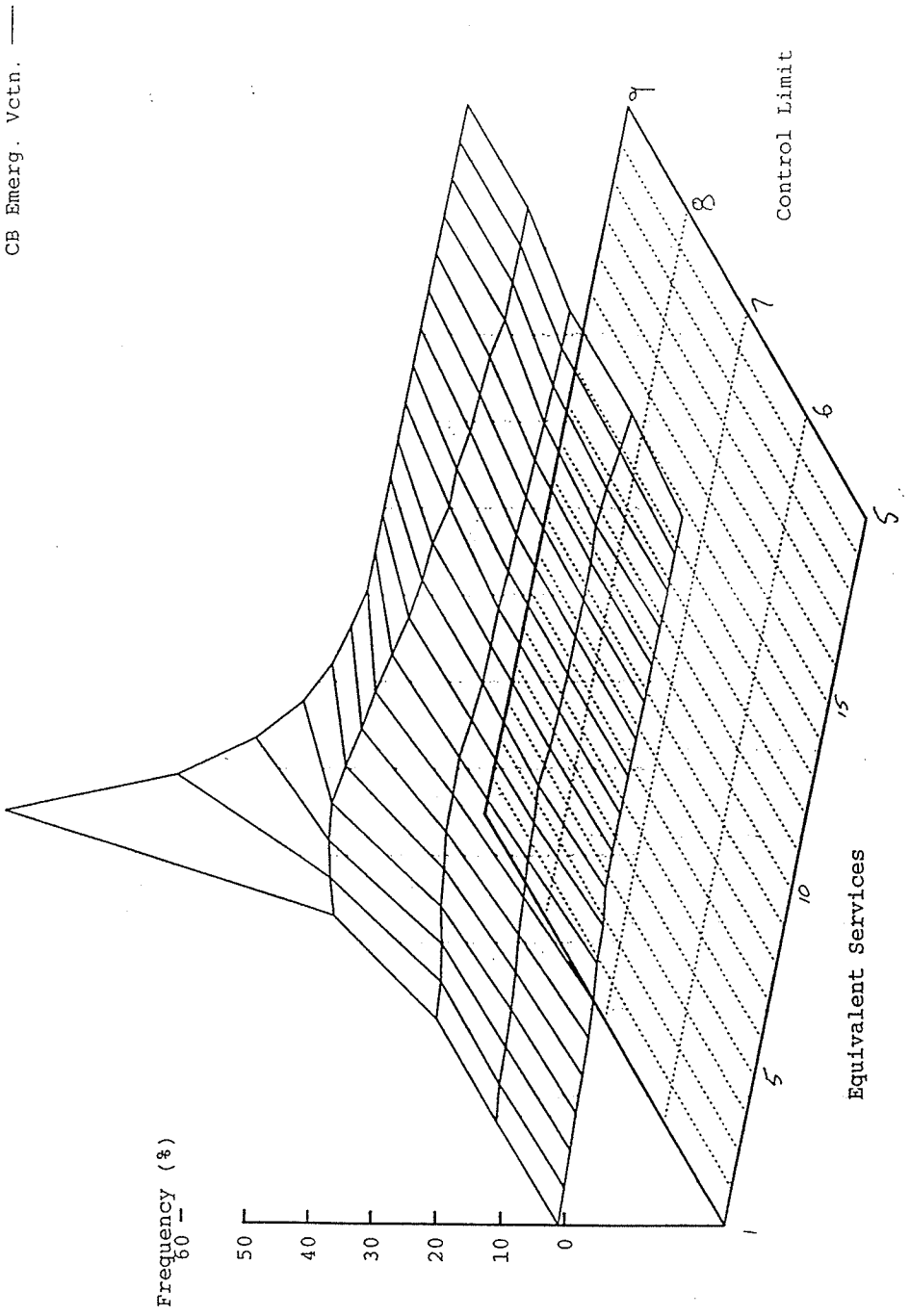
The analysis discussed in this section is valid for a symmetric n queue polling system with high traffic densities. The final grouping of vacation parameters is formed when:

$$\sum_{i=1}^{n} \rho_i \geq 1.0$$

This set is distinguished by the high queue lengths associated with it. In fact, the simulation runs have shown that all buffers are at capacity most of the time. As a result, servers are always called away before emptying a queue. Thus, normal vacations are never taken. In addition, once called away on an emergency vacation, the target queue quickly exceeds K once again. This results in a call back and the prevention of a regular vacation termination.

Therefore, when ρ is high the only vacations are of the emergency type that terminate by a server call back. For low values of K, these vacations are of an extremely long duration. This is because once a server begins a vacation, it will not be called back to the target queue until the last buffer in the cycle is served to length $< K$. As ρ increases, the length of time until this occurs also increases. Vacation duration decreases with increasing control limit until $K = N$. At this point, the vacation is usually comprised of N - 1 services. The frequency distribution graph for this grouping is provided in Figure 5.5-1.

Figure 5.5-1: Call Back Termination of Emergency Vacations - High ρ



Appropriate values for Ω and θ very near 100% for Ω and near 0% for θ when K is low and increasing to 100% for $K = N$. These probabilities are explained using the same logic used in explaining the vacation duration. The server call probabilities for the high traffic density case are graphed in Figure 5.5-2.

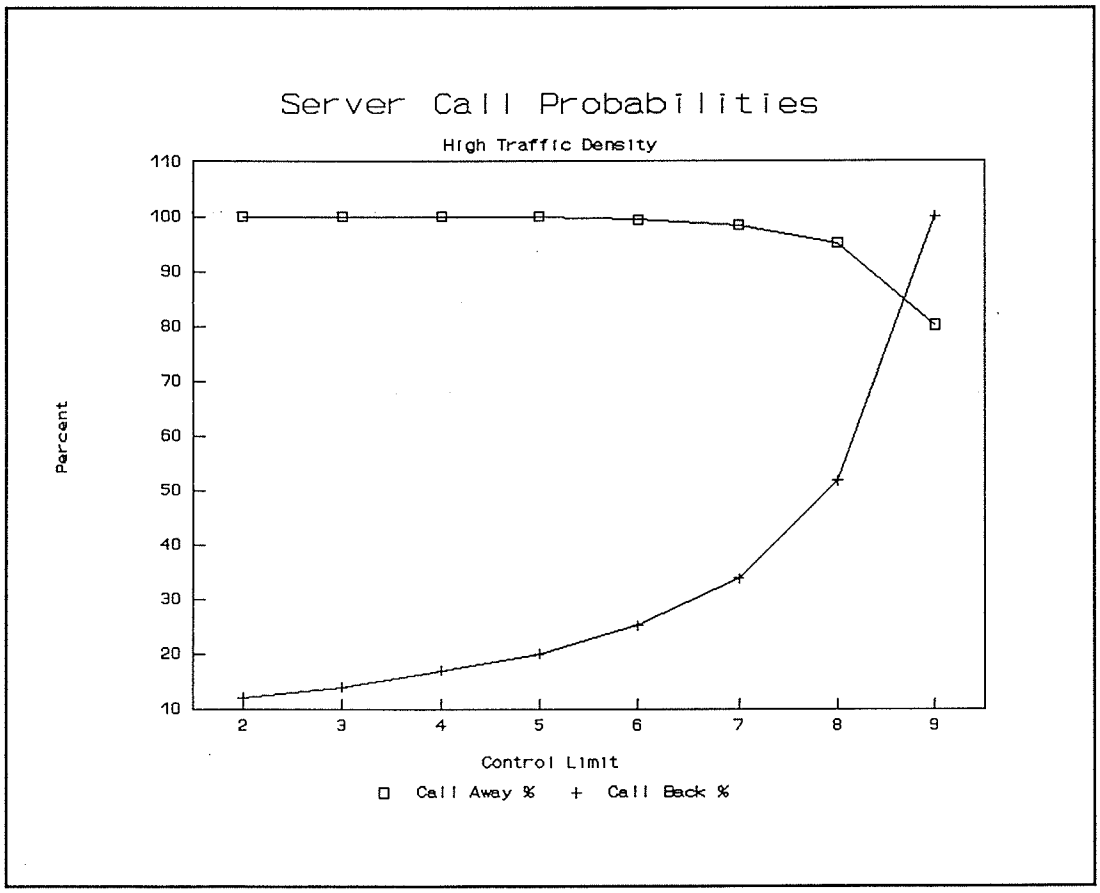


Figure 5.5-2: Server Call Probabilities - High ρ

Therefore, modelling this density group as a single server vacation model requires that normal vacations are never taken. Also, emergency vacations must be designed very long for low K and allowed to decrease as K increases. They also must be set so that they terminate as a result of a call back. The data resulting from this set of simulation experiments is given in Table 5.5-1.

Table 5.5-1: Emergency Vacation Data for High Traffic Densities

K	**Call Back Termination**	μ	σ
2	100%	239	137
3	100%	129	71
4	100%	74	45
5	100%	44.5	27.3
6	100%	24.7	15.7
7	100%	13.2	8.8
8	100%	6.0	4.4
9	100%	2.2	1.9

A summary of the simulation analysis for medium traffic densities is provided in Table 5.5-2.

Table 5.5-2: Simulation Analysis Summary - High Traffic Density

Normal vacations	Not taken
Emergency vacations	Very long with low K; Decrease rapidly as K increases
Server call away probability	Very high, regardless of K
Server call back probability	Very low with low K; Increase slowly for mid K values; Reaches 100% when $K=N$

CHAPTER 6
ANALYSIS OF THE SINGLE SERVER VACATION MODEL

6.1 General Description

This chapter discusses the results of the single server vacation analysis. It begins with a justification of using the simulation results to estimate vacation parameters for the single server model. This is followed by an explanation of the experiments to be run and the system performance measures to be analyzed. Then, these individual measures will be analyzed to determine the effectiveness of this overload control strategy.

Having obtained information on the vacation parameters through the simulation runs, it is possible to perform an evaluation of this overload control strategy. It must be noted that the actual simulation output is not what is critical. This is because the values are valid for the particular operating conditions described in the previous chapter. What are important are the parameter change trends that were uncovered through the simulation exercise.

