

# Soft Real-Time Switched Ethernet: Best-Effort Packet Scheduling Algorithm, Implementation, and Feasibility Analysis

Jinggang Wang

Thesis submitted to the Faculty of the  
Virginia Polytechnic Institute and State University  
in partial fulfillment of the requirements for the degree of

Master of Science

in

Electrical Engineering

Binoy Ravindran, Chair

Peter M. Athanas

Scott F. Midkiff

September 24, 2002

Blacksburg, Virginia

Keywords: real-time Ethernet, packet scheduling, best-effort scheduling, asynchronous  
real-time distributed systems, benefit functions, soft timeliness, feasibility conditions

# Soft Real-Time Switched Ethernet: Best-Effort Packet Scheduling Algorithm, Implementation, and Feasibility Analysis

Jinggang Wang

In this thesis, we present a MAC-layer packet scheduling algorithm, called Best-effort Packet Scheduling Algorithm(BPA), for real-time switched Ethernet networks. BPA considers a message model where application messages have trans-node timeliness requirements that are specified using Jensen's benefit functions. The algorithm seeks to maximize aggregate message benefit by allowing message packets to inherit benefit functions of their parent messages and scheduling packets to maximize aggregate packet-level benefit. Since the packet scheduling problem is NP-hard, BPA heuristically computes schedules with a worst-case cost of  $O(n^2)$ , faster than the  $O(n^3)$  cost of the best known Chen and Muhlethaler's Algorithm(CMA) for the same problem. Our simulation studies show that BPA performs the same or significantly better than CMA.

We also construct a real-time switched Ethernet by prototyping an Ethernet switch using a Personal Computer(PC) and implementing BPA in the network protocol stack of the Linux kernel for packet scheduling. Our actual performance measurements of BPA using the network implementation reveal the effectiveness of the algorithm.

Finally, we derive timeliness feasibility conditions of real-time switched Ethernet systems that use the BPA algorithm. The feasibility conditions allow real-time distributed systems to be constructed using BPA, with guaranteed soft timeliness.

## Acknowledgements

First of all, I would like to sincerely thank my advisor, Dr. Binoy Ravindran, for his continuous hard work in guiding me and polishing so much of my papers for me. Without his help and encouragement, it would have been impossible for me to finish this thesis within one year and publish papers. I learned a lot about academic research and technical writing from him, which is a big treasure of my life.

Special thanks go to my committee members, Dr. Peter M. Athanas and Dr. Scott F. Midkiff for their support and comments on my thesis. I would like to thank Dr. Athanas for pointing me in the direction of switched Ethernet research.

I would like to thank David Bacon and Mike Minter from QNX company for their support in helping me implement BPA in QNX network device driver.

No words can express my appreciation to my family for their huge contribution to my growth, especially my mother. If she had a chance to read this thesis (in Chinese), she would have been very happy. I miss her forever.

I also would like to thank my elementary and high school teachers in China. My interest in science and engineering was ignited by them.

I wish to thank all members of the real-time system research group for their company and support. Special thanks go to Peng Li for his help in many ways and to Karthik and Lakshmi for their enthusiasm in their discussions with me.

Life, study, and research far away from hometown is not easy. I would like to greatly appreciate my friends Dr. Xingsheng Liu, Mr. Liu Xingcheng, Dr. Liu Hai, and Dr. Gao Jieying for their help and encouragement in my study and research at Virginia Tech.

Thanks also go to my college advisor Dr. Li Minhong and my college friends and colleagues at JUKO Electrical Inc. and the Industrial and Commercial Bank of China at Huizhou city for their help in enriching my life and building my confidence to pursue an advanced academic degree.

Last, but not the least important, I would like to acknowledge our funding sponsor, the U.S. Office of Naval Research, for their support of this research.

# Contents

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>Introduction</b>  | <b>1</b>  |
| 1.1      | Research on Real-Time Ethernet . . . . .                               | 2         |
| 1.2      | Motivation . . . . .   | 3         |
| 1.2.1    | Difficulty with Deadlines and Deadline-based Scheduling Algorithms . . | 5         |
| 1.2.2    | Difficulty with Hard Timeliness Optimality . . . . .                   | 6         |
| 1.2.3    | Lack of Adaptivity of Deadline Timing Constraints . . . . .            | 6         |
| 1.2.4    | Difficulty with Priority Server Algorithms . . . . .                   | 7         |
| 1.3      | Overview of the Thesis . . . . .                                       | 8         |
| 1.4      | Organization of the Thesis . . . . .                                   | 11        |
| <b>2</b> | <b>The Message and System Model</b>                                    | <b>13</b> |
| 2.1      | The Message Model . . . . .  | 13        |

|          |   |           |
|----------|---|-----------|
| 2.2      | The System Model . . . . .  | 16        |
| <b>3</b> | <b>Problem Definition and Objectives</b>                                    | <b>18</b> |
| <b>4</b> | <b>Overview of CMA</b>  | <b>21</b> |
| <b>5</b> | <b>The BPA Algorithm: Heuristics and Rationale</b>                          | <b>24</b> |
| 5.1      | Sort Packets In Decreasing Order Of Their “Return of Investments” . . . . . | 24        |
| 5.2      | Move Infeasible Packets Toward The End of The Schedule . . . . .            | 25        |
| 5.3      | Maximize Local Aggregate Benefit As Much As Possible . . . . .              | 26        |
| 5.4      | Computational Complexity of BPA . . . . .                                   | 29        |
| <b>6</b> | <b>Simulation Study I: Performance in a Single Processor Environment</b>    | <b>30</b> |
| 6.1      | Experiment Setup and Parameters . . . . .                                   | 30        |
| 6.2      | Normalized Average Performance . . . . .                                    | 33        |
| 6.3      | Performance Under Increasing Task Laxity . . . . .                          | 36        |
| 6.4      | Performance Under Heterogeneous Benefit Functions . . . . .                 | 37        |
| 6.5      | Performance Comparison with DASA under Rectangular Benefit Function . . .   | 38        |
| <b>7</b> | <b>Simulation Study II: Performance in a Switched Network Environment</b>   | <b>40</b> |
| 7.1      | Experiment Setup and Parameters . . . . .                                   | 40        |

|          |   |           |
|----------|---|-----------|
| 7.2      | Normalized Average Performance . . . . .  | 42        |
| 7.3      | Performance Under Increasing Arrival Density . . . . .  | 45        |
| 7.4      | Performance Under Heterogeneous Benefit Functions . . . . .                                   | 47        |
| 7.5      | Performance Comparison with DASA under Rectangular Benefit Function . . .                     | 48        |
| 7.6      | The Importance Factor: Performance Comparison under Rectangular Benefit<br>Function . . . . . | 49        |
| <b>8</b> | <b>Implementation of BPA</b>  | <b>53</b> |
| 8.1      | Prototype Setup . . . . .   | 54        |
| 8.2      | Implementation of Packet Scheduling Algorithms in Linux Kernel . . . . .                      | 56        |
| 8.3      | Packet Switching Inside Linux Kernel Using The Linux Bridge Program . . . .                   | 58        |
| 8.3.1    | Transporting Real-Time Attributes of Real-Time Messages . . . . .                             | 58        |
| 8.3.2    | Queuing Algorithm Implementation . . . . .  | 61        |
| <b>9</b> | <b>Test Bench Setup</b>   | <b>64</b> |
| 9.1      | Experimental Parameters for Generating Message Traffic . . . . .                              | 64        |
| 9.2      | Emulating Four End-Hosts With a Single PC . . . . .   | 66        |
| 9.3      | Generating Heavy Message Traffic . . . . .  | 67        |
| 9.4      | Benefit Functions and Benefit Assignment . . . . .  | 68        |

|  |           |
|--|-----------|
| <b>10 Performance Measurement and Analysis</b>   | <b>70</b> |
| 10.1 Infeasibility of Optimal Queuing Algorithm and CMA . . . . .                      | 71        |
| 10.2 Performance Comparison by Message Size and Maximum Benefit Value . . . . .        | 72        |
| 10.3 Performance Comparison by Benefit Function Types . . . . .                        | 76        |
| 10.4 Performance Comparison Based on Increasing Traffic Arrival Density . . . . .      | 80        |
| <br>   |           |
| <b>11 Timeliness Feasibility Conditions</b>  | <b>84</b> |
| 11.1 The Soft Real-time Packet Transmission Design Problem . . . . .                   | 85        |
| 11.1.1 Sketch of Problem Invariant: Models of SRPT . . . . .                           | 85        |
| 11.1.2 Sketch of Problem Invariant: Properties of SRPT . . . . .                       | 87        |
| 11.2 A Solution Using BPA: Construction of Timeliness Feasibility Conditions . . . . . | 88        |
| 11.2.1 Construction of $R_1(s_i, p)$ . . . . .   | 89        |
| 11.2.2 Upper bound $u_1^i(p)$ . . . . .  | 89        |
| 11.2.3 Construction of $R_2(p)$ . . . . .  | 91        |
| 11.2.4 Upper bound $u_2(p)$ . . . . .  | 92        |
| 11.2.5 Feasibility Conditions . . . . .  | 92        |
| <br>   |           |
| <b>12 Conclusions, Limitations and Future Work</b>                                     | <b>94</b> |

|          |   |            |
|----------|---|------------|
| 12.1     | Limitations . . . . .   | 95         |
| 12.2     | Future Work . . . . .   | 96         |
| <b>A</b> | <b>Simulation Results for Different Benefit Functions</b>     | <b>103</b> |
| <b>B</b> | <b>Implementation Results for Different Benefit Functions</b> | <b>106</b> |
| B.1      | <i>Soft-Rectangular</i> Benefit Function . . . . .            | 106        |
| B.2      | <i>Linear</i> Benefit Function . . . . .                      | 109        |
| B.3      | <i>Quadratic</i> Benefit Function . . . . .                   | 112        |
| <b>C</b> | <b>Vita</b>   | <b>116</b> |

# List of Figures

|     |   |    |
|-----|---|----|
| 2.1 | Hard and Soft Benefit Functions . . . . .   | 14 |
| 2.2 | Example Unimodal and Multimodal Benefit Functions . . . . .   | 15 |
| 2.3 | An Example Non-Increasing Unimodal Benefit Function . . . . .   | 15 |
| 2.4 | The Real-Time Switched Ethernet Network System Model . . . . .  | 16 |
| 5.1 | High-Level Pseudo-Code of BPA Algorithm . . . . .   | 28 |
| 6.1 | Example Benefit Functions . . . . .   | 32 |
| 6.2 | Performance of BPA and CMA With Respect to Optimal Algorithm . . . . .  | 35 |
| 6.3 | Aggregate Benefit of BPA and CMA With Respect to Optimal Algorithm<br>When Task Laxity Increases . . . . .      | 36 |
| 6.4 | Performance of BPA and CMA With Respect to Optimal Algorithm Under<br>Heterogeneous Benefit Functions . . . . . | 37 |

|      |   |    |
|------|---|----|
| 6.5  | Performance of BPA, CMA and DASA With Respect to Optimal Algorithm Under Rectangular Benefit Functions . . . . .              | 38 |
| 7.1  | Performance of BPA, CMA, and EDF With Respect to FIFO . . . . .   | 43 |
| 7.2  | Performance Under Increasing Message Arrival Density and Quadratic Benefit Functions . . . . .                                | 46 |
| 7.3  | Performance Under Increasing Message Arrival Density and Heterogeneous Benefit Functions . . . . .                            | 48 |
| 7.4  | Performance Under Increasing Message Arrival Density and Rectangular Benefit Functions . . . . .                              | 49 |
| 7.5  | Performance Under Increasing Message Arrival Density and Rectangular Benefit Functions for FIFO and CMA . . . . .             | 50 |
| 7.6  | Performance Under Increasing Message Arrival Density and Rectangular Benefit Functions for BPA and DASA . . . . .             | 51 |
| 8.1  | The BPA Prototype . . . . .   | 55 |
| 8.2  | Real-Time Parameters Within IP Header . . . . .   | 60 |
| 10.1 | Average Aggregate Benefit With Respect to FIFO Under 5 Traffic Types for Rectangular and Soft-Rectangular Functions . . . . . | 77 |