It is reasonable to assume that these trends (eg. Increase emergency vacation rate with increasing control limit for high traffic density.) are valid regardless of the number of queues in the polling system and their respective buffer sizes. The only thing these factors will affect are the magnitude of these change trends. Therefore, the analysis of the single server vacation model incorporates these trends in a consistent manner yielding consistent results.

6.2 Experimental Design

As explained in the model description, vacations were to be defined by a phase type distribution. This definition agrees with the results from the simulation. These results do, however, allow for a further simplification of the analysis. Given the previously described vacation behaviour, it is possible to use an Erlang-2 distribution for both vacation types. This distribution was initially created to aid in the modelling of telephone traffic as it provides flexibility with easier implementation. Being two phase, μ_{V1}, μ_{V2}, μ_{E1} and μ_{E2} still exist but vacations last through both phases only once. As such, the following parameter values remain constant for all program runs:

- $\alpha=1$
- $\beta=0$
- $\phi=1$

Having determined appropriate vacation parameters (ie. μ_{V1}, μ_{V2}, μ_{E1}, μ_{E2}, Ω, and θ) with the aid of simulation, a final parameter selection must be made prior to performing the experimental runs. This decision is the choice of an appropriate buffer size. It involves a trade-off between realism and program complexity. That is, the larger the buffer size, the more realistic the model (to a point) for certain situations. Unfortunately, a conflict exists in that it also causes the state space of the model to increase, resulting in longer program running times and potential exposure to round off errors.

Another advantage to a lower setting of the queue length is that it will provide stronger emphasis on the effect of the overload control strategy. Regardless of queue size, the overall behaviour of the system with control limit implementation remains the same. By tightening the range from an empty to a full buffer (ie. shortening the capacity of the buffer), the system is forced to react more strongly with smaller changes in the level of the control limit. Keeping these issues in mind, a number of trial program runs were completed under a variety of initial conditions. These experiments indicated that a buffer size of thirty was a suitable compromise between all factors.

As the primary motivation for this thesis is investigating the effectiveness of this control scheme, the experiments were designed to provide this information. In order to see how various control levels react in different operating conditions, the analysis of the single server vacation model includes three density groups: low ($\rho=0.1$), medium ($\rho=0.5$) and high ($\rho=1.0$). This agrees with the results of the simulation analysis that experiments can be classified according to traffic density.

Therefore, for each of the density levels the value of the control limit, K, was varied from a low of two to a high of thirty (recall that $N=30$). An additional run was performed for each density level that modelled the situation of a zero control level (ie. as though the system had no overload control mechanism in place). This effect is achieved by setting the following performance measures:

- $K = N/A$
- $\theta = 0\%$
- $\Omega = 0\%$

Each execution of the program produces output for a number of performance measures. These measures provide information on vacations taken (ie. How often they occur and their behaviour), system occupancy (ie. Average queue length, idle probability) and loss frequency (ie. Probability that the queue is full). For a listing of the parameter initial conditions and corresponding program output, please refer to Appendix D. These experiments provide the basis for rating the effectiveness of the control scheme. The remainder of this chapter is devoted to the analysis of these measures.

6.3 Vacation Behaviour

In designing the polling system, it is of interest to know how often the server is on each type of vacation, and the length of the target queue while vacations are occurring. For example, if the server is found to be on vacation or the queue length is larger than K an excessive amount of time, this may indicate that the control limit needs to be redefined, especially if the target queue contains important jobs. To gain this knowledge, each program execution provides the following information:

- **NVAC:** The portion of time that the server is taking a normal vacation. It is defined by:

$$\sum_{k=1}^{2}\sum_{j=0}^{N} P_{0,j,k}$$

- **EVAC:** The portion of time that the server is taking an emergency vacation. It is defined by:

$$\sum_{k=1}^{2}\sum_{j=1}^{N} P_{2,j,k}$$

- **VLTK:** While the server is on vacation, the percentage of time that the length of the target queue is less than the control limit. It is defined by:

$$\sum_{k=1}^{2}\sum_{j=0}^{K-1} P_{0,j,k} + \sum_{k=1}^{2}\sum_{j=1}^{K-1} P_{2,j,k} * (NVAC + EVAC)^{-1}$$

- **VGEK:** While the server is on vacation, the percentage of time that the length of the target queue is greater than or equal to the control limit. It is defined by:

$$\sum_{k=1}^{2}\sum_{j=K}^{N} P_{0,j,k} + \sum_{k=1}^{2}\sum_{j=K}^{N} P_{2,j,k} * (NVAC + EVAC)^{-1}$$

In running the experiments with low traffic densities, it is found that the value of the control limit has very little effect on the total amount of time that a server is on vacation. With the initial conditions as given in appendix A, it is seen that the server is on vacation a large portion of the time (approximately 90%). For lower levels of K, the server is seen to take emergency vacations once in a while. However, when $K>18$, these instances become nonexistent. This information is presented graphically in Figure 6.3-1.

Due to the low value of ρ, instances of high queue lengths will be limited. This is shown in the percentage of time that the target queue exceeds K while the server is on vacation (See Figure 6.3-2). Here the queue exceeds K only when $K<6$. Even in these experiments this occurrence is limited.

The effect of varying the control limit is the greatest when modelling medium values of ρ. For this set of experiments, implementing the control strategy lowers the total vacation percentage by about 4% (See Figure 6.3-3). The vacation level remains fairly constant until $K=N$. At this point, the server is seen to be on vacation a greater percentage of the time compared to the no control situation. This test shows that normal

Figure 6.3-1

Vacation Probabilities

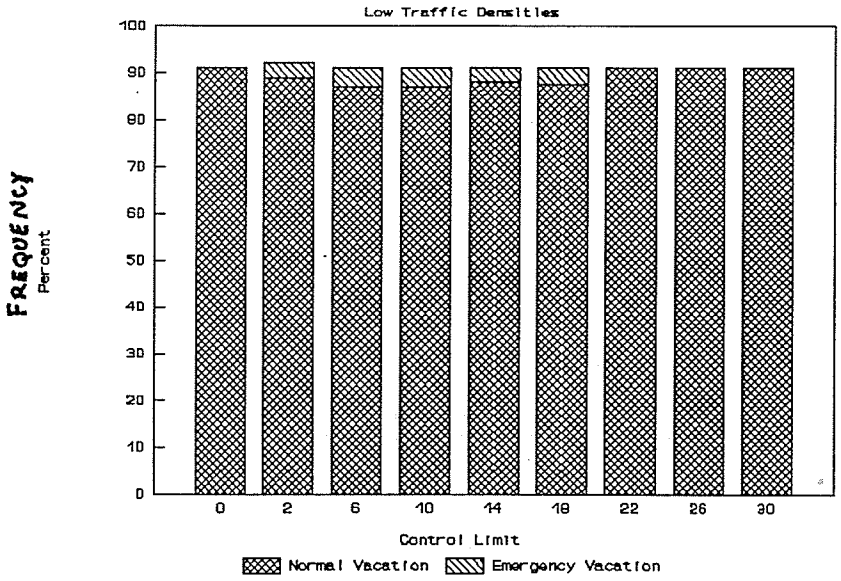
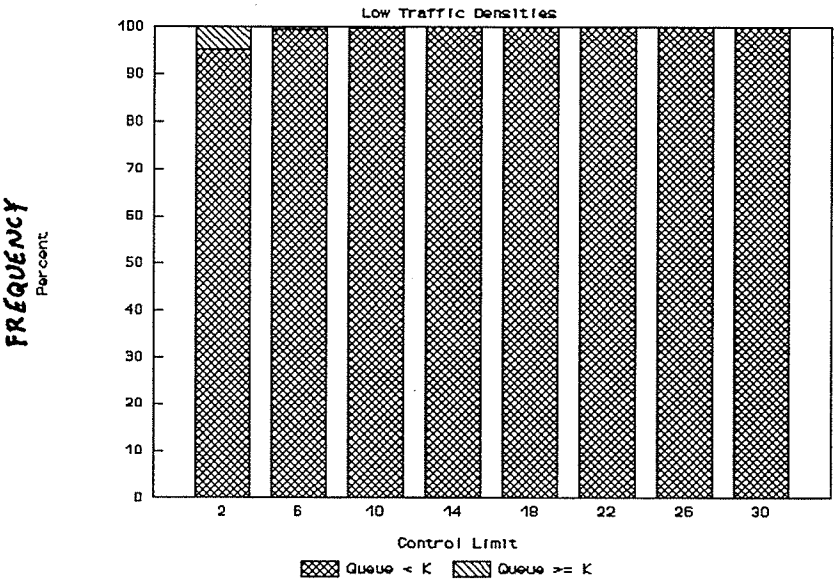


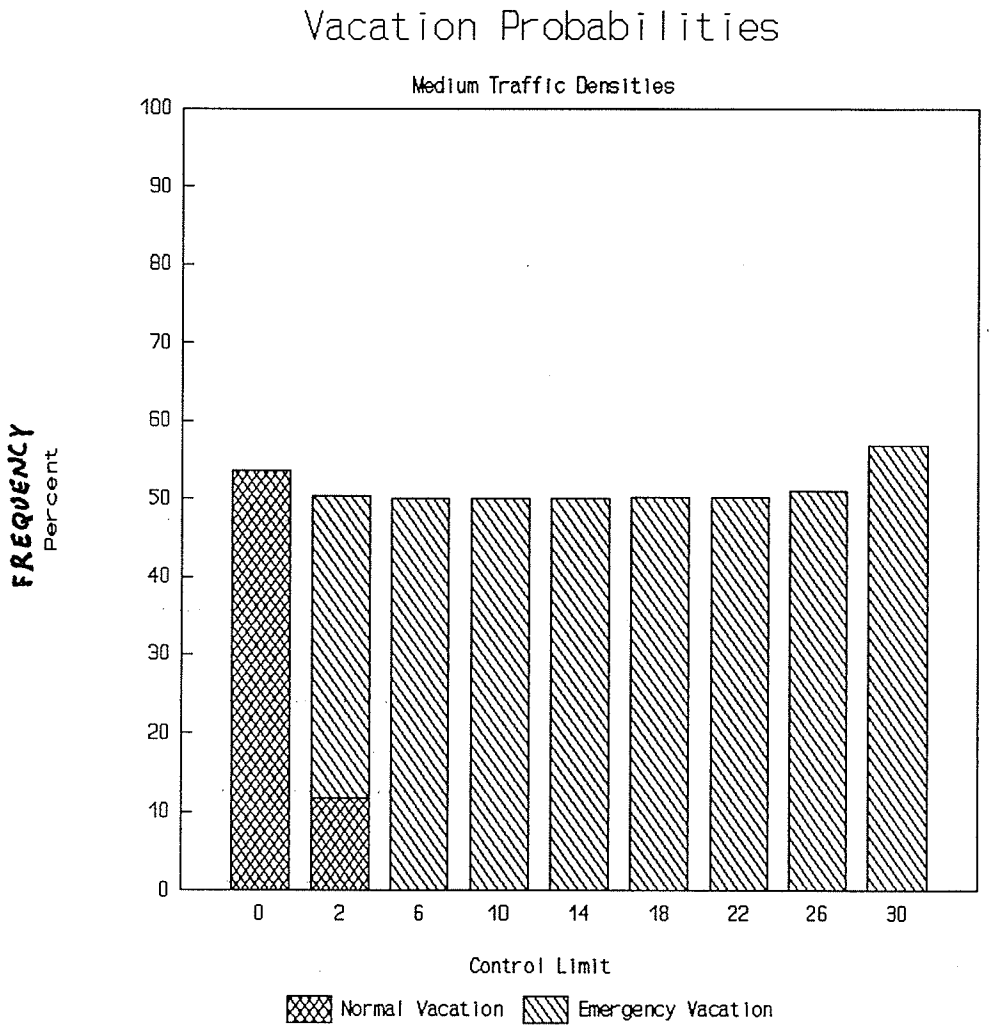
Figure 6.3-2

Target Queue Length While On Vacation



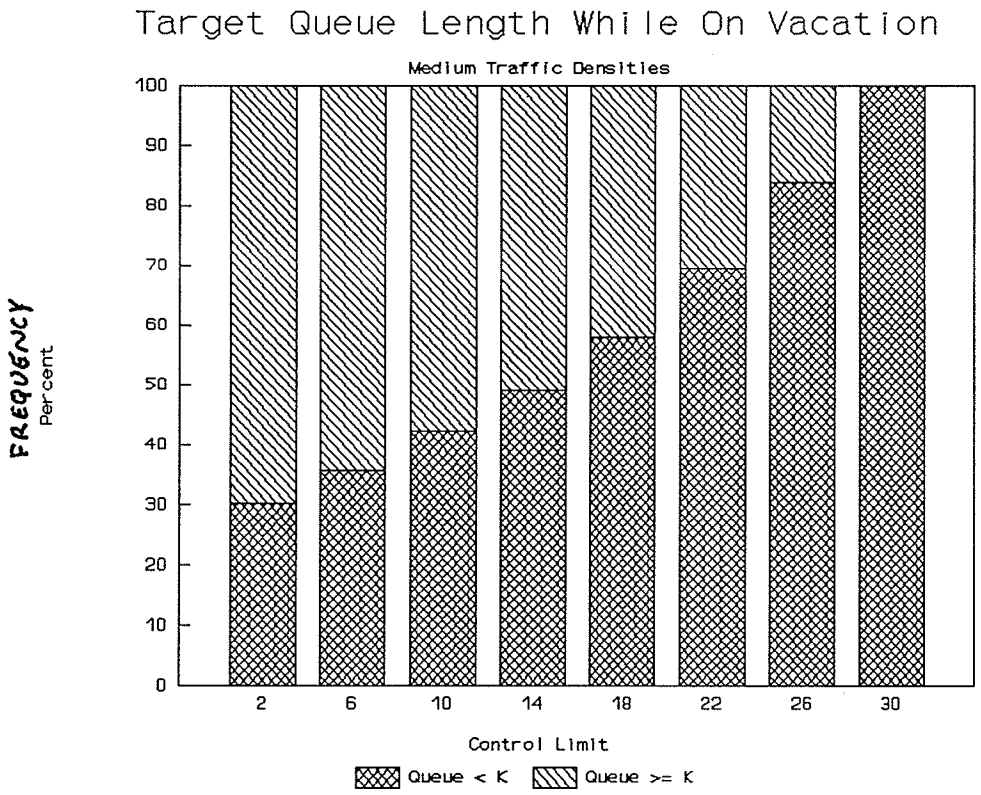
vacations are only taken for the case when $K=2$ (and the no control situation, of course). All other K values yield emergency vacations.

Figure 6.3-3



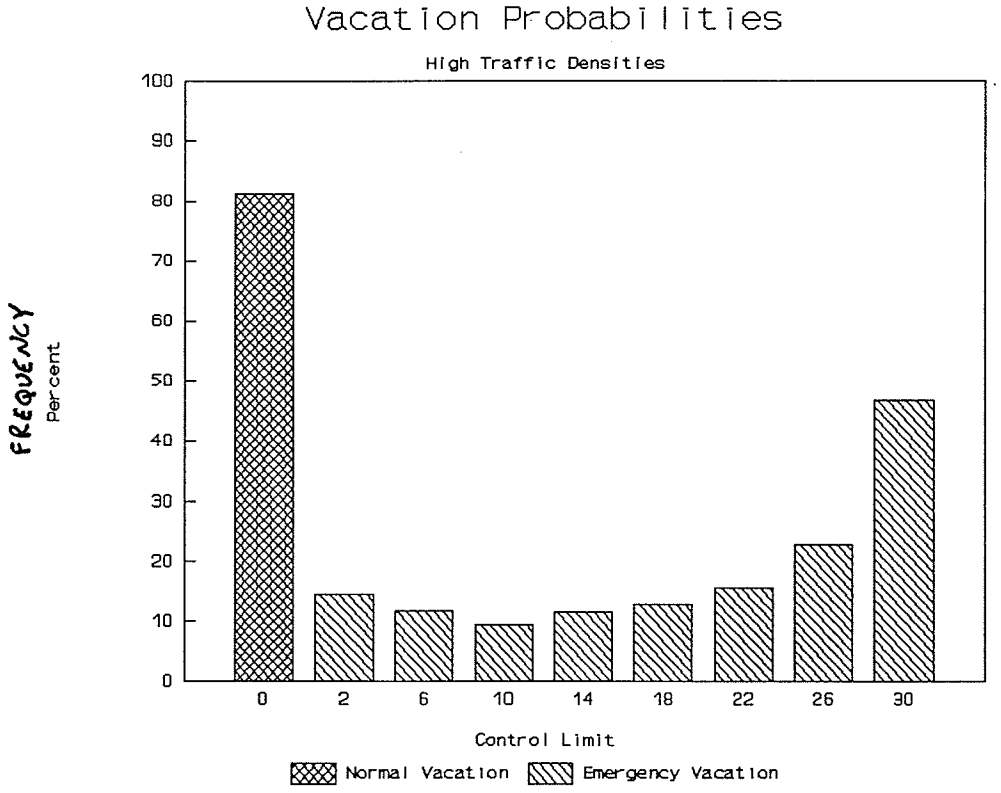
The target queue exceeds K while the server is on vacation is equal to approximately 70% when $K=2$ (See Figure 6.3-4). This portion decreases steadily until it reaches zero when $K=N$. This makes sense as medium values for ρ yield the most fluctuations in queue length. As control level increases, there are fewer opportunities for the queue length to exceed K, yet remain on vacation.

Figure 6.3-4



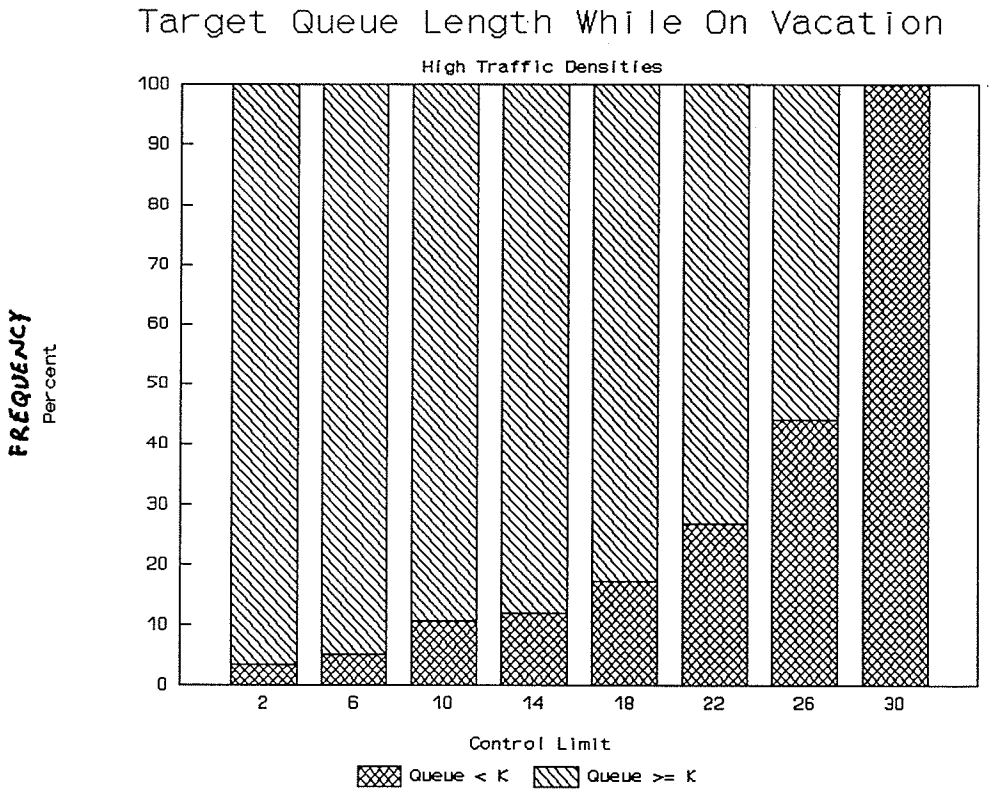
In running the experiments with high traffic densities, the implementation of control limit is shown to have a very positive impact on keeping the server in the target queue. Without any control, the server is on vacation over 80% of the time. This value drops and remains under 20% for $2 < K < 26$ as shown in Figure 6.3-5. If the polling system being controlled has one queue with higher priority customers, these results would indicate that an overload strategy such as this one would be beneficial.