|   |     |
|---|-----|
| 10.2 Average Aggregate Benefit With Respect to FIFO Under 5 Traffic Types for<br>Linear and Quadratic Functions . . . . . | 77  |
| 10.3 Average Deadline Miss Ratio Under 5 Traffic Types for Rectangular and Soft-<br>Rectangular Functions . . . . .       | 79  |
| 10.4 Average Deadline Miss Ratio Under 5 Traffic Types for Linear and Quadratic<br>Functions . . . . .                    | 79  |
| 10.5 Performance Under Traffic Type T0 for Rectangular Benefit Function . . . . .   | 80  |
| 10.6 Performance Under Traffic Type T1 for Rectangular Benefit Function . . . . .   | 81  |
| 10.7 Performance Under Traffic Type T2 for Rectangular Benefit Function . . . . .   | 81  |
| 10.8 Performance Under Traffic Type T3 for Rectangular Benefit Function . . . . .   | 82  |
| 10.9 Performance Under Traffic Type T4 for Rectangular Benefit Function . . . . .   | 83  |
| A.1 Performance Under Increasing Message Arrival Density and <i>Composite</i> Benefit<br>Functions . . . . .              | 103 |
| A.2 Performance Under Increasing Message Arrival Density and <i>Exponential</i> Ben-<br>efit Functions . . . . .          | 104 |
| A.3 Performance Under Increasing Message Arrival Density and <i>Linear</i> Benefit<br>Functions . . . . .                 | 104 |

|      |  |     |
|------|--|-----|
| A.4  | Performance Under Increasing Message Arrival Density and <i>Rectangular</i> Benefit Functions . . . . .      | 105 |
| A.5  | Performance Under Increasing Message Arrival Density and <i>Soft-Rectangular</i> Benefit Functions . . . . . | 105 |
| B.1  | Performance Under Traffic Type T0 . . . . .  | 106 |
| B.2  | Performance Under Traffic Type T1 . . . . .  | 107 |
| B.3  | Performance Under Traffic Type T2 . . . . .  | 107 |
| B.4  | Performance Under Traffic Type T3 . . . . .  | 108 |
| B.5  | Performance Under Traffic Type T4 . . . . .  | 108 |
| B.6  | Performance Under Traffic Type T0 . . . . .  | 109 |
| B.7  | Performance Under Traffic Type T1 . . . . .  | 109 |
| B.8  | Performance Under Traffic Type T2 . . . . .  | 110 |
| B.9  | Performance Under Traffic Type T3 . . . . .  | 110 |
| B.10 | Performance Under Traffic Type T4 . . . . .  | 111 |
| B.11 | Performance Under Traffic Type T0 . . . . .  | 112 |
| B.12 | Performance Under Traffic Type T1 . . . . .  | 112 |
| B.13 | Performance Under Traffic Type T2 . . . . .  | 113 |

|  |     |
|--|-----|
| B.14 Performance Under Traffic Type T3 . . . . . | 113 |
| B.15 Performance Under Traffic Type T4 . . . . . | 114 |

# List of Tables

|     |   |    |
|-----|---|----|
| 4.1 | CMA's Precedence Matrix . . . . .   | 22 |
| 6.1 | Experimental Parameters . . . . .   | 31 |
| 6.2 | Example 9-Task Set . . . . .  | 33 |
| 6.3 | Performance of BPA and CMA for Two Task Sets With Respect to Optimal<br>Algorithm . . . . . | 34 |
| 7.1 | Experimental Parameters . . . . .   | 41 |
| 7.2 | Performance of BPA, CMA, and EDF With Respect to FIFO . . . . .                             | 44 |
| 8.1 | Round Trip Time Comparison for Different Network Configurations . . . . .                   | 58 |
| 8.2 | Implementation of Queuing (or Scheduling) Algorithms . . . . .                              | 62 |
| 9.1 | Expressions for Real-Time Message Attributes . . . . .                                      | 65 |
| 9.2 | IP Address Configuration for Senders and the Receiver . . . . .                             | 66 |

|   |    |
|---|----|
| 10.1 T0: Big Message Size and Normal Distribution Benefit . . . . .   | 73 |
| 10.2 T1: Small Message Size and Normal Distribution Benefit . . . . . | 73 |
| 10.3 T2: Small Message Size and $BA_L$ Maximum Benefit . . . . .      | 75 |
| 10.4 T3: Small Message Size and $BA_H$ Maximum Benefit . . . . .      | 75 |
| 10.5 T4: Big Message Size and $BA_H$ Maximum Benefit . . . . .        | 76 |
| 10.6 BPA Favored Traffic Types . . . . .                              | 78 |

# Chapter 1

## Introduction

The IEEE 802.3 Ethernet standard is unsuited for real-time applications due to the random strategy employed by Ethernet's Carrier Sense Media Access/Collision Detection(CSMA/CD) Media Access Control(MAC) protocol for resolving access contention to the network media. This is a fundamental limitation of Ethernet for real-time applications as it is conceptually equivalent to random scheduling of resources — a highly undesirable feature for satisfying real-time objectives. However, Ethernet is attractive for real-time applications due to its wide availability, low cost, and high performance such as that offered by the emerging 10 Gigabit Ethernet standard [1, 2]. This has motivated research on circumventing the non-determinism of CSMA/CD so that Ethernet networks can be used for building real-time distributed applications.

## 1.1 Research on Real-Time Ethernet

Earlier research on real-time Ethernet focused on shared Ethernets that use hubs or buses.

In shared Ethernets, two schemes are generally used for supporting real-time traffic [3]: (1) *collision-free* schemes and (2) schemes that *allow collisions*.

In collision-free schemes, end-hosts are only allowed to access the network using a pre-defined access control strategy, thereby completely avoiding collisions. The access control strategies employed can be broadly classified into (a) token-based access control and (b) end-host traffic-based access control. Examples of collision-free Ethernets that employ token-based access control strategy include RETHER [4], Table-Driven Proportional Access (TDPA)-based Real-Time Ethernet [5], MpRAM protocol [6], and Center Controller-based Ethernet [7]. Collision-free Ethernets that use end-host traffic-based access control strategy are presented in TEMPRA [8], Beck IPC [6], SIXNET [9], [10], and [11]. Most of these works provide “per-connection” delay and bandwidth guarantees. Thus, an application requests a real-time connection that has a given delay, i.e., a deadline, and a bandwidth requirement. If the request can be honored, the scheme admits the request and continuously honors the request as long as the requested bandwidth is not exceeded by the application.

In schemes that allow collisions, end-hosts are allowed to arbitrarily access the network, thereby causing collisions. However, once collisions occur, they are deterministically resolved. Examples include the recently developed CSMA/DDCR(Deadline-Driven Collision Resolution) MAC protocol [12] and previously developed Virtual Time Protocols [13, 14].

The low throughput offered by real-time shared Ethernets motivated research on real-time switched Ethernets, where end-hosts are interconnected through switches using full-duplex Ethernet segments. Though switches are traditionally used and intended for interconnecting multiple Ethernet segments that are internally built using shared Ethernets, there is an increasing trend toward using switches for building single-segment networks. Thus, unlike in shared Ethernets, where a hub simply broadcasts every message that it receives and causes collisions, a switch determines the destination host of incoming messages and schedules them on the appropriate outgoing network segment. Thus, the switch avoids collisions by “directing” the network traffic.

The real-time switched Ethernet network approach is presented in EtheReal [15], SIXNET Industrial Ethernet Switch [9], [16], and [17].<sup>1</sup> Since a switched Ethernet is also a packet-switching network (PSN), most of the real-time PSN research [19] are also relevant for real-time switched Ethernets. Similar to real-time shared Ethernet efforts, the real-time switched Ethernet and PSN efforts also provide per-connection delay and bandwidth guarantees.

## 1.2 Motivation

Thus, the existing real-time Ethernet networks can be used for building real-time applications where timing constraints are delays or deadlines. However, deadline timing constraints have

---

<sup>1</sup>The switch-based approach is gaining support in the real-time industry, with several companies developing Ethernet switches that are now in widespread usage [9, 18].

the well known drawback that they implicitly or explicitly indicate that the deadlines are “hard” [20]. Thus, completing a deadline-constrained activity before its deadline implies the accrual of some “benefit” and that benefit remains the same if the activity were to complete anytime before the deadline. Furthermore, completing the activity after the deadline implies a timing failure, i.e., the accrual of zero benefit or, sometimes, a negative benefit.

With deadlines, it, therefore, becomes difficult to express timing constraints that are not hard, but “soft” in the sense that completing the time-constrained activity at anytime will result in some benefit and that benefit *varies* with the completion time of the activity. Furthermore, with deadlines, it becomes difficult to specify a collective timeliness optimality criterion that is not hard, but soft in the sense that completing as many soft time-constrained activities as possible at their optimal completion times—completion times that will lead to maximal benefit—is what is desirable.

Jensen’s benefit functions [20] precisely address the specification of soft time constraints. Furthermore, Jensen’s benefit accrual predicates [20] allow the specification of soft collective timeliness optimality criteria.

Benefit functions and benefit accrual predicates assume great significance in the context of “asynchronous” real-time distributed systems that are emerging in many domains including defense, telecommunication, and industrial automation for strategic mission management [20, 21, 22]. Such systems are fundamentally distinguished by the significant run-time uncertainties that are inherent in their application environment and system resource states. Consequently, it is difficult to postulate upper bounds on application workloads for such

systems that will always be respected at run-time. Thus, they violate the deterministic foundations of hard real-time theory that ensures that all deadlines are always satisfied under deterministic postulations of application workloads.

We believe that the timing constraints and the collective timeliness optimality criteria of asynchronous real-time distributed systems are best described by soft timing constraints using benefit functions and soft optimality criteria using the benefit accrual predicates, respectively. We describe the rationale of our belief in the subsections that follow.

### **1.2.1 Difficulty with Deadlines and Deadline-based Scheduling Algorithms**

Many timing constraints in asynchronous real-time distributed systems are soft in the aforementioned sense [20, 23, 24]. With deadline-based scheduling algorithms, such soft timing constraints must be converted to deadlines. This will cause the deadline-based algorithms to seek the completion of the soft activities at anytime before the deadlines, violating their *non-uniform* benefit semantics. i.e., the non-uniform benefit that is accrued as a function of completion times before the deadlines.

On the contrary, benefit functions allow the semantics of soft timing constraints to be precisely specified. This allows scheduling algorithms that use such specifications to seek the completion of soft activities that are precisely consistent with their timing constraint specifications i.e., seek to complete soft activities at times that will yield their optimal benefit.

### 1.2.2 Difficulty with Hard Timeliness Optimality

The difficulty in postulating upper bounds on application workloads for asynchronous real-time distributed systems that will never be violated makes the hard timeliness optimality criterion difficult to achieve. Overcoming this difficulty would require strong workload presumptions, resulting in an over-supply of resources that will be poorly utilized, and thus causing low cost-effectiveness.

On the other hand, the benefit accrual model allows the specification of timeliness optimality criterion that facilitate application timeliness to be optimized in a cost-effective way. For example, a highly desirable criterion would be to complete all activities at their optimal completion times if situation permits, such as during under-load conditions when resources are sufficient with respect to the current application workload. Furthermore, during overload conditions, the desirable criterion would be to complete as many activities as possible at their optimal completion times, and less at their suboptimal completion times, and thereby facilitate graceful degradation of application timeliness. We believe that such an approach leads to greater cost-effectiveness.

### 1.2.3 Lack of Adaptivity of Deadline Timing Constraints

Given that we cannot predict the future, it is possible that the actual operating conditions of a system is “stronger” than what was assumed when the system was designed. Thus, it is possible that the models that are used to design a real-time system solution can be violated by

the “adversary” when the system is in operation. When such conditions occur, scheduling is complicated with deadlines and collective timeliness criteria that are specified with deadlines in that they do not indicate what objectives must be sought (during such situations). Thus, there is a *lack of inherent adaptivity* in deadline timing constraint specifications.

On the other hand, benefit accrual predicates allow the specification of timeliness optimality criterion that facilitate application timeliness to be optimized in an application-specific and inherently adaptive way. For example, a highly desirable criterion would be to complete all activities at their optimal completion times if situation permits, such as during conditions when the “adversary” behaves as reasoned and permitted by the design assumptions. Furthermore, when the “adversary” violates design assumptions, a desirable criterion would be to complete as many activities as possible at their optimal completion times, less at their suboptimal completion times, and thereby facilitate graceful degradation of application timeliness. Such timeliness optimality criteria can be specified with benefit accrual predicates.

#### 1.2.4 Difficulty with Priority Server Algorithms

Many asynchronous real-time distributed applications have a mixture of activities that have hard deadlines, soft timing constraints (in the aforementioned sense), and no timing constraints. Examples include [23]. The majority of the existing approaches that have been developed to solve such problems, such as the fixed priority server algorithms [25] and dynamic priority server algorithms [26], schedule the soft time-constrained actions and the

non-time-constrained-actions in a manner that will not interfere those with hard deadlines. The collective timeliness optimality criterion of such scheduling schemes is twofold: (1) always satisfy the hard deadlines and (2) minimize the average response time of all others.