Figure 6.3-5



When working with high traffic densities, this control strategy usually calls the server back from vacation shortly after the target queue exceeds *K*. Thus, the percentage of time that the target queue exceeds the control limit while the server is on vacation is minimal for low *K*. *As* K increases, this percentage increases slowly at first and rapidly as *K* approaches *N*. The graphical output for these program runs are provided in Figure 6.3-6

Figure 6.3-6



6.4 System Occupancy

System occupancy performance measures are of interest in determining the utilization of the server and the buffer. It would be beneficial if control strategy causes the average queue lengths to decline. If this is so, then the network may be designed so that the server performs more work. To observe this effect, the program provides the following output:

- **IDLE:** The portion of time that the target queue is empty. This measure indicates the amount of time that could be utilized for additional work. It is defined by:

$$\sum_{k=1}^{2} P_{0,0,k}$$

- **MEAN:** The average length of the target queue at steady state. It is defined by:

$$\sum_{j=1}^{N} (P_{1,j}+P_{0,j,1}+P_{0,j,2}+P_{2,j,1}+P_{2,j,2}) * j$$

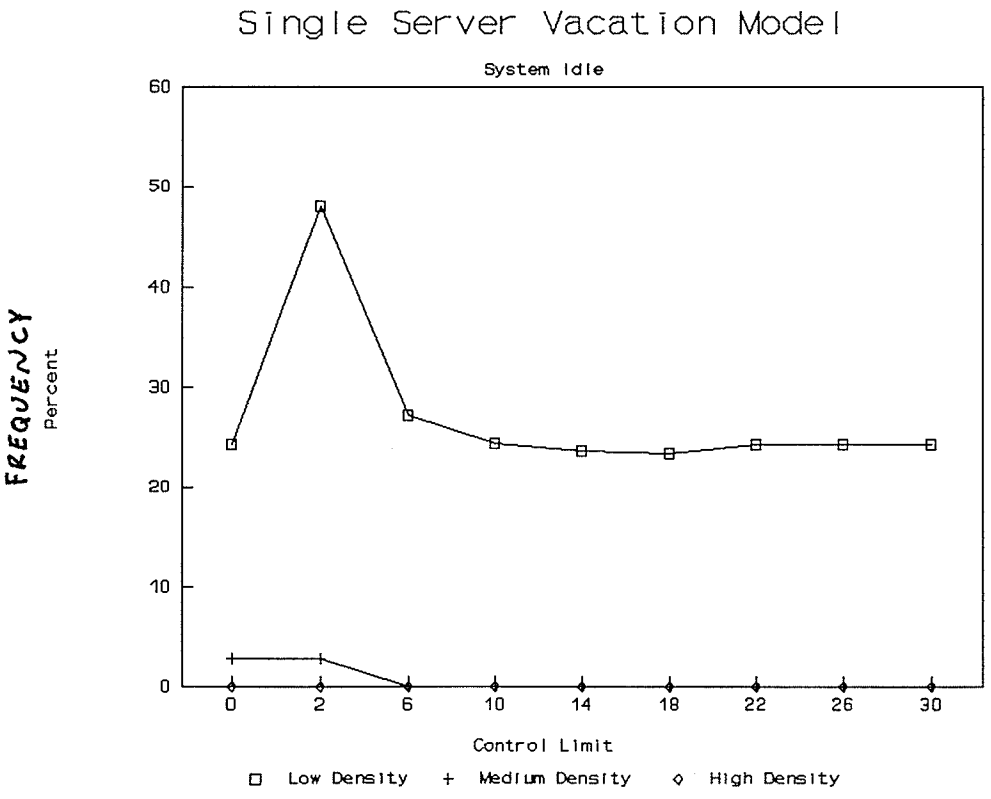
- **SDEV:** The standard deviation of the average queue length. It is defined by:

$$\sqrt{\sum_{j=1}^{N} (P_{1,j}+P_{0,j,1}+P_{0,j,2}+P_{2,j,1}+P_{2,j,2}) * j^2 - MEAN^2}$$

The values for performance measure **IDLE** are presented graphically in Figure 6.4-1. From this figure, it is seen that the control strategy has the most effect on systems having low traffic densities. For this case, the idle percentage doubles from 25% for the no control case to 50% when $K=2$. This benefit then declines rapidly with increasing K until it settles back down to 25% when the control limit is set at ten or greater.

The advantages of the control strategy are not as noticeable when using medium or high values of ρ. This is due to the fact that arrivals occur at rates that prevent the server from being idle to often.

Figure 6.4-1



The experimental results of the average queue length analysis are provided in Figure 6.4-2. In this figure, the mean values are shown without their corresponding standard distributions to prevent a "busy" graph. Refer to Table 6.4-1 for the experimental data. The table indicates that the distributions tighten as K increases.

Figure 6.4-2

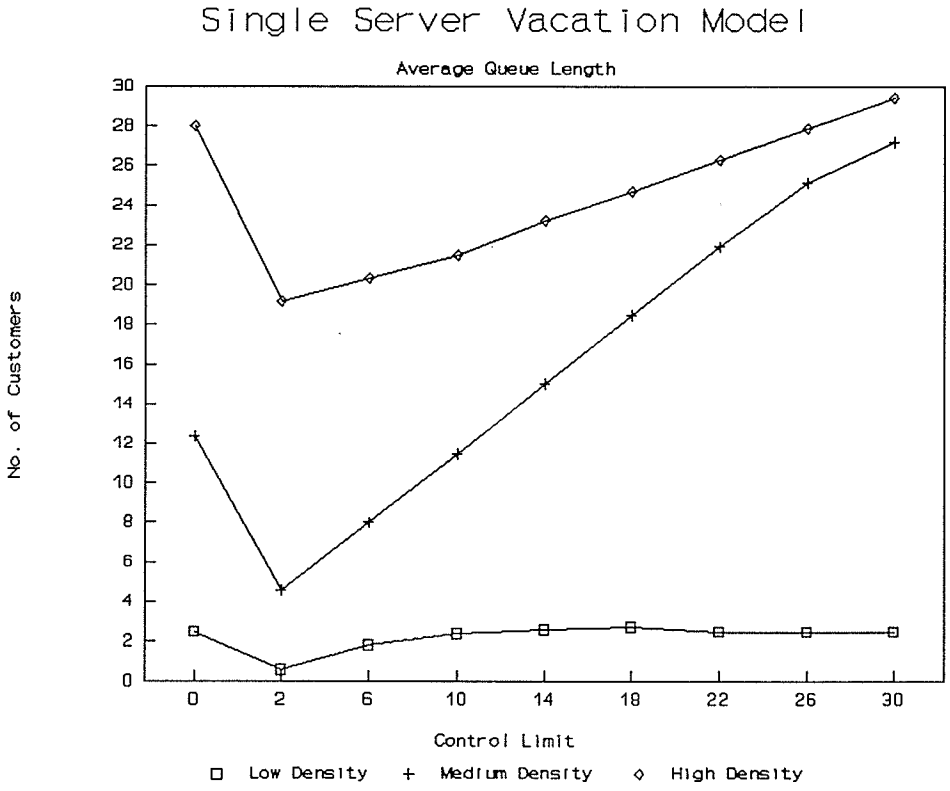


Table 6.4-1: Experimental Data for Average Queue Length

Traffic Density	**Control Limit**								
	0	**2**	**6**	**10**	**14**	**18**	**22**	**26**	**30**
0.1: Mean	2.5	0.6	1.8	2.4	2.6	2.7	2.5	2.5	2.5
Sdev	2.7	0.7	1.6	2.3	2.6	2.8	2.6	2.7	2.7
0.5: Mean	12.4	4.6	8	11.5	15	18.5	21.9	25.1	27.2
Sdev	8.7	3.5	3.2	2.9	2.6	2.4	2.2	2.2	2.7
1.0: Mean	28	19.2	20.3	21.5	23.2	24.7	26.3	27.9	29.4
Sdev	5.2	8.2	7	5.9	4.9	3.8	2.7	1.7	0.6

The average queue length analysis provides further evidence that the overload control strategy yields significant benefits. It is seen that average queue length drops immediately upon control limit implementation. Then, as the value of K increases the queue lengths climbs again until it becomes longer than the no control case.

This analysis shows that the advantages of the control strategy increases with a corresponding increase in ρ. For this set of experiments, maximum decreases for low, medium and high traffic density are 2, 7 and 8, respectively. Once the ρ value becomes too large, however, the system will become so congested that the control mechanism will lose its effectiveness.