The average response time minimization criterion for all non-hard actions is in fact antagonistic with the genuine soft timeliness optimality criteria desired by the soft actions. Further complication arises when soft actions are *more important* than hard actions. In fact, in many asynchronous real-time distributed systems such as [23], importance is a dynamically changing attribute. However, importance is a “hard-wired” attribute—hard more important than soft—of the priority server algorithms. Thus, such algorithms will operate contrary to the desired objectives.

With benefit accrual predicates, “global,” collective timeliness optimality criterion that encompasses hard collective timeliness optimality criterion (for hard actions) and soft collective timeliness optimality criterion (for soft actions) can be specified. Further, algorithms can be designed that satisfy such global, collective timeliness optimality criteria. Furthermore, algorithms can be designed that explicitly treat urgency and importance as orthogonal properties.

### 1.3 Overview of the Thesis

In this thesis, we advance the real-time Ethernet switch-based technology by presenting a packet scheduling algorithm called Best-Effort Packet Scheduling Algorithm (or BPA) for

constructing asynchronous real-time distributed systems.

BPA considers a message model where application messages have trans-node timeliness requirements that are specified using Jensen’s benefit functions [20]. Thus, delivering a trans-node message at its destination host will yield a benefit that is specified by the message benefit function. Furthermore, the algorithm considers a single-segment switched Ethernet network where end-hosts are interconnected using a switch through full-duplex Ethernet segments.

Given such a timeliness model and system model, the objective of BPA is to maximize the aggregate message-level benefit, i.e., a soft timeliness optimality criterion. Thus, BPA is a “best-effort” algorithm in the sense that it seeks to provide the best benefit to the application, where the best benefit that the application can accrue is application-specified using benefit functions.

BPA schedules outgoing packets of messages from source end-hosts to the switch and from the switch to destination hosts to maximize aggregate message-level benefit. The algorithm thus solves a packet scheduling problem, which is equivalent to the non-preemptive task scheduling problem solved in [27] and shown to be NP-hard. In [27], Chen and Muhlethaler present a heuristic algorithm for this problem with a worst-case cost of  $O(n^3)$ . For convenience, we call this Chen and Muhlethaler’s Algorithm (or CMA). In [27], CMA is experimentally shown to perform very well with respect to the best optimal algorithm for this problem.

BPA heuristically solves the same problem as that solved by CMA. However, we show that the cost of BPA is only  $O(n^2)$ . Furthermore, our simulation studies show that BPA performs

the same or significantly better than CMA for a broad set of benefit functions and when message arrival density increases.

We also implement BPA and, thus, construct a soft real-time switched Ethernet network. We prototype an Ethernet switch using a Pentium PC that is equipped with multiple, multi-port Ethernet cards to interconnect end-hosts. We implement BPA in a software layer between the IP and the Ethernet device driver layers in the Linux kernel for packet scheduling at end-hosts and at the switch. Furthermore, we implement end-host socket libraries and construct an application program interface. Our actual performance measurements using the implementation and experimental comparisons reveal the strong effectiveness of the algorithm.

Finally, we conduct schedulability analysis and derive timeliness feasibility conditions of real-time switched Ethernet network systems that use the BPA algorithm. We express the conditions as non-valued benefit accrual predicates that satisfy the worst-case conditions embodied in the models of a given application problem. Quantification of the feasibility conditions will thus allow real-time distributed systems to be constructed with guaranteed soft timeliness properties.

Thus, the contribution of the thesis is threefold:

1. the BPA packet scheduling algorithm that seeks to maximize aggregate message benefit in real-time distributed systems;
2. the construction of a soft real-time switched Ethernet network using BPA; and
3. timeliness feasibility conditions for constructing real-time distributed systems using

BPA with guaranteed soft timeliness.

Besides CMA that BPA is shown to outperform in this thesis, we are not aware of any other efforts that solve the problem solved by BPA.

## 1.4 Organization of the Thesis

The rest of the thesis is organized as follows: We discuss the message model and the system model that we are considering in this work in Chapter 2. We formally describe the problem that we are addressing and state our objectives in Chapter 3. In Chapter 4, we present an overview of the CMA algorithm, for completeness. We describe the BPA algorithm, the heuristics employed by the algorithm, and the rationale behind the heuristics in Chapter 5.

In Chapter 6, we discuss BPA's performance in a single processor environment that was measured through simulation studies. We describe BPA's performance in a switched network environment (measured through simulation studies) in Chapter 7.

We discuss the implementation of BPA and the construction of a real-time switched Ethernet in Chapter 8. Chapter 9 discusses the test bed setup of the switched Ethernet implementation for measuring actual performance. In Chapter 10, we describe the performance measurements of BPA that were obtained using the implementation.

In Chapter 11, we derive timeliness feasibility conditions for constructing real-time distributed systems using BPA.

Finally, the thesis concludes with a summary of the work, its contributions, and identify future work in Chapter 12.

# Chapter 2

## The Message and System Model

### 2.1 The Message Model

We denote the set of messages of the application by the set  $M = \{M_1, M_2, \dots, M_n\}$ . Each message has an end-to-end timing requirement that is expressed using Jensen's benefit function [20]. We denote the benefit function of a message  $M_i$  as  $B_i(t)$ . Thus, the arrival of a message  $M_i$  at its destination host (since the release of the message at the source host) at a time  $t$  will yield a benefit  $B_i(t)$ .

Example benefit functions are shown in Figure 2.1. The classical "hard" deadline requirements are "step" benefit functions such as the one shown in Figure 2.1(a), where the arrival of a message at anytime before its deadline will result in uniform benefit; the arrival of the message after the deadline will result in zero benefit. All other benefit functions express soft timing

requirements [20].

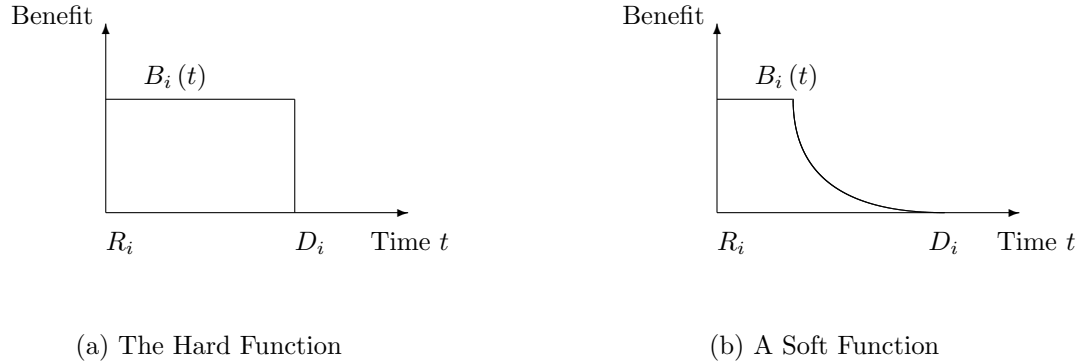
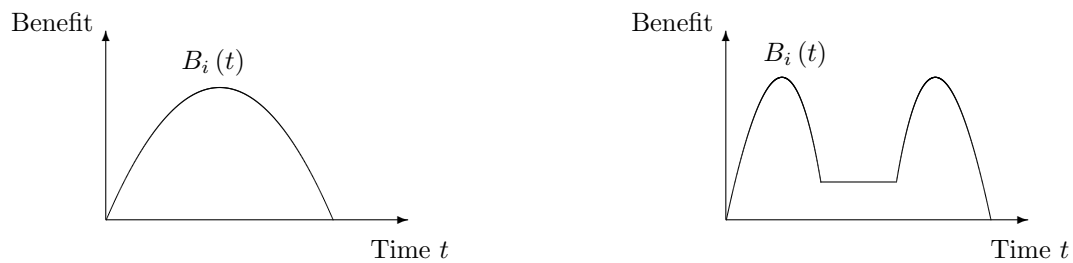


Figure 2.1: Hard and Soft Benefit Functions

Though benefit functions can take arbitrary shapes, in this thesis, we restrict our focus to *unimodal* functions that are *non-increasing*. Unimodal functions are those benefit functions for which any decrease in benefit cannot be followed by an increase in benefit [20]. Benefit functions which are not unimodal are called *multimodal*. Example unimodal and multimodal functions are shown in Figure 2.2.

Unimodal functions that are non-increasing are simply those benefit functions for which benefit never increases when time advances. Figure 2.3 shows an example. The class of non-increasing unimodal functions allow the specification of a broad range of timing constraints, including hard constraints and most soft constraints.

We denote the release time and deadline of a message  $M_i$  as  $R_i$  and  $D_i$ , respectively. We assume that benefit functions take positive values and  $B_i(t) = 0, \forall t \notin [R_i, D_i], i = 1, \dots, n$ .



(a) A Unimodal Function

(b) A Multimodal Function

Figure 2.2: Example Unimodal and Multimodal Benefit Functions

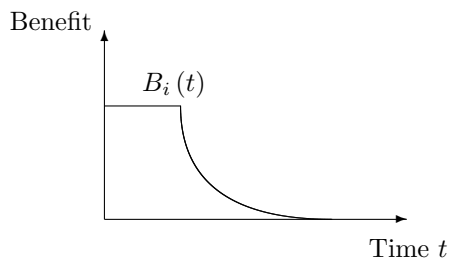


Figure 2.3: An Example Non-Increasing Unimodal Benefit Function

## 2.2 The System Model

We consider a single-segment switched Ethernet network, where hosts are interconnected through a centralized switch as our target platform (see Figure 2.4). Each host is connected to the switch using a full-duplex Ethernet segment (IEEE 802.3) and to a port at the switch that is dedicated for the host. Thus, the link between each host and the switch is a dedicated link for simultaneous two-way communication between the host and the switch.

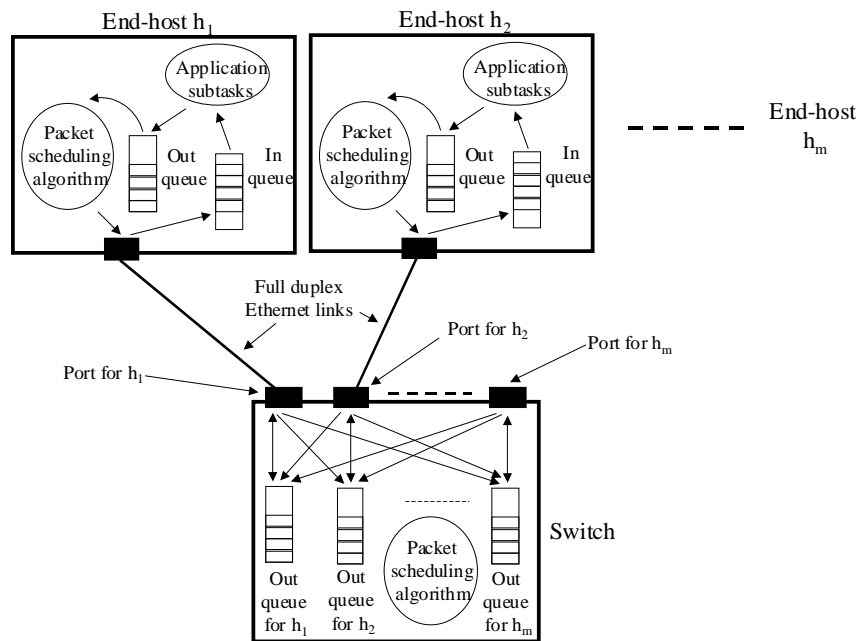


Figure 2.4: The Real-Time Switched Ethernet Network System Model

A message that is sent from an application process is first packetized on the host of the process. The message packets are then deposited into the MAC-layer of the host. When packets arrive at the MAC-layer of the host, they are queued in the outgoing packet queue of the host. Whenever the network segment from the host to the switch becomes free for

transmission, it triggers the packet scheduling algorithm at the end-host, which then executes and schedules a packet from the packet queue for transmission.

The switch is assumed to maintain a list of packet ready-queues, one queue per host. Each queue stores the packets that are destined for the corresponding host. When packets arrive at the switch, they are queued in the outgoing packet queue at the switch for the corresponding destination host. Whenever the network segment from the switch to an end-host becomes free for transmission, it triggers the packet scheduling algorithm at the switch, which then executes and schedules a packet from the packet queue of the destination host for transmission (on the corresponding output port).

We denote the set of packets of a message  $M_i$  as  $P_i = \{p_1^i, p_2^i, \dots, p_m^i\}$ . For convenience, we denote the set of all message packets of the application as the set  $P = \bigcup_{i=1}^n P_i$ . From now on, for convenience, we will drop the superscript  $i$  of a packet  $p_j^i$ , unless  $i$  is needed. Thus, we will simply denote the set of all packets as  $P = \{p_1, p_2, \dots, p_{m \times n}\}$ .