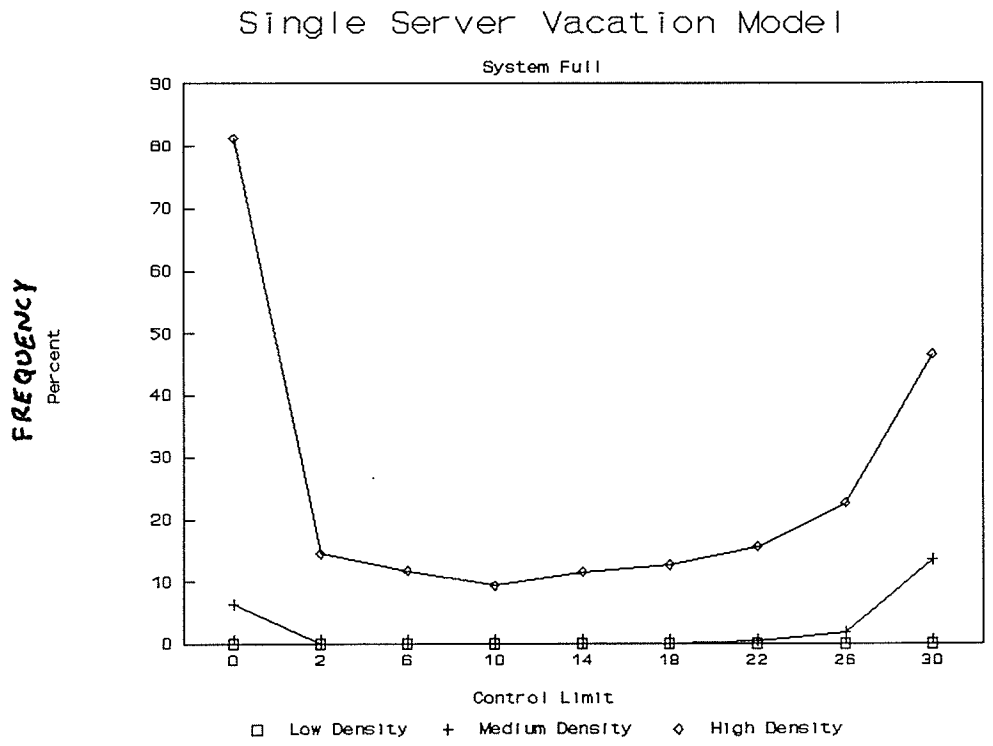
6.5 Loss Frequency

Minimizing customer loss to a full buffer is the most important design aspect. Since all buffers are finite, these loss instances do occur and when they do, expensive retransmissions costs are incurred. If an overload control scheme is found to cut down the loss frequency, it would be economically desirable to implement it. To perform this analysis, the program produces the following output:

- **FULL:** The probability that a customer arrives to a full buffer. It is defined by:

$$\sum_{k=1}^{2} (P_{0,N,k}+P_{2,N,k}) + P_{1,N}$$

Figure 6.5-1



The results from the program executions are graphed in Figure 6.5-1. From this graph, it is apparent that the control strategy does in fact provide for improved performance. When working with low traffic densities, the probability that the buffer is full is essentially zero even when no control is in place. Therefore, the control limit is unable to provide any improvement in this case.

The control strategy provides marked decreases in job losses using networks having medium or high traffic densities. Although maximum benefits occur when low K is used, the strategy only results in higher losses when medium ρ is combined with $K=N$. The greatest benefit occurs with $\rho=1$. Here, the maximum reduction of over 70% occurs at a middle control limit value ($K=10$).

CHAPTER 7
AREAS FOR FURTHER RESEARCH

This thesis presents the first stage of analysis for an overload control strategy. As such, further research can take a number of directions. One of these is to study the system without the assumption of symmetric queues. If this is the case, the simulation would have to be coded to allow for different control limits, arrival rates and service rates for each queue. This would lead to different input values for vacation parameters to the single server vacation analysis.

In many polling systems, customers are assigned to a specific queue according to their priority. By employing priorities, important customers can be given special protection against losses. A variety of ways of modelling these priority schemes exist and all would effect the simulation program and associated vacation parameters.

The analysis in this thesis is limited to the steady state case. Therefore, another way in which the effectiveness of the control strategy can be measured is to study how it behaves under transient conditions. This can be accomplished by solving the system of simultaneous differential equations that describe the flow of customers through the network. Solutions to equations of this type have been handled in the literature by Runge-Kutta techniques.

Another related strategy utilizes two control limits. This may improve network performance and should be evaluated. In this scheme the server may be called away to any queue with length exceeding K_1. Once it is called away, it cannot be called away again until the queue length decreases below the lower control limit K_2. This would prevent the server from spending an excessive amount of time moving between buffers due to call aways.

CHAPTER 8
CONCLUSIONS

In the design of communication networks, a single server is often required to perform a multitude of tasks in a configuration known as a polling system. As the space available for jobs to line up in is limited, it is important to minimize losses resulting from customers arriving to a full queue. Most of the literature in congestion control, however, is dedicated to the modelling of infinite queues. Therefore, the objective of this thesis is to evaluate a specific control strategy for a polling system under a finite queue context.

The polling system was to be solved as a single server vacation model. This method has been proven successful in modelling polling systems. Before the evaluation could be performed, an investigation of vacation behaviour is required. This was accomplished through the use of computer simulation.

Analysis of the single server vacation model was then achieved through a **FORTRAN** computer program. The overload control strategy was then evaluated in terms of vacation behaviour, system occupancy and loss frequency. A number of conclusions can be made by observing these performance measures.

If the network being designed is expected to operate under conditions of low traffic density, the problem of lost customers does not exist. Benefits of implementing the control limit in this situation are related to increased idleness and lower average queue lengths. By increasing idleness, the server can actually be allocated a greater

amount of work. Also, decreasing queue length will lead to a corresponding decrease in customer waiting time and increased system throughput. Improvements are maximized with lower control limits and decrease with increasing K.

Networks having medium traffic densities saw improvements when the congestion control scheme was used. Here also, maximum benefits are seen with low K values. In these networks, decreases in average queue lengths and loss frequency occur leading to efficient system performance.

The most dramatic improvements are seen when modelling networks subjected to high traffic densities. In these experiments, the loss frequency is minimized by using a median control level. The number of lost customers was decreased over 70% with this control strategy. Average queue lengths were most favourably affected by low K values. Therefore, choice of control level would depend on whether customer losses or waiting time was the priority.

It can be concluded that this particular congestion control strategy is an effective method for improving system performance. The results indicate that further analysis into this field, as suggested in the previous chapter, is encouraged.

APPENDIX A
FEASIBLE STATE TRANSITION RATES

This appendix provides valid transitions using the state space definition in Section 4.2.

From	To	Rate	
(1,j) →	(1,j+1) →	λ	for $1 \leq j < N$
(1,1) →	(0,0,1) →	$\phi\mu_S$	
(1,1) →	(0,0,2) →	$(1-\phi)\mu_S$	
(1,j) →	(2,j-1,1) →	$\Omega\phi\mu_S$	for $2 \leq j \leq K$
(1,j) →	(2,j-1,2) →	$\Omega(1-\phi)\mu_S$	for $2 \leq j \leq K$
(1,j) →	(1,j-1) →	$(1-\Omega)\mu_S$	for $2 \leq j \leq K$
(1,j) →	(1,j-1) →	μ_S	for $K < j \leq N$
(0,0,k) →	(0,1,k) →	λ	for k=1,2
(i,j,k) →	(i,j+1,k) →	λ	for i=0,2; $1 \leq j < K-1$; k=1,2
(i,j,k) →	(i,j+1,k) →	$(1-\theta)\lambda$	for i=0,2; $K-1 \leq j < N$; k=1,2
(i,j,k) →	(1,j+1) →	$\theta\lambda$	for i=0,2; $K-1 \leq j < N$; k=1,2
(0,j,1) →	(0,j,2) →	$\alpha\mu_{V1}$	for $1 \leq j \leq N$
(0,j,1) →	(1,j) →	$(1-\alpha)\mu_{V1}$	for $1 \leq j \leq N$
(0,j,2) →	(0,j,1) →	$\beta\mu_{V2}$	for $1 \leq j \leq N$
(0,j,2) →	(1,j) →	$(1-\beta)\mu_{V2}$	for $1 \leq j \leq N$
(2,j,1) →	(2,j,2) →	$\alpha\mu_{E1}$	for $1 \leq j \leq N$
(2,j,1) →	(1,j) →	$(1-\alpha)\mu_{E1}$	for $1 \leq j \leq N$
(2,j,2) →	(2,j,1) →	$\beta\mu_{E2}$	for $1 \leq j \leq N$
(2,j,2) →	(1,j) →	$(1-\beta)\mu_{E2}$	for $1 \leq j \leq N$

APPENDIX B
VACATION MODEL PROGRAM LISTING

This appendix provides the **FORTRAN** code used to solve the single server vacation model.

```
C  THIS PROGRAM USES GRASSMANN'S ALGORITHM TO COMPUTE THE
C  STEADY STATE DISTRIBUTION FOR THE SINGLE SERVER, VACATION
C  MODEL.
C
C
      REAL AR,SV,NV1,NV2,EV1,EV2,VB1,M12,M21
      REAL NCB,NCA,P(252,252),PR(252)
      INTEGER N,K,NS
C
      CALL INITIAL(AR,SV,NV1,NV2,EV1,EV2,VB1,M12,
     * M21,NCB,NCA,N,K)
       IF(K.EQ.1)THEN
        NS=3*N+2
       ELSE
        NS=5*N+2
       ENDIF
      CALL RATE(AR,SV,NV1,NV2,EV1,EV2,VB1,M12,
     * M21,NCB,NCA,N,K,P,PR,NS)
      CALL PROB(P,NS)
      CALL SOLVE(P,PR,NS)
      CALL OUTPUT(PR,SV,K,N,NS)
 2    CONTINUE
C
      STOP
      END
C
C  THIS SUBROUTINE IS USED TO INITIALIZE THE INPUT VARIABLES
C
      SUBROUTINE INITIAL(AR,SV,NV1,NV2,EV1,EV2,VB1,M12,
     * M21,NCB,NCA,N,K)
C
      REAL AR,SV,NV1,NV2,EV1,EV2,VB1,M12,M21,NCB,NCA
      INTEGER N,K
C
      WRITE(*,1)
 1    FORMAT(3X,'MEAN',4X,'SDEV',4X,'IDLE',4X,'FULL',4X,'NVAC'
     *  ,4X,'EVAC',4X,'VLTK',4X,'VGEK')
      AR=100
```

```
      SV=100
      K=5
      NV1=0.35
      NV2=0.35
      EV1=0.35
      EV2=0.35
      NCB=.87
      NCA=0
      VB1=1
      M12=1
      M21=0
      N=30
C
      RETURN
      END
C
C  THIS SUBROUTINE DEFINES THE TRANSITION RATE MATRIX FOR
C  THE SYSTEM
C
      SUBROUTINE RATE(AR,SV,NV1,NV2,EV1,EV2,VB1,M12,
     * M21,NCB,NCA,N,K,P,PR,NS)
C
      REAL AR,SV,NV1,NV2,EV1,EV2,VB1,M12,M21
      REAL NCB,NCA,P(NS,NS),PR(NS)
      INTEGER N,K,I,J,NS,ROW
C
      DO 100 I=1,NS
       PR(I)=0
       DO 105 J=1,NS
 105     P(I,J)=0
 100  CONTINUE
C
      ROW=1
      P(ROW,1)=-(AR+SV)
      P(ROW,2)=AR
      P(ROW,N+1)=VB1*SV
      P(ROW,2*N+2)=(1-VB1)*SV
      ROW=ROW+1
C
      IF(K.EQ.1)THEN
       DO 110 I=2,N-1
        P(ROW,ROW-1)=SV
        P(ROW,ROW)=-(AR+SV)
        P(ROW,ROW+1)=AR
        ROW=ROW+1
 110    CONTINUE
```

```
      ELSE
       DO 115 I=2,K
        P(ROW,ROW-1)=SV*NCA
        IF(I.EQ.N)THEN
         P(ROW,ROW)=-SV
        ELSE
         P(ROW,ROW)=-(AR+SV)
        ENDIF
        IF(I.NE.N) P(ROW,ROW+1)=AR
        P(ROW,3*N+1+ROW)=SV*(1-NCA)*VB1
        P(ROW,4*N+1+ROW)=SV*(1-NCA)*(1-VB1)
        ROW=ROW+1
  115   CONTINUE
       IF(K.NE.N)THEN
        DO 120 I=K+1,N-1
         P(ROW,ROW-1)=SV
         P(ROW,ROW)=-(AR+SV)
         P(ROW,ROW+1)=AR
         ROW=ROW+1
  120    CONTINUE
       ENDIF
      ENDIF
      IF(K.NE.N)THEN
       P(ROW,ROW-1)=SV
       P(ROW,ROW)=-SV
       ROW=ROW+1
      ENDIF
C
      IF(K.EQ.1)THEN
       P(ROW,1)=AR*(1-NCB)
       P(ROW,ROW)=-(AR+NV1)
       P(ROW,ROW+1)=AR*NCB
       P(ROW,ROW+N+1)=NV1
       ROW=ROW+1
       DO 125 I=1,N-1
        P(ROW,1)=(1-M12)*NV1
        P(ROW,I+1)=AR*(1-NCB)
        P(ROW,ROW)=-(AR+NV1)
        P(ROW,ROW+1)=AR*NCB
        P(ROW,ROW+N+1)=M12*NV1
        ROW=ROW+1
  125   CONTINUE
      ELSE
       P(ROW,ROW)=-(AR+NV1)
       P(ROW,ROW+1)=AR
       P(ROW,ROW+N+1)=NV1
```

```
      ROW=ROW+1
      DO  130 I=1,K-2
       P(ROW,I)=(1-M12)*NV1
       P(ROW,ROW)=-(AR+NV1)
       P(ROW,ROW+1)=AR
       P(ROW,ROW+N+1)=M12*NV1
       ROW=ROW+1
 130   CONTINUE
      DO 135 I=K-1,N-1
       P(ROW,I)=(1-M12)*NV1
       P(ROW,I+1)=(1-NCB)*AR
       P(ROW,ROW)=-(AR+NV1)
       P(ROW,ROW+1)=AR*NCB
       P(ROW,ROW+N+1)=M12*NV1
       ROW=ROW+1
 135   CONTINUE
     ENDIF
     P(ROW,N)=(1-M12)*NV1
     P(ROW,ROW)=-NV1
     P(ROW,ROW+N+1)=M12*NV1
     ROW=ROW+1
C
     IF(K.EQ.1)THEN
      P(ROW,1)=AR*(1-NCB)
      P(ROW,ROW)=-(AR+NV2)
      P(ROW,ROW+1)=AR*NCB
      P(ROW,ROW-(N+1))=NV2
      ROW=ROW+1
      DO 140 I=1,N-1
       P(ROW,1)=(1-M21)*NV2
       P(ROW,I+1)=AR*(1-NCB)
       P(ROW,ROW)=-(AR+NV2)
       P(ROW,ROW+1)=AR*NCB
       P(ROW,ROW-(N+1))=M21*NV2
       ROW=ROW+1
 140   CONTINUE
     ELSE
      P(ROW,ROW)=-(AR+NV2)
      P(ROW,ROW+1)=AR
      P(ROW,ROW-(N+1))=NV2
      ROW=ROW+1
      DO  145 I=1,K-2
       P(ROW,I)=(1-M21)*NV2
       P(ROW,ROW)=-(AR+NV2)
       P(ROW,ROW+1)=AR
       P(ROW,ROW-(N+1))=M21*NV2
```

```
      ROW=ROW+1
145   CONTINUE
      DO 150 I=K-1,N-1
       P(ROW,I)=(1-M21)*NV2
       P(ROW,I+1)=(1-NCB)*AR
       P(ROW,ROW)=-(AR+NV2)
       P(ROW,ROW+1)=AR*NCB
       P(ROW,ROW-(N+1))=M21*NV2
       ROW=ROW+1
150   CONTINUE
     ENDIF
     P(ROW,N)=(1-M21)*NV2
     P(ROW,ROW)=-NV2
     P(ROW,ROW-(N+1))=M21*NV2
     ROW=ROW+1
C
     IF(K.NE.1)THEN
      DO 155 I=1,K-2
       P(ROW,I)=(1-M12)*EV1
       P(ROW,ROW)=-(AR+EV1)
       P(ROW,ROW+1)=AR
       P(ROW,ROW+N)=M12*EV1
       ROW=ROW+1
155   CONTINUE
      DO 160 I=K-1,N-1
       P(ROW,I)=(1-M12)*EV1
       P(ROW,I+1)=(1-NCB)*AR
       P(ROW,ROW)=-(AR+EV1)
       P(ROW,ROW+1)=AR*NCB
       P(ROW,ROW+N)=M12*EV1
       ROW=ROW+1
160   CONTINUE
      P(ROW,N)=(1-M12)*EV1
      P(ROW,ROW)=-EV1
      P(ROW,NS)=M12*EV1
      ROW=ROW+1
      DO 165 I=1,K-2
       P(ROW,I)=(1-M21)*EV2
       P(ROW,ROW)=-(AR+EV2)
       P(ROW,ROW+1)=AR
       P(ROW,ROW-N)=M21*EV2
       ROW=ROW+1
165   CONTINUE
      DO 170 I=K-1,N-1
       P(ROW,I)=(1-M21)*EV2
       P(ROW,I+1)=(1-NCB)*AR
```

```
      P(ROW,ROW) = -(AR + EV2)
      P(ROW,ROW + 1) = AR*NCB
      P(ROW,ROW-N) = M21*EV2
      ROW = ROW + 1
 170  CONTINUE
     P(ROW,N) = (1-M21)*EV2
     P(ROW,ROW) = -EV2
     P(ROW,ROW-N) = M21*EV2
     ROW = ROW + 1
     ENDIF
C
     RETURN
     END
C
C  THIS SUBROUTINE TAKES THE TRANSITION RATE MATRIX
C  DEFINED IN SUBROUTINE RATE AND DISCRETIZES IT
C
     SUBROUTINE PROB(P,NS)
C
     REAL P(NS,NS),MAX,DT
     INTEGER NS,I,J
C
     MAX = 0
     DO 200 I = 1,NS
      IF(ABS(P(I,I)).GT.MAX) MAX = ABS(P(I,I))
 200  CONTINUE
     DT = 0.9/MAX
     DO 205 I = 1,NS
      DO 210 J = 1,NS
       IF(I.EQ.J) THEN
        P(I,J) = (P(I,J)*DT) + 1
       ELSE
        P(I,J) = P(I,J)*DT
       ENDIF
 210   CONTINUE
 205  CONTINUE
C
     RETURN
     END
C
C  THIS SUBROUTINE USES GRASSMAN'S ALGORITHM TO SOLVE
C  FOR THE STEADY STATE OF THE SYSTEM
C
     SUBROUTINE SOLVE(A,P,NS)
C
     REAL A(NS,NS),P(NS),S,TOT
```

```
      INTEGER NS,I,J,N
C
      DO 300 N=NS,2,-1
       S=0
       DO 305 J=1,N-1
 305     S=S+A(N,J)
       DO 310 I=1,N-1
 310     A(I,N)=A(I,N)/S
       DO 315 I=1,N-1
        IF(A(I,N).EQ.0) GOTO 315
        DO 320 J=1,N-1
 320      A(I,J)=A(I,J)+(A(I,N)*A(N,J))
 315    CONTINUE
 300    CONTINUE
C
      TOT=1
      P(1)=1
C
      DO 325 J=2,NS
       S=0
       DO 330 I=1,J-1
 330     S=S+P(I)*A(I,J)
       P(J)=A(1,J)+S
       TOT=TOT+P(J)
 325   CONTINUE
C
      DO 335 I=1,NS
 335    P(I)=P(I)/TOT
C
      RETURN
      END
C
C  THIS SUBROUTINE CALCULATES AND PRINTS THE SYSTEM
C  PERFORMANCE MEASURES
C
      SUBROUTINE OUTPUT(P,SV,K,N,NS)
C
      INTEGER I,N,K,NS,L,COUNT
      REAL SV,P(3000),MEAN,VAR,SDEV
      REAL IDLE,FULL,NVAC,EVAC,VLTK,VGEK
C
      MEAN=0
      VAR=0
      NVAC=0
      EVAC=0
      VLTK=0
```

```
      VGEK=0
      DO 300 I=1,N
       IF(K.EQ.1)THEN
        MEAN=MEAN+(P(I)+P(I+N+1)+P(2*N+I+2))*I
        VAR=VAR+(P(I)+P(I+N+1)+P(2*N+I+2))*(I**2)
       ELSE
        MEAN=MEAN+(P(I)+P(I+N+1)+P(2*N+I+2)+P(3*N+I+2)
     *    +P(4*N+I+2))*I
        VAR=VAR+(P(I)+P(I+N+1)+P(2*N+I+2)+P(3*N+I+2)
     *    +P(4*N+I+2))*(I**2)
       ENDIF
  300  CONTINUE
      VAR=VAR-(MEAN**2)
      SDEV=SQRT(ABS(VAR))
      IDLE=P(N+1)+P(2*N+2)
      IF(K.EQ.1)THEN
       FULL=P(N)+P(2*N+1)+P(3*N+2)
      ELSE
       FULL=P(N)+P(2*N+1)+P(3*N+2)+P(4*N+2)+P(5*N+2)
      ENDIF
      DO 310 I=N+1,3*N+2
  310   NVAC=NVAC+P(I)
      DO 320 I=3*N+3,NS
  320   EVAC=EVAC+P(I)
      L=N+1
      DO 330 I=1,4
        IF(I.LE.2)THEN
         COUNT=0
        ELSE
         COUNT=1
        ENDIF
        DO 340 J=COUNT,K-1
         VLTK=VLTK+P(L)
         L=L+1
  340   CONTINUE
        DO 350 J=K,N
         VGEK=VGEK+P(L)
         L=L+1
  350   CONTINUE
  330   CONTINUE
      IDLE=IDLE*100
      FULL=FULL*100
        NVAC=NVAC*100
        EVAC=EVAC*100
        VLTK=VLTK*100
        VGEK=VGEK*100
```

```
      WRITE(*,410) MEAN,SDEV,IDLE,FULL,NVAC,EVAC,VLTK,VGEK
410   FORMAT(F7.1,2X,F6.1,6(3X,F5.1))
      RETURN
      END
```