The bit length of a packet  $p_i$  is denoted as  $b_i$ . Thus, the transmission latency of a packet  $p_i$  is given by  $l_i = b_i/\psi$ , where  $\psi$  denotes the nominal throughput of the underlying Ethernet network segment (e.g.,  $10^9$  bits/s for Gigabit Ethernet).

We assume that the clocks of the end-hosts are synchronized using a protocol such as [28].

# Chapter 3

## Problem Definition and Objectives

Given the timeliness model and the system model described in Chapter 2, our objective is to maximize the aggregate benefit accrued by the arrival of all application messages at their destination hosts. Thus, the problem that we are addressing in this thesis can be described as *Maximize*  $\sum_{k=1}^n B_k(t_k)$ , where  $t_k$  is the time at which message  $M_k$  arrives at its destination host.

BPA is a packet scheduling algorithm. Thus, packets constitute the “input” to the algorithm. Therefore, if packets have benefit functions, that will enable the algorithm to reason about scheduling packets such that it will maximize the aggregate message-level benefit. This requires us to translate benefit functions of messages into benefit functions for packets.

Since the packets of a message do not have any precedence relations, the packets can be transmitted in any order from the source host of the message. Furthermore, the benefit of a

message is accrued only when all packets of the message arrive at the destination host and are reassembled. Thus, the packets of a message can simply inherit the benefit function of its parent message.

Thus, BPA reasons that by scheduling packets at outgoing queues of end-hosts and at the switch such that the aggregate packet benefit is maximized, the algorithm can maximize the aggregate message benefit.

We now formalize the objective of BPA as follows:

Let  $\mathcal{A} \subseteq P$  denote the set of packets in the outgoing packet queue at an end-host or at the switch at a time  $t$ . (Note that the switch has one outgoing queue per host.) Let  $\alpha \subseteq (m \times n)$  denote the number of packets in the set  $\mathcal{A}$ . Let  $\mathcal{S}(\mathcal{A})$  denote all possible sequences of packets of the set  $\mathcal{A}$ , and let  $\sigma \in \mathcal{S}(\mathcal{A})$  denotes one of the possible packet sequence—or a packet schedule—of the packets in  $\mathcal{A}$ . Let  $\sigma(i)$  denotes the packet occupying the  $i^{\text{th}}$  position in the schedule  $\sigma$ . Then, the objective of BPA is to:

$$\text{Maximize}_{\sigma \in \mathcal{S}(\mathcal{A})} B(\sigma) = \sum_{k=1}^{\alpha} B_{\sigma(k)}(t + t_k), \text{ where } t_k = \sum_{i=1}^k l_{\sigma(i)}$$

Thus, the objective of BPA is to determine a packet schedule that will maximize the aggregate packet benefit in terms of the sum of the individual benefits of each packet that is accrued when the packet arrives at its destination host.

This optimization problem is equivalent to the non-preemptive task scheduling problem addressed by Chen and Muhlethaler in [27]. In [27], Chen and Muhlethaler show that their task scheduling problem is NP-hard by establishing the equivalence of their problem to the

scheduling problem addressed in [29]. Thus, the problem that we are addressing in this thesis is NP-hard.

In [27], Chen and Muhlethaler present a heuristic algorithm to solve this problem. As discussed in Chapter 1, we refer to this algorithm as CMA, for convenience. Given,  $n$  tasks (i.e.,  $\alpha = n$  packets in our case), the complexity of CMA is shown to be  $O(n^3)$  in [27]. Through simulation studies, Chen and Muhlethaler show that CMA yields an aggregate benefit that is generally close to the maximum possible—or the optimal—aggregate benefit. To determine the optimal aggregate benefit, they use Held and Karp’s dynamic programming solution for the same problem presented in [30]. Held and Karp’s solution has an exponential complexity of  $O(n2^n)$ .

We believe that  $O(n^3)$  is too high for a scheduling algorithm, especially for an on-line algorithm such as a packet scheduling algorithm. This is because, higher the cost of the algorithm, higher will be the scheduling overhead. Thus, when used in the switch, the algorithm will “slow down” the scheduling of packets at the switch and thereby reduce the utilization of the network segments. Furthermore, a faster scheduling algorithm will require less buffer space for storing the outgoing packet queues. If the scheduler is slow, packets will quickly queue-up over a short period of time.

Thus, in designing BPA, our objective is twofold: (1) compute scheduling decisions faster than CMA’s  $O(n^3)$  time, and (2) compute scheduling decisions that will yield an aggregate packet benefit that is as close as possible to that of CMA, if not better.

# Chapter 4

## Overview of CMA

The CMA algorithm directly exploits the precedence-relation property that Chen and Muhlethaler presents in [27].

Chen and Muhlethaler uses the precedence-relation property to define a precedence relation between tasks. A task  $i$  is said to *precede* a task  $j$  at a time  $t$ , denoted as  $i \prec_t j$ , if  $\Delta_{i,j}(t) \geq 0$ . Now, if the precedence relationship between all task pairs can be determined, the task that gains the maximum number of precedences is a very good candidate to be scheduled at time  $t$ . Thus, Chen and Muhlethaler reason that this maximum-precedence task can be determined at each scheduling instant and thereby a good scheduling decision can be made. This intuition is exploited by the CMA algorithm.

CMA constructs a *precedence matrix* at each scheduling instant to store the precedence relations. Each row and column of this matrix represents a task. Furthermore, each entry of

Table 4.1: CMA's Precedence Matrix

|     | 1 | 2         | ... | $i$       | ... | $n$       |
|-----|---|-----------|-----|-----------|-----|-----------|
| 1   | – | $I(1, 2)$ | ... | $I(1, i)$ | ... | $I(1, n)$ |
| 2   | – | –         | ... | $I(2, i)$ | ... | $I(2, n)$ |
| ... | – | –         | ... | ...       | ... | ...       |
| $i$ | – | –         | ... |           | ... | $I(i, n)$ |
| ... | – | –         | ... | ...       | ... | ...       |
| $n$ | – | –         | –   | –         | –   | –         |

the matrix contains a boolean value that indicates whether there exists a precedence relation between the tasks represented on the row and on the corresponding column. Thus, given a precedence matrix  $I(1, 2, \dots, n)(1, 2, \dots, n)$  that represents the precedence relationship of  $n$  tasks at a scheduling instant  $t$ ,  $I(i, j)$  is *true*, if and only if  $i \prec_t j$ ; otherwise  $I(i, j)$  is *false*.

Table 4.1 illustrates this concept.

Once the precedence matrix is constructed, CMA computes the schedule by examining the rows of the matrix. For each row of the matrix, the algorithm counts the number of true values that are present in the columns of the row. The number of true values in a row  $i$  indicates the number of tasks over which task  $i$  has a precedence relation. Once the “true counts” are determined for rows, CMA simply selects the task that has the largest true count, inserts the task into the schedule, and marks the row of the task as “examined” in the matrix (so that subsequent examinations can ignore the row). The algorithm repeats this process until all tasks are inserted into the schedule.

Given  $n$  tasks, it would cost the algorithm  $O(n^2)$  computations to first construct the precedence matrix. This is followed by  $O(n^2)$  number of examinations to determine the task with the largest true count. The  $O(n^2)$  examinations has to be repeated for each of the  $n$  tasks. Thus, the complexity of CMA is clearly  $O(n^3)$ .

## Chapter 5

# The BPA Algorithm: Heuristics and Rationale

We discuss the heuristics employed by BPA and their rationale in the sections that follow.

### 5.1 Sort Packets In Decreasing Order Of Their “Return of Investments”

The potential benefit that can be obtained by spending a unit amount of network transmission time for a packet defines a measure of the “return of investment” for the packet. Thus, by ordering packets in the schedule in the decreasing order of their return of investments, we “greedily” collect as much “high return” packets into the schedule as early as possible.

Furthermore, since a packet included in the schedule at any instant in time is always the one with the next “highest-return” packet among the set of non-examined packets, we increase our chance of collecting as much “high return” packets into the schedule as early as possible. This will increase the likelihood of maximizing the aggregate packet benefit as packets yield greater benefit if they arrive earlier at their destinations, since all packet benefit functions that we consider are unimodal and non-increasing.

The return of investment for a packet can be determined by computing the slope of the packet benefit function. However, computing slopes of arbitrary unimodal benefit functions can be computationally expensive. Thus, we determine the return of investment for a packet as simply the ratio of the maximum possible packet benefit, specified by the packet benefit function, to the packet deadline. This is just a single division, costing  $O(1)$  time. We call this ratio, the “pseudo-slope” of a packet. The slope is “pseudo” as it does not represent the correct slope and only gives a crude measure of the slope.

## 5.2 Move Infeasible Packets Toward The End of The Schedule

Infeasible packets are packets that cannot arrive at their destination before their deadlines, no matter what. This is because the transmission time of such packets are longer than the time interval between the scheduling instant—the time at which the scheduler is triggered,

which is the arrival of a packet into the outgoing packet queue at a host or the switch—and the packet deadlines. Packets that are not infeasible are feasible packets.

By moving infeasible packets to the end of the schedule, we collect as much feasible packets to the beginning of the schedule as possible. This will increase the likelihood of maximizing the aggregate packet benefit as feasible packets yield greater benefit if they arrive earlier at their destinations since we consider only unimodal benefit functions that are non-increasing. Furthermore, infeasible packets yield zero benefit if they arrive at their destinations after their deadlines. Thus, there is no reason for transmitting them early and jeopardize the potential benefit that can be accrued from feasible packets.

### 5.3 Maximize Local Aggregate Benefit As Much As Possible

We derive the notion of local aggregate benefit from the *precedence-relation* property presented by Chen and Muhlethaler in [27]. For completeness, we summarize this here.

Consider two schedules  $\sigma_a = \langle \sigma_1, p_i, p_j, \sigma_2 \rangle$  and  $\sigma_b = \langle \sigma_1, p_j, p_i, \sigma_2 \rangle$  of a packet set  $\mathcal{A}$ , such that  $\sigma_1 \neq \emptyset$ ,  $\sigma_2 \neq \emptyset$ ,  $\sigma_1 \cup \sigma_2 = \mathcal{A} - \{p_i, p_j\}$ , and  $\sigma_1 \cap \sigma_2 = \emptyset$ . Consider a scheduling time instant  $t = \sum_{k \in \sigma_1} l_k$ , when a scheduling decision has to be made. That is,  $t$  is the instant in time after all packets in the schedule  $\sigma_1$  has been transmitted.

Now, the scheduling decision at time  $t$  can be made by determining  $\Delta_{i,j}(t)$ , where:

$$\Delta_{i,j}(t) = [B_i(t + l_i) + B_j(t + l_i + l_j)] - [B_j(t + l_j) + B_i(t + l_j + l_i)].$$

Thus, if  $\Delta_{i,j}(t) \geq 0$ , then schedule  $\sigma_a$  will yield a higher aggregate benefit than schedule  $\sigma_b$ . Otherwise,  $\sigma_b$  is better than  $\sigma_a$ . This is the precedence-relation property presented by Chen and Muhlethaler in [27].

Now, by examining adjacent packets  $p_i$  and  $p_j$  in a schedule  $\langle \sigma_1, p_i, p_j, \sigma_2 \rangle$  and ensuring that  $\Delta_{i,j}(t) \geq 0$ , we can maximize the *local* aggregate benefit of packets  $p_i$  and  $p_j$ . If all adjacent packets in the schedule have such locally maximized aggregate benefit, this will increase the likelihood of maximizing the global aggregate benefit.

The maximization of the local aggregate benefit can be done in a manner similar to that of Bubble sort. We can examine adjacent pairs of packets in the schedule, compute  $\Delta$ , and swap the packets, if the reverse order can lead to higher local aggregate benefit. Furthermore, the procedure can be repeated until no swaps are required.

Thus, BPA computes packet schedules according to the heuristics discussed in Sections 5.1, 5.2, and 5.3.

Pseudo-code of BPA at a high-level of abstraction is shown in Figure 5.1.

BPA( $\mathcal{A}, \alpha, t$ ) /\*  $\mathcal{A}$ : set of packets in out queue;  $\alpha$ : # of packets in  $\mathcal{A}$ ;  $t$ : time of sched. event \*/

1.  $\sigma = \emptyset$ ; /\* Initialize packet schedule to empty \*/
2. **For** each packet  $p_i \in \mathcal{A}$ 
  - 2.1 PseudoSlope( $p_i$ ) =  $B_i(0)/D_i$ ; /\* Max benefit is at time 0; benefit fns are unimodal \*/
3. Sort packets in  $\mathcal{A}$  in decreasing order of their pseudo-slopes; /\*  $\mathcal{A}$  is now sorted \*/
4.  $\sigma = \mathcal{A}$ ; /\* packet schedule  $\sigma$  is set equal to sorted set  $\mathcal{A}$  \*/
5. **For**  $k = 1$  to  $\alpha$ 
  - 5.1 InOrder = TRUE;
  - 5.2 **For**  $i = 1$  to  $\alpha - 1$ 
    - 5.2.1  $j = i + 1$ ; /\*  $p_j$  is the packet that follows  $p_i$  in schedule  $\sigma$  \*/
    - 5.2.2 **If** ( $t + l_i > D_i$ ) /\* Check for feasibility of packet  $p_i$  \*/
      - Move  $p_i$  to end of schedule  $\sigma$ ; /\* packet  $p_i$  is not feasible \*/
      - **Continue**; /\* Skip and continue to another iteration \*/
    - 5.2.3 **If** ( $t + l_j > D_j$ ) /\* Check for feasibility of packet  $p_j$  \*/
      - Move  $p_j$  to end of schedule  $\sigma$ ; /\* packet  $p_j$  is not feasible \*/
      - **Continue**; /\* Skip and continue to another iteration \*/
    - 5.2.4  $\Delta_{i,j}(t) = [B_i(t + l_i) + B_j(t + l_i + l_j)] - [B_j(t + l_j) + B_i(t + l_j + l_i)]$ ;
    - 5.2.5 **If** ( $\Delta_{i,j}(t) < 0$ ) /\* Out of order, so swap \*/
      - $\sigma(i) = p_j$ ;  $\sigma(j) = p_i$ ;  $t = t + l_j$ ; InOrder = FALSE;
    - 5.2.6 **Else** •  $t = t + l_i$ ;
  - 5.3 **If** (InOrder = TRUE) /\* No swaps; so all packets are inorder \*/ • 5.3.1 **Break**;
6.  $\sigma$  is the final schedule; return packet  $\sigma(1)$  as the packet selected for transmission;

---

Figure 5.1: High-Level Pseudo-Code of BPA Algorithm

## 5.4 Computational Complexity of BPA

The computational complexity of BPA depends upon the complexity of Step (5). The complexity of all other steps are dominated by this step.

The complexity of Step (5) is dominated by that of Step (5.2); all other sub-steps of Step (5) take  $O(1)$  time.

Step (5.2) can iterate a maximum of  $\alpha$  times, and, therefore, costs  $O(\alpha)$ . Step (5) can iterate a maximum of  $\alpha$  times, and thus costs  $O(\alpha^2)$ .

Given  $m$  application messages, where each message can be packetized into at most  $n$  packets,  $\alpha = O(mn)$ . Thus, the complexity of BPA is  $O(m^2n^2)$ .

To compare the complexity of BPA with that of CMA, we need to uniformly express the problem size. Thus, given  $n$  application packets, BPA has a complexity  $O(n^2)$ . Given  $n$  non-preemptable tasks, which form the input to CMA, CMA has a complexity of  $O(n^3)$  [27]. Thus, BPA is an order of magnitude faster than CMA.

# Chapter 6

## Simulation Study I: Performance in a Single Processor Environment

To study how well BPA performs with respect to CMA and with respect to the optimal algorithm, we first consider a single processor environment, where all three algorithms are used as task scheduling algorithms for scheduling non-preemptive tasks. Note that non-preemptive task scheduling is equivalent to packet scheduling on end-hosts and at the switch.

We consider a single processor environment for the simplicity of the environment.

### 6.1 Experiment Setup and Parameters

To study the performance of the algorithms in a large data space, we randomly generate all task parameters such as execution times, deadlines, and maximum benefit of the benefit

functions using probability distribution functions.

Details of the experimental parameters are shown in Table 6.1.

We determine the execution time as the maximum of 0.5 and the value of a random variable that is exponentially distributed with a mean value of 11.5. We take the maximum of the two values to avoid negative and extremely small values.

Deadlines are determined as the maximum of execution times, a “LevelKnob,” and a random value that is exponentially distributed with a mean of 2.0. We use the LevelKnob variable to control the task laxity. The task laxity increases with the increase of the LevelKnob value.

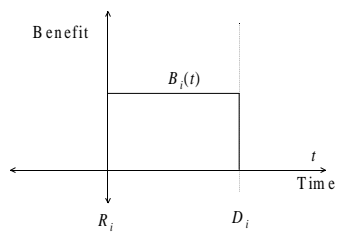
Table 6.1: Experimental Parameters

| <i>Task Properties</i>       | <i>Parameters</i>                                 |
|------------------------------|---|
| Execution Time ( $l_i$ )     | Max (0.5, $Exp(11.5)$ )                           |
| Deadline ( $D_i$ )           | Max ( $l_i + LevelKnob + Exp(2.0)$ , $Exp(3.0)$ ) |
| Maximum Benefit ( $B_i(0)$ ) | Max (0.5, $Norm(10.0, 60.0)$ )                    |

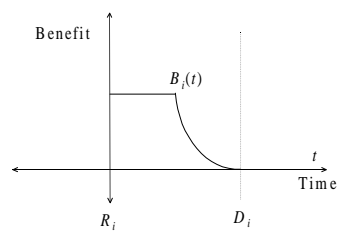
The maximum value of the benefit functions occur at time 0 and is randomly generated according to a normal distribution with a mean 10.0 and variance 60.0 (see Table 6.1).

An example task set of 9 tasks is shown in Table 6.2.

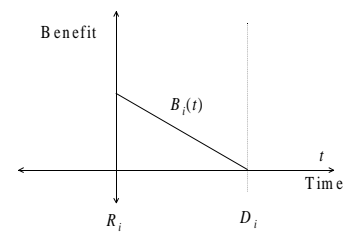
We consider six benefit functions for our experimental study. The functions are shown in Figure 6.1. The functions include rectangular, soft-rectangular, linear, exponential, quadratic, and composite.



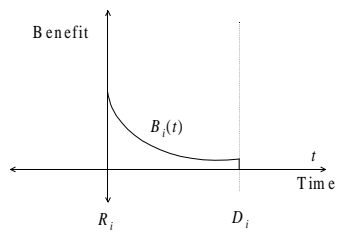
(a) Rectangular



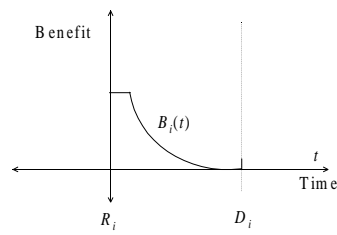
(b) Soft-Rectangular



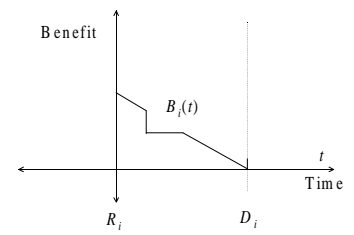
(c) Linear



(d) Exponential



(e) Quadratic



(f) Composite

Figure 6.1: Example Benefit Functions

Table 6.2: Example 9-Task Set

| <i>Task</i> | $l_i$ | $D_i$  | $B_i(0)$ |
|-------------|-------|--------|----------|
| 1           | 42.90 | 73.73  | 5.19     |
| 2           | 5.95  | 42.80  | 99.79    |
| 3           | 0.69  | 41.83  | 90.06    |
| 4           | 26.72 | 74.41  | 11.99    |
| 5           | 0.58  | 66.91  | 64.84    |
| 6           | 0.85  | 56.97  | 29.66    |
| 7           | 26.47 | 122.32 | 67.10    |
| 8           | 4.44  | 73.79  | 41.67    |
| 9           | 1.02  | 73.98  | 14.57    |

## 6.2 Normalized Average Performance

We considered two task sets—a 9 task set and a 10 task set—and measured the aggregate benefit yielded by BPA, CMA, and the optimal algorithm for the task sets. The optimal algorithm determines optimal schedules through an exhaustive search.

Since the input task workload to BPA and CMA is randomly generated, the absolute value of the aggregate benefit produced by the algorithms is not of much significance. Thus, we compute the ratio of the aggregate benefit produced by BPA and CMA to that produced by the optimal algorithm. We call this ratio, the *normalized aggregate benefit*.

We repeated our measurements for the two task sets—2500 times for the 9 task-set and 400

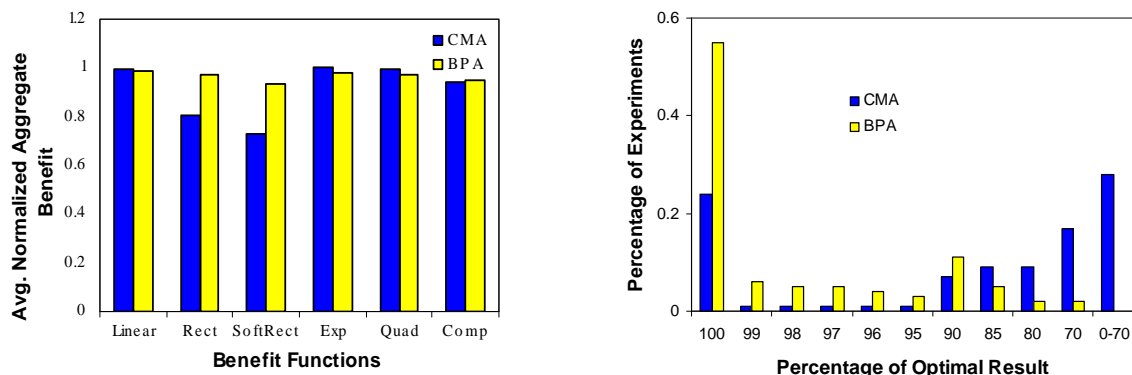
Table 6.3: Performance of BPA and CMA for Two Task Sets With Respect to Optimal Algorithm

|                 | <i>9-Task Set</i> |            |            |            | <i>10-Task Set</i> |            |            |            |
|-----------------|-------------------|------------|------------|------------|--------------------|------------|------------|------------|
|                 | <i>CMA</i>        |            | <i>BPA</i> |            | <i>CMA</i>         |            | <i>BPA</i> |            |
|                 | <i>Avg</i>        | <i>Dev</i> | <i>Avg</i> | <i>Dev</i> | <i>Avg</i>         | <i>Dev</i> | <i>Avg</i> | <i>Dev</i> |
| <i>Linear</i>   | 0.9944            | 0.0192     | 0.9873     | 0.0394     | 0.9865             | 0.0265     | 0.9802     | 0.0402     |
| <i>Rect</i>     | 0.8028            | 0.1774     | 0.97       | 0.0549     | 0.6833             | 0.1886     | 0.9457     | 0.0712     |
| <i>SoftRect</i> | 0.7264            | 0.2105     | 0.9361     | 0.1214     | 0.6724             | 0.2312     | 0.8988     | 0.1476     |
| <i>Exp</i>      | 1                 | 8E-6       | 0.9781     | 0.1079     | 1                  | 0          | 0.9589     | 0.1449     |
| <i>Quad</i>     | 0.9976            | 0.0202     | 0.9738     | 0.1146     | 0.9992             | 0.0038     | 0.9534     | 0.1473     |
| <i>Comp</i>     | 0.9411            | 0.0995     | 0.9462     | 0.0938     | 0.8982             | 0.1253     | 0.917      | 0.1108     |

times for the 10 task-set—and determined the average values of the normalized aggregate benefit. The experiments were conducted for all six benefit functions shown in Figure 6.1.

Table 6.3 shows the average values and their standard deviations for BPA and CMA for the six benefit functions for the two task sets. Figure 6.2(a) shows the average values for the 9-task set as bar charts, for convenience.

From Table 6.3 and Figure 6.2(a), we observe that the performance of BPA is very close ( $\geq 93\%$ ) to that of the optimal algorithm for all six benefit functions. Furthermore, we observe that CMA performs quite close to that of the optimal algorithm for some benefit functions such as quadratic, linear, and exponential, but performs poorly for functions such as the rectangular and soft-rectangular benefit functions.