APPENDIX C
SIMULATION PROGRAM LISTING

This appendix provides the **PCModel** coding used to simulate the operation of a three queue polling system.

```
O=(=)
%ARRIV1=(1:0)
%ARRIV2=(1:0)
%ARRIV3=(1:0)
%SERV=(1:0)
%NVBEG=(0)
%EVBEG=(0)
%VACEND=(0)
%EVAC=(0)
%NVAC=(0)
;
*INIT=(XY(3,2))
*ENTRY1=(XY(10,4))
*ENTRY2=(XY(25,4))
*ENTRY3=(XY(40,4))
*SERV1=(XY(10,14))
*SERV2=(XY(25,14))
*SERV3=(XY(40,14))
*K1=(XY(7,9))
*K2=(XY(22,9))
*K3=(XY(37,9))
*TIMESCA=(XY(65,6))
*TIMESCB=(XY(65,13))
*NOSCA=(XY(65,5))
*NOSCB=(XY(65,12))
*PERSCA=(XY(65,7))
*PERSCB=(XY(65,14))
*EVCTN=(XY(65,18))
*NVCTN=(XY(65,17))
*NV=(XY(30,17))
*EV=(XY(30,18))
*NV1=(XY(38,17))
*EV1=(XY(38,18))
;
#K=(9)
#SEED=(160)
;
@COUNT=(0)
```

```
@Q1 = (0)
@Q2 = (0)
@Q3 = (0)
@TIMESCA = (0)
@TIMESCB = (0)
@NOSCA = (0)
@NOSCB = (0)
@NV = (0)
@EV = (0)
;
&PER = (0)
;
J = (1,1,1,0,0,0,32767)
J = (2,2,2,0,0,0,32767)
J = (3,3,3,0,0,0,32767)
J = (20,X,4,0,0,0,1)
;
 U = (1,QUEUE 1,*SERV1)
 U = (2,QUEUE 2,*SERV2)
 U = (3,QUEUE 3,*SERV3)
;
BL(!SV)
        WT(%SERV)
        RV(E,%SERV,2:00)
EL
;
BL(!NOSCA)
        IV(@NOSCA)
        PV(*NOSCA,@NOSCA)
EL
;
BL(!TIMESCA)
        IV(@TIMESCA)
        PV(*TIMESCA,@TIMESCA)
EL
;
BL(!PER)
        IV(@COUNT)
        IF(@COUNT,GE,5000,THEN,:OUT)
                SV(@NOSCA,0)
                SV(@TIMESCA,0)
                SV(@NOSCB,0)
                SV(@TIMESCB,0)
                SV(&PER,0)
                SV(%EVAC,0)
                SV(%NVAC,0)
```

```
            SV(@NV,0)
            SV(@EV,0)
        :OUT
         IF(@COUNT,EQ,30000,THEN,NEXT,ELSE,:GO)
            AO(%NVAC,/,@NV)
            AO(%NVAC,/,0:42)
            PV(*NVCTN,%NVAC)
            AO(%EVAC,/,@EV)
            AO(%EVAC,/,0:42)
            PV(*EVCTN,%EVAC)
            QX
        :GO
         SV(&PER,@TIMESCA)
         AO(&PER,*,100)
         IF(@NOSCA,EQ,0,:JUMP)
         AO(&PER,/,@NOSCA)
         PV(*PERSCA,&PER)
         SV(&PER,@TIMESCB)
         AO(&PER,*,100)
         IF(@NOSCB,EQ,0,:JUMP)
         AO(&PER,/,@NOSCB)
         PV(*PERSCB,&PER)
        :JUMP
EL
;
BL(!NOSCB)
        IV(@NOSCB)
        PV(*NOSCB,@NOSCB)
EL
;
BL(!TIMESCB)
        IV(@TIMESCB)
        PV(*TIMESCB,@TIMESCB)
EL
;
BL(!VCTN)
        PV(*EVCTN,%EVAC)
        PV(*NVCTN,%NVAC)
        PV(*EV1,@EV)
        PV(*NV1,@NV)
EL
;
;
BR(4,*INIT,0)
        RS(#SEED)
        PO(*SERV2)
```

```
        PO(*SERV3)
        PV(XY(21,15),#K)
ER
;
BR(1,*ENTRY1,%ARRIV1)
        RV(E,%ARRIV1,1:0)
        IV(@Q1)
        IF(@Q1,GE,#K,THEN,NEXT,ELSE,:K1D)
                PO(*K1)
                JC(*SERV1,:K1D)
                LK(!NOSCB)
       :K1D
        MD(8,0)
        TP(*SERV1)
        JC(*NV,*EV,:CALLED)
        JB(*NV,:NV)
        CL(*EV)
        SV(%VACEND,CLOCK)
        AO(%VACEND,-,%EVBEG)
        IV(@EV)
        AO(%EVAC,+,%VACEND)
        LK(!VCTN)
        JP(:CALLED)
       :NV
        CL(*NV)
        SV(%VACEND,CLOCK)
        AO(%VACEND,-,%NVBEG)
        IV(@NV)
        AO(%NVAC,+,%VACEND)
        LK(!VCTN)
       :CALLED
        LK(!SV)
        DV(@Q1)
        IF(@Q1,LT,#K,THEN,NEXT,ELSE,:K1P)
                CL(*K1)
                LK(!NOSCA)
       :K1P
        IF(@Q1,EQ,0,THEN,NEXT,ELSE,:CONT1)
                SV(%NVBEG,CLOCK)
                PO(*NV)
                IF(@Q2,GT,0,THEN,NEXT,ELSE,:CHANGE1)
                        PO(*SERV1)
                        CL(*SERV2)
                        JP(:CONT1)
                      :CHANGE1
                IF(@Q3,GT,0,THEN,NEXT,ELSE,:UPDATE1)
```

```
                PO(*SERV1)
                CL(*SERV3)
                JP(:CONT1)
            :UPDATE1
                WE
                CL(*NV)
                JP(:K1P)
        :CONT1
         JB(*K1,:LEAVE1)
         JC(*K2,*K3,:LEAVE1)
         IF(@Q2,GE,@Q3,THEN,NEXT,ELSE,:MOVE3)
              PO(*SERV1)
              CL(*SERV2)
              LK(!TIMESCA)
              JB(*NV,:LEAVE1)
              SV(%EVBEG,CLOCK)
              PO(*EV)
              JP(:LEAVE1)
        :MOVE3
              PO(*SERV1)
              CL(*SERV3)
              LK(!TIMESCA)
              JB(*NV,:LEAVE1)
              SV(%EVBEG,CLOCK)
              PO(*EV)
              JP(:LEAVE1)
        :LEAVE1
         LK(!PER)
ER
;
BR(2,*ENTRY2,%ARRIV2)
        RV(E,%ARRIV2,1:0)
        IV(@Q2)
        IF(@Q2,GE,#K,THEN,NEXT,ELSE,:K2D)
              PO(*K2)
        :K2D
         MD(8,0)
         TP(*SERV2)
         LK(!SV)
         DV(@Q2)
         IF(@Q2,LT,#K,THEN,NEXT,ELSE,:K2P)
              CL(*K2)
        :K2P
         IF(@Q2,EQ,0,THEN,NEXT,ELSE,:CONT2)
              IF(@Q3,GT,0,THEN,NEXT,ELSE,:CHANGE2)
                   PO(*SERV2)
```

```
                CL(*SERV3)
                JP(:CONT2)
            :CHANGE2
             IF(@Q1,GT,0,THEN,NEXT,ELSE,:UPDATE2)
                PO(*SERV2)
                CL(*SERV1)
                JP(:CONT2)
            :UPDATE2
                WE
                JP(:K2P)
        :CONT2
         JB(*K2,:LEAVE2)
         JC(*K1,*K3,:LEAVE2)
         IF(@Q3,GE,@Q1,THEN,NEXT,ELSE,:MOVE1)
            PO(*SERV2)
            CL(*SERV3)
            CL(*NV)
            CL(*EV)
            JP(:LEAVE2)
        :MOVE1
            PO(*SERV2)
            CL(*SERV1)
            LK(!TIMESCB)
            CL(*NV)
            CL(*EV)
            JP(:LEAVE2)
        :LEAVE2
         LK(!PER)
ER
;
BR(3,*ENTRY3,%ARRIV3)
        RV(E,%ARRIV3,1:0)
        IV(@Q3)
        IF(@Q3,GE,#K,THEN,NEXT,ELSE,:K3D)
            PO(*K3)
        :K3D
         MD(8,0)
         TP(*SERV3)
         LK(!SV)
         DV(@Q3)
         IF(@Q3,LT,#K,THEN,NEXT,ELSE,:K3P)
            CL(*K3)
        :K3P
         IF(@Q3,EQ,0,THEN,NEXT,ELSE,:CONT3)
            IF(@Q1,GT,0,THEN,NEXT,ELSE,:CHANGE3)
                PO(*SERV3)
```

```
                CL(*SERV1)
                JP(:CONT3)
        :CHANGE3
         IF(@Q2,GT,0,THEN,NEXT,ELSE,:UPDATE3)
                PO(*SERV3)
                CL(*SERV2)
                JP(:CONT3)
        :UPDATE3
                WE
                JP(:K3P)
    :CONT3
     JB(*K3,:LEAVE3)
     JC(*K1,*K2,:LEAVE3)
     IF(@Q1,GE,@Q2,THEN,NEXT,ELSE,:MOVE2)
            PO(*SERV3)
            CL(*SERV1)
            LK(!TIMESCB)
            CL(*NV)
            CL(*EV)
            JP(:LEAVE3)
    :MOVE2
            PO(*SERV3)
            CL(*SERV2)
            CL(*NV)
            CL(*EV)
            JP(:LEAVE3)
    :LEAVE3
     LK(!PER)
ER
```

APPENDIX D
SINGLE SERVER VACATION MODEL DATA

This appendix provides the actual values input for the experimental runs. Note that $N=30$, $\mu_{V1}=\mu_{V2}$ and $\mu_{E1}=\mu_{E2}$.