(a) Average Normalized Aggregate Benefit

(b) Performance Distribution

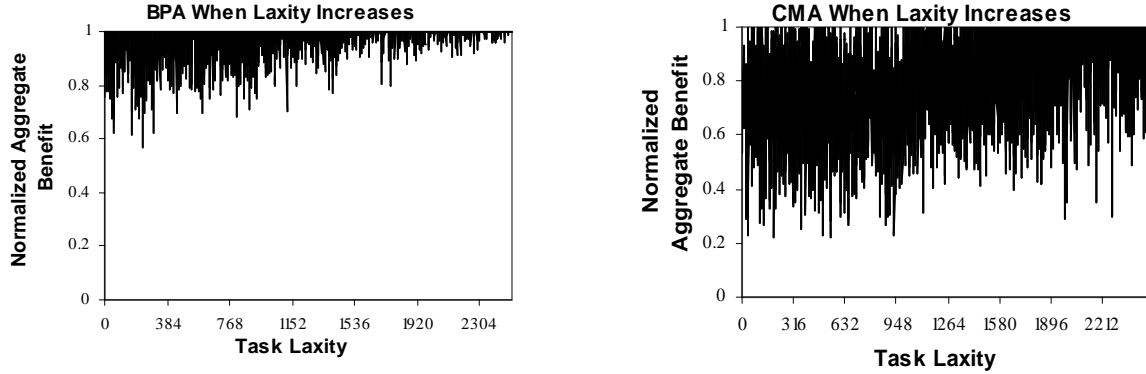
Figure 6.2: Performance of BPA and CMA With Respect to Optimal Algorithm

We also observe that the performance of BPA is close to that of CMA for linear, exponential, quadratic, and composite benefit functions. However, BPA outperforms CMA for rectangular and soft-rectangular benefit functions. We note that this observation is true for both task sets that we considered in our study.

To determine how the performance of BPA and CMA were distributed over the range of experiments that were conducted for a given benefit function, we plot different percentages of the normalized aggregate benefit produced by the algorithms in terms of the percentage of the experiments.

Figure 6.2(b) shows the distribution of the algorithms' performance for the rectangular benefit function. From the figure, we observe that more than 55% of BPA's aggregate benefit are exactly the same as the optimal value. Furthermore, no results are found to be less than 70% of the optimal value.

### 6.3 Performance Under Increasing Task Laxity



(a) BPA's Normalized Aggregate Benefit

(b) CMA's Normalized Aggregate Benefit

Figure 6.3: Aggregate Benefit of BPA and CMA With Respect to Optimal Algorithm When Task Laxity Increases

We also studied how the algorithms behave when the task laxity increases. Figure 6.3(a) and Figure 6.3(b) show the normalized aggregate benefit produced by BPA and CMA (for rectangular benefit functions) as laxity increases, respectively. The fluctuation of the values shown in the plots is due to the randomly generated input data.

We observe that the performance of both algorithms increase to that of the optimal algorithm as laxity increases. Furthermore, BPA converges faster to the optimal values than CMA.

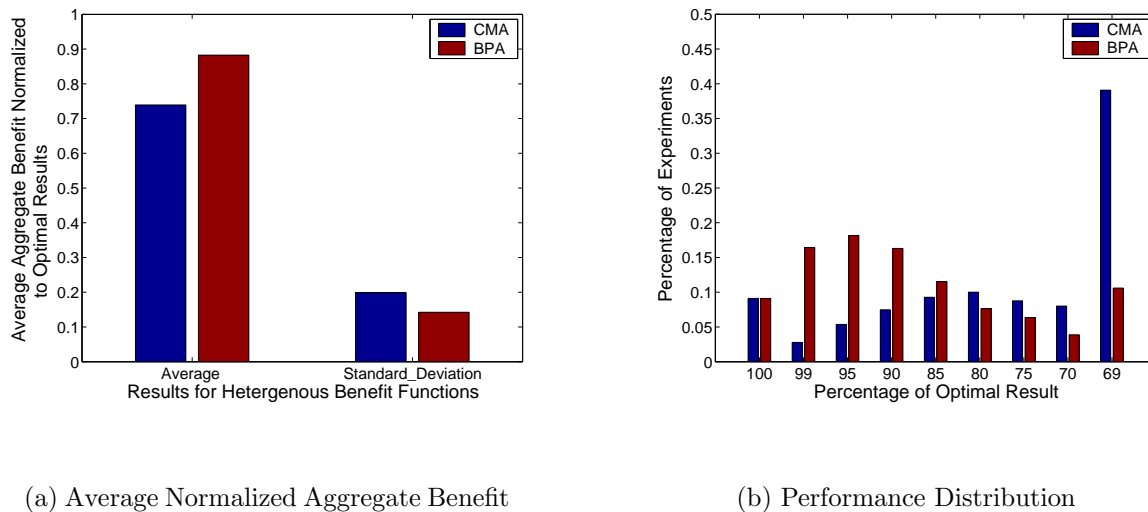


Figure 6.4: Performance of BPA and CMA With Respect to Optimal Algorithm Under Heterogeneous Benefit Functions

## 6.4 Performance Under Heterogeneous Benefit Functions

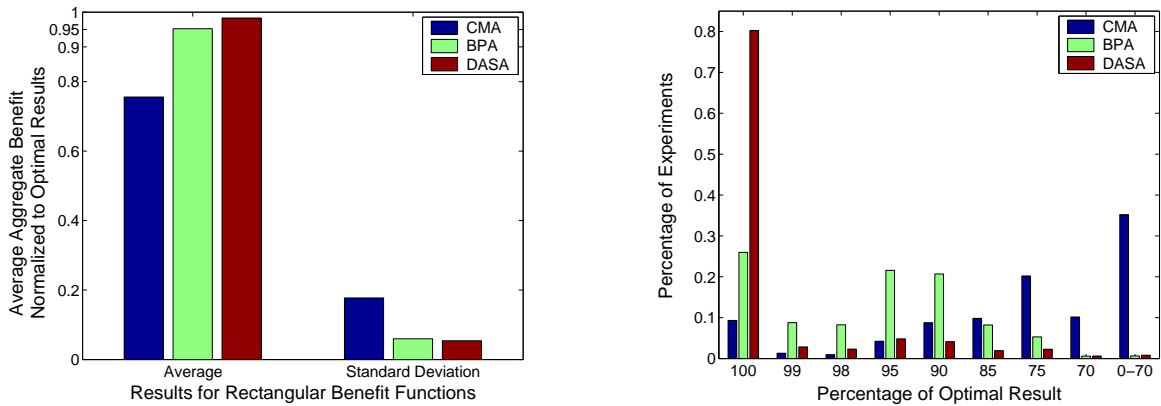
We also studied how the algorithms behave under heterogeneous benefit functions where all tasks of a processor or all packets in a single queue may have different benefit functions from any of the six categories.

Figure 6.4(a) shows the average results and standard deviations for both BPA and CMA algorithms as bar charts. We observe that the performance of BPA is very close ( $\geq 90\%$ ) to that of the optimal algorithm, which is much better than CMA. Furthermore, the standard deviation of BPA is less than that of CMA, which means that the results of BPA are more converged.

Similar to the homogeneous case, to determine how the performance of BPA and CMA were distributed over the range of all experiments, we plot different percentages of the normalized aggregate benefit produced by the algorithms in terms of the percentage of the experiments.

Figure 6.4(b) shows the distribution of the algorithms' performance. From the figure, we observe that most of BPA's results are located above 90% of the optimal value, whereas most of CMA's results are located below 90%.

## 6.5 Performance Comparison with DASA under Rectangular Benefit Function



(a) Average Normalized Aggregate Benefit

(b) Performance Distribution

Figure 6.5: Performance of BPA, CMA and DASA With Respect to Optimal Algorithm Under Rectangular Benefit Functions

Since DASA (Dependent Activity Scheduling Algorithm) [31] is one of the first best-effort

real-time scheduling algorithms to be developed, we also would like to compare BPA with DASA. DASA performs the same as EDF during underload situations and outperforms EDF during overload situations. However, one major drawback of DASA is that it works only for rectangular benefit functions. Therefore, we compare BPA with DASA for rectangular functions.

Figure 6.5(a) shows the average results and standard deviations for BPA, CMA, and DASA algorithms as bar charts. We observe that the performance of DASA is slightly better than that of BPA in terms of both the averaged result and the standard deviation. However, the comparison between BPA and CMA yields the same results as shown previously.

Figure 6.5(b) shows the distribution of the algorithms' performance. From the figure, we observe that 80% of DASA's results are exactly the same as the optimal value, which is better than BPA. But on the average, BPA performs very close to DASA as shown in Figure 6.5(a). Note that BPA works for unimodal benefit functions that are non-increasing, whereas DASA can only handle rectangular benefit functions.

# Chapter 7

## Simulation Study II: Performance in a Switched Network Environment

Our further simulation study focuses on a switched network environment.

### 7.1 Experiment Setup and Parameters

Similar to the single processor experimental setup, we randomly generate all message parameters in the network environment using probability distribution functions. The parameters include destination address of messages, message length, message deadline, maximum benefit of message benefit functions, and message inter-arrival time. Details of the experimental parameters are shown in Table 7.1.

We considered a switched network of five end-hosts, where each host contains five independent application processes that generate trans-node messages. Further, we assumed that the message traffic is uniformly distributed across all hosts. Thus, the destination addresses of messages are generated from a uniform distribution. Furthermore, the message deadlines are determined to be larger than message transmission latency and an exponentially distributed random value. Also, we use the “LevelKnob” variable to control the message arrival density. Higher the value for LevelKnob, higher is the value for message length and shorter is the value for the message inter-arrival time.

Table 7.1: Experimental Parameters

| <i>Message Properties</i>      | <i>Parameters</i>   |
|--------------------------------|---|
| Destination address            | (Int) <i>Uniform</i> (0, 4)   |
| Message Length ( $b_i$ )       | (Int) Max ( $Exp(2500 + LevelKnob \times 500)$ , 400)                                   |
| Transmission Latency ( $l_i$ ) | $b_i/\psi$  |
| Deadline ( $D_i$ )             | Max ( $l_i + 0.007$ , $Exp(l_i + 0.007)$ )  |
| Maximum Benefit ( $B_i(0)$ )   | <i>Norm</i> (30, 60)  |
| Inter-arrival Time             | Max (0.0001, $Norm(0.0015 - LevelKnob \times 0.0001, 0.081 - LevelKnob \times 0.005)$ ) |

In the simulation model, we implemented all the scheduling algorithms for packet scheduling at the MAC-layer of end-hosts and at the switch.

We conducted the simulation experiments by repeating the exact same traffic data from all processes on all end-hosts for each of the four scheduling algorithms. Furthermore, for each benefit function, we conducted experiments at different message arrival densities. A total of

480 independent experiments were performed for each benefit function.

Besides BPA and CMA, we also considered EDF and the First-In-First-Out (FIFO) algorithms for a broader comparative study. However, we excluded the optimal algorithm from this study, as we observed that the actual execution time of the optimal algorithm was extremely large even for moderately loaded situations. Thus, the optimal algorithm is not practically feasible.

We used the OmNet++ simulation toolkit [32] for our simulation study.

## 7.2 Normalized Average Performance

Since we exclude the optimal algorithm from this study, we normalize the aggregate benefit by computing the ratio of the aggregate benefit produced by the algorithms to that produced by FIFO.<sup>1</sup> We believe that it is reasonable to use FIFO as a baseline algorithm, since FIFO has the least scheduling cost of  $O(1)$  time.

Thus, in this study, the normalized aggregate benefit of an algorithm indicates how much performance gain can be obtained for the increased scheduling cost invested for the algorithm, with respect to FIFO. Therefore, higher the normalized aggregate benefit, better is the algorithm.

Figure 7.1 shows the average of the normalized aggregate benefit produced by BPA, CMA,

---

<sup>1</sup>Note that the input message traffic to the algorithms is randomly generated. Thus, the absolute value of the aggregate benefit produced by the algorithms is not of much significance.

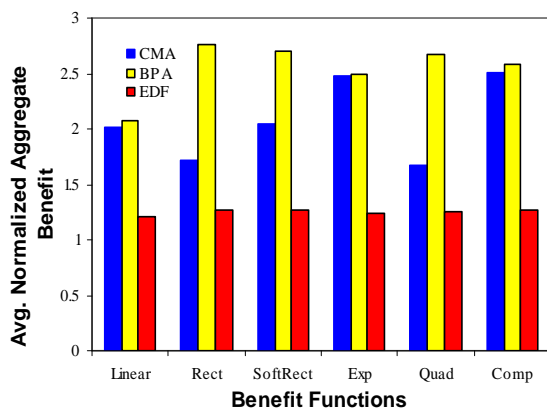


Figure 7.1: Performance of BPA, CMA, and EDF With Respect to FIFO

and EDF for all experiments that were conducted for the algorithms, for each of the six benefit functions shown in Figure 6.1.

In Table 7.2, we show the maximum, minimum, and the standard deviation of the normalized aggregate benefits, besides the average.

From Figure 7.1 and Table 7.2, we observe that BPA performs the best and EDF performs the worst, for all six benefit functions. The performance of CMA lies in between BPA and EDF. Further, we observe that BPA maintains an average normalized aggregate benefit of 2.5. Furthermore, we note that the performance of BPA and CMA is just about the same for linear, exponential, and composite benefit functions. However, BPA significantly outperforms CMA for rectangular, soft-rectangular, and quadratic benefit functions. These results are consistent with that of the single processor environment presented in Chapter 6, except for quadratic functions.