Table D-1: Program Input

ρ	K	μ_{V1}, μ_{V2}	μ_{E1}, μ_{E2}	θ	Ω
0.1	0	6.3	4.5	0%	0%
	2	6.8	8	86%	8%
	6	6.8	7.8	87%	3%
	10	6.5	6.8	89%	2%
	14	6.3	5.3	90%	1%
	18	6.3	4.5	93%	1%
	22	6.3	4.5	94%	0%
	26	6.3	4.5	95%	0%
	30	6.3	4.5	100%	0%
0.5	0	5.5	5	0%	0%
	2	12	4.5	23%	70%
	6	11.3	4.3	27%	64%
	10	9.8	5.3	31%	57%
	14	8.5	6.3	36%	54%
	18	8	6.3	43%	48%
	22	7	6.8	53%	42%
	26	6	6.8	70%	34%
	30	5.5	5.3	100%	22%

Table D-1 (Continued): Program Input

ρ	K	μ_{V1}, μ_{V2}	μ_{E1}, μ_{E2}	θ	Ω
1	0	6.3	4.5	0%	0%
	2	6.8	8	86%	8%
	6	6.8	7.8	87%	3%
	10	6.5	6.8	89%	2%
	14	6.3	5.3	90%	1%
	18	6.3	4.5	93%	1%
	22	6.3	4.5	94%	0%
	26	6.3	4.5	95%	0%
	30	6.3	4.5	100%	0%

Table D-2: Program Output

ρ	K	MEAN	SDEV	IDLE	FULL	NVAC	EVAC	VLTK	VGE K
0.1	0	2.5	2.7	24.3%	0%	91.1%	0%	N/A	N/A
	2	0.6	0.7	48%	0%	89%	3.2%	95.1%	4.9
	6	1.8	1.6	27.2%	0%	87%	4.2%	99.5%	.5%
	10	2.4	2.3	24.4%	0%	87%	4.1%	99.9%	.1%
	14	2.6	2.6	23.6%	0%	88.3%	2.8%	100%	0%
	18	2.7	2.8	23.4%	0%	87.7%	3.4%	100%	0%
	22	2.5	2.6	24.3%	0%	91.1%	0%	100%	0%
	26	2.5	2.7	24.3%	0%	91.1%	0%	100%	0%
	30	2.5	2.7	24.3%	0%	91.1%	0%	100%	0%
0.5	0	12.4	8.7	2.9%	6.4%	53.6%	0%	N/A	N/A
	2	4.6	3.5	2.9%	0%	11.8%	39%	30.2%	69.8
	6	8	3.2	0%	0%	0.1%	50%	35.8%	64.2
	10	11.5	2.9	0%	0%	0%	50%	42.4%	57.6
	14	15	2.6	0%	0%	0%	50%	49.2%	50.8
	18	18.5	2.4	0%	0.1%	0%	50%	58.1%	41.9
	22	21.9	2.2	0%	0.4%	0%	50%	69.5%	30.5
	26	25.1	2.2	0%	2%	0%	51%	83.9%	16.1
	30	27.2	2.7	0%	13.6%	0%	57%	100%	0%
1.0	0	28	5.2	0%	81.2%	81.2%	0%	N/A	N/A
	2	19.2	8.2	0%	14.5%	0%	15%	3.4%	96.6
	6	20.3	7	0%	11.8%	0%	12%	5.1%	94.9
	10	21.5	5.9	0%	9.4%	0%	9%	10.6%	89.4
	14	23.2	4.9	0%	11.6%	0%	12%	12.1%	87.9
	18	24.7	3.8	0%	12.8%	0%	13%	17.2%	82.8
	22	26.3	2.7	0%	15.7%	0%	16%	26.8%	73.2
	26	27.9	1.7	0%	22.7%	0%	23%	44.1%	55.9
	30	29.4	0.6	0%	46.7%	0%	47%	100%	0%

References

[1] H. Takagi. *Analysis Of Polling Systems.* The Massachusetts Institute of Technology, 1986.

[2] H. Takagi. *Queueing Analysis of Polling Models.* ACM Computing Surveys **20-1**, pgs. 5-28, 1988.

[3] H. Levy. *Polling Systems: Applications, Modelling, and Optimization.* IEEE Transactions on Communications **38-10**, pgs. 1750-1759, 1990.

[4] D.S. Lee and B. Sengupta. *An Approximate Analysis Of A Cyclic Server Queue With Limited Service And Reservations With An Application To Satellite Communications.* QUESTA **11**, pgs. 153-178, 1992.

[5] H. Levy. *Binomial-Gated Service: A Method for Effective Operation and Optimization of Polling Systems.* IEEE Transactions on Communications **39-9**, pgs. 1341-1349, 1991.

[6] P. Tran-Gia. *Analysis of Polling Systems with General Input Process and Finite Capacity*. IEEE Transactions on Communications **40-2**, pgs. 337-344, 1992.

[7] B.T. Doshi. *Queueing Systems With Vacations - A Survey*. Queueing Systems **1**, pgs. 29-66, 1986.

[8] D. Kendall. *Some Problems in the Theory of Queues*. Journal of the Royal Statistical Society **Series B-13**, pgs. 151-185, 1951.

[9] T.T. Lee. *M/G/1/N Queue with Vacation Time and Exhaustive Service Discipline*. Operations Research **32-4**, pgs. 774-784, 1984.

[10] R.O. LaMaire. *M/G/1/N Vacation Model with Varying E-Limited Service Discipline.* QUESTA **11**, pgs. 357-375, 1992.

[11] R.O. LaMaire. *M/G/1/N Vacation Model with Varying G-Limited Service Discipline.* IBM Research Report RC 17275, 8/91.

[12] R. Guerin and L.Y.C. Lien. *Overflow Analysis for Finite Waiting Room Systems.* IEEE Transactions on Communications **38-9**, pgs. 1569-1577, 1990.

[13] G. Hebuterne and A. Gravey. *A Space Priority Queuing Mechanism for Multiplexing ATM Channels.* Computer Networks and ISDN Systems **20**, pgs. 37-43, 1990.

[14] D. Towsley, P.D. Sparaggis and C.G. Cassandras. *Stochastic Ordering Properties and Optimal Routing Control for a Class of Finite Capacity Queueing Systems.* Proceedings of the IEEE Conference on Decision and Control, 1990 **Vol 2**, pgs. 658-663.

[15] C.M. Harris and W.G. Marchal. *Computing Optimal Server-Vacation Policies*. IIE Transactions, **21-3**, pgs. 258-265, 1989.

[16] O. Kella. *Optimal Control of the Vacation Scheme in an M/G/1 Queue*. Operations Research **38-4**, pgs 724-728, 1990

[17] M.F. Neuts. *A Queueing Model for a Storage Buffer in Which the Arrival Rate is Controlled by a Switch with a Random Delay*. Performance Evaluation **5**, pgs. 243-256, 1985.

[18] H.R. van As. *Dynamic Behaviour of Network Access Congestion Control Mechanisms*. IBM Zurich Research Laboratory 8803, 1986.

[19] S. Li. *Overload Control in a Finite Message Storage Buffer*. Center for Telecommunications Research, Columbia University, 1988.

[20] W.K. Grassmann, M.I. Taksar, D.P. Heyman. *Regenerative Analysis and Steady State Distributions for Markov Chains*. Operations Research **33-5**, pgs. 1107-1116, 1985.

[21] T. Altiok and S. Stidham Jr. *The Allocation of Interstage Buffer Capacities in Production Lines*. IIE Transactions, **15-4**, pgs. 292-308, 1983.

[22] J. Baker and I. Rubin. *Polling with a General-Service Order Table.* IEEE Transactions on Communications vol **COM-35**, pgs. 283-288, 1987.

[23] Y. Wardi, M.A. Zazanis and M. Luo. *Consistency of Perturbation Analysis for a Queue with Finite Buffer Space and Loss Policy*. Journal of Optimization Theory and Applications **68-1**, pgs. 181-202, 1991.

[24] H.R. van As. *Congestion Control in Packet Switching Networks by a Dynamic Foreground-Background Storage Strategy*. Performance of Computer-Communication Systems, pgs. 433-448, 1984.

[25] A.W. Berger. *Performance Analysis of a Rate-Control Throttle where Tokens and Jobs Queue.* IEEE Journal on Selected Areas in Communications **9-2**, pgs 165-170, 1991.

[26] H. Saito. *Queueing Analysis of Cell Loss Probability Control in ATM Networks.* Queueing, Performance and Control in ATM, pgs. 19-24, 1991.

[27] J. Ye and S.Q. Li. *Analysis of Multimedia Traffic Queues with Finite Buffer and Overload Control - Part I: QBD-Folding Algorithm.* Proceedings of IEEE INFOCOM'91, Bal Harbour, Florida, pgs. 1464-1474, 1991.

[28] J. Ye and S.Q. Li. *Analysis of Multimedia Traffic Queues with Finite Buffer and Overload Control - Part II: Applications.* Proceedings of IEEE INFOCOM'91, Bal Harbour, Florida, pgs. 1464-1474, 1991.