From Table 7.2, we also observe that EDF performs worse than FIFO in many cases (the minimum value of EDF's normalized aggregate benefit is less than 1), although its average

Table 7.2: Performance of BPA, CMA, and EDF With Respect to FIFO

|            |            | <i>Rect</i> | <i>SoftRect</i> | <i>Linear</i> | <i>Exp</i> | <i>Quad</i> | <i>Comp</i> |
|------------|------------|-------------|-----------------|---------------|------------|-------------|-------------|
| <i>CMA</i> | <i>Max</i> | 6.5460      | 7.2393          | 5.4552        | 10.0590    | 5.5322      | 10.4821     |
|            | <i>Min</i> | 0.9388      | 0.9882          | 1.0612        | 0.9998     | 0.9426      | 0.9990      |
|            | <i>Avg</i> | 1.7228      | 2.0483          | 2.0169        | 2.4754     | 1.6665      | 2.5129      |
|            | <i>Dev</i> | 0.9009      | 1.1423          | 0.8728        | 1.7103     | 0.8267      | 1.7569      |
| <i>BPA</i> | <i>Max</i> | 13.1441     | 10.4888         | 5.5426        | 11.2589    | 8.6893      | 9.8991      |
|            | <i>Min</i> | 1.0029      | 1.0022          | 1.0631        | 1.0004     | 1.0027      | 0.9993      |
|            | <i>Avg</i> | 2.7621      | 2.7081          | 2.0806        | 2.4924     | 2.6704      | 2.5827      |
|            | <i>Dev</i> | 2.0514      | 1.8661          | 0.9331        | 1.7284     | 1.8234      | 1.8104      |
| <i>EDF</i> | <i>Max</i> | 5.2358      | 3.8456          | 3.8664        | 3.4387     | 4.4670      | 4.7252      |
|            | <i>Min</i> | 0.5038      | 0.3060          | 0.5204        | 0.4395     | 0.3236      | 0.5102      |
|            | <i>Avg</i> | 1.2632      | 1.2758          | 1.2148        | 1.2411     | 1.2525      | 1.2696      |
|            | <i>Dev</i> | 0.5327      | 0.5491          | 0.4104        | 0.4837     | 0.5582      | 0.5715      |

performance is always better. This is because EDF works well with rectangular benefit functions that have same maximum benefits i.e., when all packets yield the same benefit for arriving at their destinations anytime before their deadlines. (The optimality of EDF [33] holds only under such conditions). But, this is not the case here for rectangular functions. When the maximum benefit of rectangular functions differ, EDF performs worse.

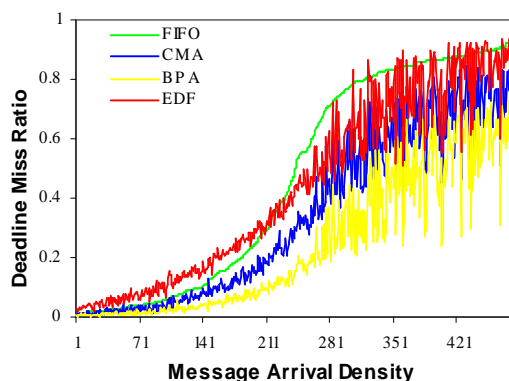
### 7.3 Performance Under Increasing Arrival Density

We were also interested to determine how BPA and CMA perform when the arrival density of messages increases. The arrival density of a message—and thus packets of the message—is simply the number of times the message arrives during an interval of time. Thus, a larger arrival density implies a larger message traffic. Our interest in this performance metric is clearly due to the asynchronous nature of the applications that we are targeting, where run-time increases in message traffic are common.

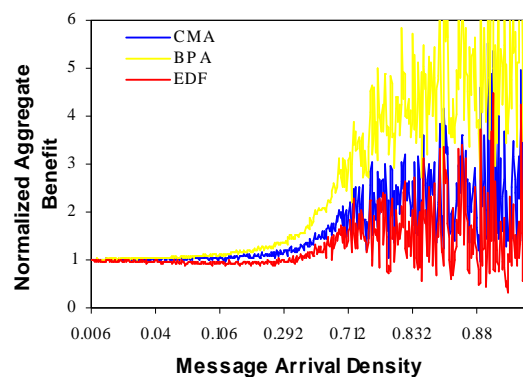
Thus, we repeated the experiments described in Section 7.2 for 16 message arrival densities that are progressively increasing.

Figure 7.2(a) shows the (end-to-end) message deadline-miss ratio—ratio of the number of deadline misses to the total number of message releases—of BPA, CMA, EDF, and FIFO under quadratic benefit functions.

From Figure 7.2(a), we observe that the deadline-miss ratio of all algorithms increases as the message arrival density increases. Further, we observe that the deadline-miss ratio of



(a) Deadline Miss Ratio



(b) Normalized Aggregate Benefit

Figure 7.2: Performance Under Increasing Message Arrival Density and Quadratic Benefit Functions

BPA is the smallest. This is followed by the ratios of CMA and EDF. FIFO has the largest deadline-miss ratio.

In Figure 7.2(b), we normalize the aggregate benefit of the algorithms by dividing their aggregate benefits with that of FIFO. Thus, FIFO has a normalized aggregate benefit of 1 in this figure. From the figure, we observe that when the message arrival density is less than 0.25, BPA, CMA, and EDF yield just about the same aggregate benefit as that of FIFO. The smaller gain in the aggregate benefit only implies that the algorithms do not lose much benefit during such small workloads.

When the message arrival density becomes larger than 0.25, we observe that the algorithms perform significantly better than FIFO. Again, we observe that BPA performs the best here, EDF the worst, and CMA in between.

We observed similar consistent results for all other benefit functions. These are shown in

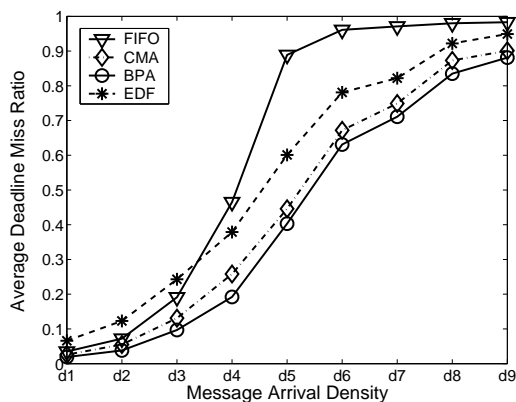
Appendix A.

## 7.4 Performance Under Heterogeneous Benefit Functions

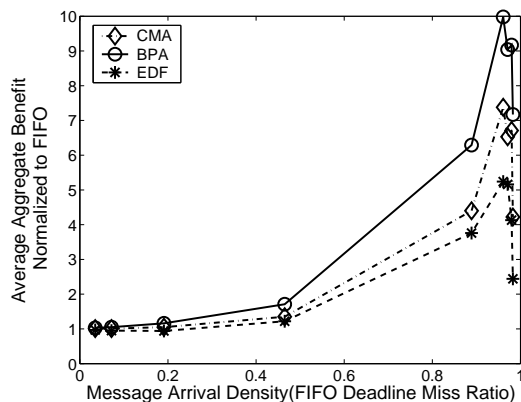
We were also interested to determine how BPA and CMA perform when the arrival density of messages increases under heterogeneous benefit functions.

Figure 7.3(a) shows the (end-to-end) message deadline-miss ratio of BPA, CMA, EDF, and FIFO under heterogeneous benefit functions. From the figure, we observe that the deadline-miss ratio of all algorithms increases as the message arrival density increases. Further, we observe that the deadline-miss ratio of BPA is the smallest. This is followed by the ratios of CMA and EDF. FIFO has the largest deadline-miss ratio. The results are the same as that of the homogeneous functions.

In Figure 7.3(b), we normalize the aggregate benefit of the algorithms by dividing their aggregate benefits with that of FIFO. Thus, FIFO has a normalized aggregate benefit of 1 in this figure. From the figure, we observe that BPA performs the best here, EDF the worst, and CMA in between. Since BPA can perform 3 to 9 times better than that of FIFO during overload situations, this demonstrates that BPA works quite well for heterogeneous benefit functions.



(a) Deadline Miss Ratio under Heterogeneous



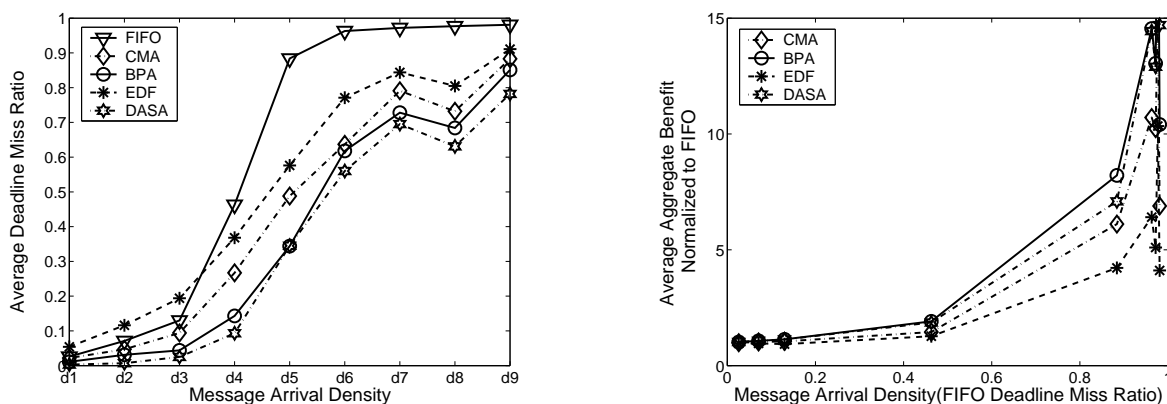
(b) Normalized Aggregate Benefit under Heterogeneous

Figure 7.3: Performance Under Increasing Message Arrival Density and Heterogeneous Benefit Functions

## 7.5 Performance Comparison with DASA under Rectangular Benefit Function

Similar to the single processor case study, we were also interested in comparing BPA with DASA for rectangular benefit functions in a switched network environment.

Figure 7.4(a) shows the (end-to-end) message deadline-miss ratio of BPA, CMA, DASA, EDF, and FIFO. Figure 7.4(b) shows the normalized average benefit comparisons for all algorithms. Both figures indicate that BPA and DASA perform very close to each other and that both are better than EDF and CMA.



(a) Deadline Miss Ratio for DASA Comparison

(b) Normalized Aggregate Benefit for DASA Comparison

Figure 7.4: Performance Under Increasing Message Arrival Density and Rectangular Benefit Functions

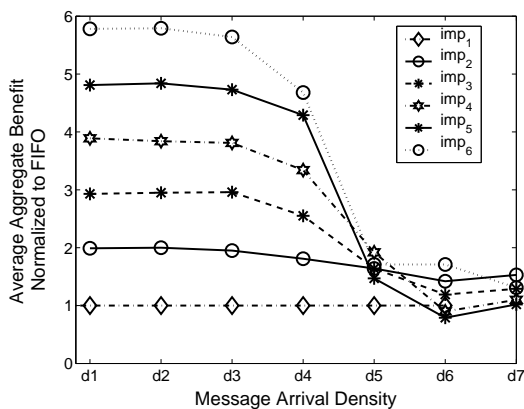
## 7.6 The Importance Factor: Performance Comparison under Rectangular Benefit Function

In all of the previous studies, we assume that the maximum benefit of each packet could be any value even if all the packets were generated from a single application. This was to facilitate the performance evaluation of the algorithms in a general manner.

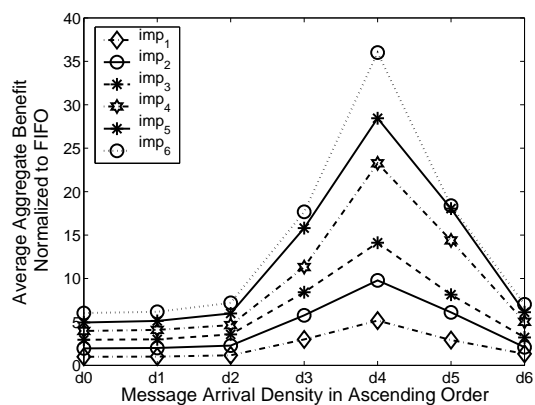
In practice, maximum benefits may not have such an infinite value domain. In fact, it is possible for packets belonging to the same application to have the same maximum benefit. However, packets belonging to different applications may have different maximum benefits. Moreover, the more *functionally important* the application is, the bigger the maximum benefit

value will be. Note that such functional importance can be completely orthogonal to timing constraints.

To accommodate the functional importance of applications, we consider an *importance factor*. We consider the maximum benefit of the least important application as a baseline value. The maximum benefit of all other applications are then obtained by multiplying their importance factor with the baseline value. By doing so, we expect a more important application flow to obtain more aggregate benefit.



(a) Average Aggregate Benefit for FIFO

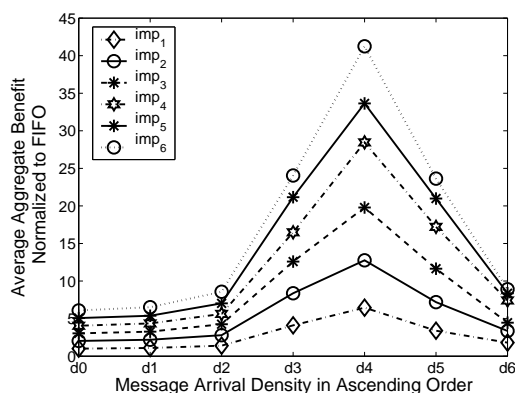


(b) Average Aggregate Benefit for CMA

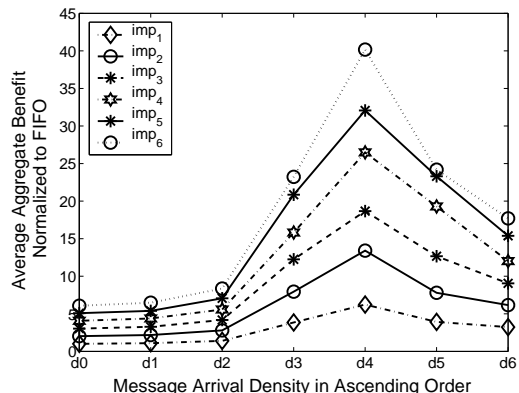
Figure 7.5: Performance Under Increasing Message Arrival Density and Rectangular Benefit Functions for FIFO and CMA

Since the functional importance factor is independent of the heterogeneity (or homogeneity) of timing constraints, we consider homogeneous benefit functions for our simulation study. Furthermore, we focus on rectangular functions as that will allow us to include DASA in our comparisons.

From Figure 7.5(a), we observe that when message arrival density is less than some value, application flows with bigger importance factors obtain higher average aggregate benefit. However, when the message arrival density further increases, the advantage of importance factor disappears. This is because in the situation of underload there are few messages that missed their deadlines, and high maximum benefit value means higher aggregate benefit; On the other hand, in the situation of overload, deadline misses happen more frequently and an application with higher importance factors may miss more deadlines.



(a) Average Aggregate Benefit for BPA



(b) Average Aggregate Benefit for DASA

Figure 7.6: Performance Under Increasing Message Arrival Density and Rectangular Benefit Functions for BPA and DASA

Figure 7.5(b) illustrates that when the message arrival density increases, CMA can achieve higher and higher aggregate benefit until some optimal "point". After that point, the performance decreases. More importantly, an application with high importance factor is guaranteed to obtain higher aggregate benefit no matter how large the message arrival density is. This is due to the effectiveness of CMA queuing algorithm.

Figure 7.6(a) illustrates the result for BPA, which is quite similar to that of CMA except that BPA achieves more benefit gains over CMA. This is because BPA performs better than CMA.

Figure 7.6(b) illustrates the result for the DASA algorithm, which is very close to that of BPA. This is because both algorithms have similar queuing performance in a switched network environment. This demonstrates that CMA, BPA and DASA can ensure that an application flow with higher importance factor obtains higher aggregate benefit. Furthermore, BPA and DASA can achieve more benefit gains than CMA and DASA performs very close to BPA.

# Chapter 8

## Implementation of BPA

Our goal in implementing BPA is to construct a prototype soft real-time switched Ethernet network, conduct experiments using the network implementation by comparing BPA with other packet scheduling algorithms, and thus measure the actual performance of BPA. As discussed in Chapter 1, we construct the network by prototyping an Ethernet switch using a Pentium PC. Furthermore, we implement BPA in software—in the network protocol stack of the Linux kernel—for packet scheduling at end-hosts and at the switch.

In Section 8.1, we overview the prototype network. We discuss how we implement packet scheduling algorithms and conduct packet switching in the Linux kernel in Sections 8.2 and 8.3, respectively. Section 8.3.1 describes how we transport real-time attributes of real-time messages in the network. We summarize our implementation of the packet scheduling algorithms in the Linux kernel in Section 8.3.2.

## 8.1 Prototype Setup

We use a Pentium PC with a single processor of 450MHZ and 256MB of memory as the central switch. Within the switch, we use two ZX346Q 4-port Ethernet adapters from ZNYX [18] to handle packet switching. The ZX346Q uses PCI bus to communicate with the PC system.

Since we are studying a switch scheduling algorithm, we need several PCs to generate sufficient traffic on the outgoing ports at the switch. However, since the computational demand on each end-host is quite simple—capability to send and receive packets—we can use a single PC to emulate several end-hosts that send messages. Such an emulation not only saves cost and space, but also simplifies the testing procedures.

Thus, we use another ZX346Q 4-port Ethernet adapter in a PC with a single processor of 500 MHZ and 256MB memory to emulate four end-hosts.

For the receiving end-host, we use another Pentium PC with a single processor of 250 MHZ and 64 MB of memory for receiving packets. Therefore, in the prototype system, there are a total of four sending “hosts” and one receiving host.

The four sending ports connect to the switch with 100Mbps full duplex links and the central switch connects to the receiving host with a 10Mbps half-duplex link. In our experiments, we assume that real-time messages unidirectionally flow from the four sending “hosts” to the receiving host. Since in real-time packet transmission, 2-way hand shaking mode is not used, the unidirectional real-time message flow and 10Mbps half-duplex link do not affect our performance comparison for real-time packet scheduling algorithms.

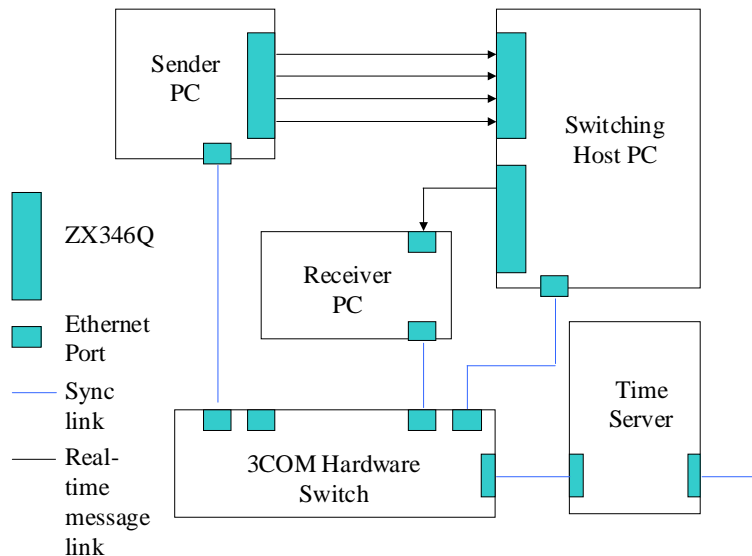


Figure 8.1: The BPA Prototype

The purpose of selecting a 10Mbps link as the receiving link is to generate sufficiently large packet traffic on the switch output port e.g., several thousands of packets, so that the effect of the complexity of the packet scheduling algorithm (at the switch) can be evaluated.

To synchronize the clocks of the PCs in the prototype LAN, which is a critical assumption made by BPA, we use the `xntpd` [34] implementation of the NTP protocol [28] for Linux platforms. To make time synchronization as good as possible, we use a dedicated 100Mbps link in all three PCs of the prototype LAN to connect to the time server in our laboratory through another 3COM hardware-level Ethernet switch. The laboratory time server is synchronized with the Virginia Tech time server.

The entire prototype system is shown in Figure 8.1.

## 8.2 Implementation of Packet Scheduling Algorithms in Linux Kernel

We believe that the best location to implement BPA is at the MAC layer because of the accuracy with which queueing (or scheduling) results can be obtained. Since the Ethernet MAC layer and the on-board sending poll are implemented in hardware, it is not practical to implement BPA within the Network Interface Card's (NIC's) MAC layer by software means. However, if we implement BPA very close to the hardware in the switched structure, the only factor affecting the queueing accuracy will be the on-board sending poll, because the CSMA/CD protocol has no effect due to the collision-free situation on the network.

On the other hand, since the on-board sending poll is small compared with the total queue length, we believe that it will not significantly affect the accuracy of the experimental results characterizing the performance of the scheduling algorithms. Therefore, we select a layer just above the Ethernet device driver to implement BPA.

Linux has a set of mature mechanisms to support implementation of packet scheduling algorithms or queuing disciplines.<sup>1</sup> For the flexibility of adding any queuing discipline to a network device, a field called `qdisc` was added to the `device` structure in the Linux kernel. Linux kernel requires every network device driver to check if there is any queuing discipline associated with the device either when (a) the hardware device is requesting for packets or

---

<sup>1</sup>In Sections 8, 9, and 10 that discuss implementation of BPA, we use the term *queueing discipline* to imply a *scheduling algorithm*, due to the synonymous usage of these terms in Linux kernel implementations.

when (b) a new packet arrives from the local network layer.

If a queuing discipline is associated with the device and when the device is requesting for packets, the device driver will awaken the `dequeue` function from the associated queuing discipline. The `dequeue` function will select a new packet to be sent to the on-board sending poll. On the other hand, when a new packet arrives from the local network layer, the device driver will call the `enqueue` function from the associated queuing discipline to enqueue the new packet.

All packet queuing programs in Linux reside under the directory `net/sched` [35]. To associate a new queuing discipline to a network device, there are two choices. The first is to compile the implementation of the queuing discipline into the kernel by registering through `pktsched_init` in `net/sched/sch_api.c`. The second way is to register the implementation of the queuing discipline using `register_qdisc` from the `init_module` so that the queuing discipline can be linked as a module. We use the second method due to its flexibility.

To facilitate the addition and deletion of queuing disciplines, we use the `tc` traffic controller utility that is available in the Linux public domain [36].

In our prototype LAN, all three PCs contain implementations of the same queuing discipline associated with each real-time Ethernet device, with a minor difference. The difference in the queuing disciplines is that the transmission time for the BPA feasibility test in the sender is twice as that in the switch, since each packet needs to travel two Ethernet devices.

## 8.3 Packet Switching Inside Linux Kernel Using The Linux Bridge Program

In the central switch, we need a robust and fast software switch engine to conduct packet switching. We use Bohme and Buytenhak's Linux Bridge program [37, 38] as our switch engine.

Table 8.1: Round Trip Time Comparison for Different Network Configurations

| <i>Ping Condition</i>                   | <i>Average Round Trip Time</i> |
|---|--------------------------------|
| Direct Point-to-Point Connection        | 140 $\mu$ s                    |
| 3COM Hardware Switch                    | 180 $\mu$ s                    |
| Linux Bridge Software Switch            | 240 $\mu$ s                    |
| Linux Bridge Plus BPA Queuing Algorithm | 280 $\mu$ s                    |

To evaluate the performance of the Linux Bridge switching software, we compare its ping results with that of a 3COM hardware switch and direct point-to-point connections. The round-trip average time measurements are shown in Table 8.1.

### 8.3.1 Transporting Real-Time Attributes of Real-Time Messages

In the UDP/IP or TCP/IP protocol suite, each message from an application program is encapsulated into one or more IP packets. IP packets are then transmitted as one or more Ethernet frames over the physical layer.

Since each real-time message has some inherent real-time attributes, these attributes must be carried by each corresponding IP packet so that IP packet-based real-time queuing discipline can be implemented using the real-time attributes of the packet.

The real-time attributes of a real-time message as well as that of a real-time IP packet include benefit function type, queuing algorithm type, maximum benefit value, relative deadline, time stamp, and absolute deadline. Although we can calculate absolute deadline from the time stamp and the relative deadline, we include relative deadline, time stamp, and the absolute deadline in each IP packet for the purpose of saving computational costs. The side-effect of doing so is the waste in network bandwidth due to the increased IP packet header size. However, if the average real-time message size is large enough, the overhead of several bytes is not significant. In a real system design, we may need to trade off the waste in network bandwidth against the computational overhead, based on real-time message size and other system configurations.

We include the queuing algorithm type in an IP packet only for the convenience in testing and performance comparison.

Since all the real-time attributes need to be set within the IP header of each packet, we want to make good use of each IP header field so that the network traffic overhead can be reduced as much as possible. Thus, we use the 8-bit TOS(Type of Service) field to carry both the queuing algorithm type (highest 2 bits) and the benefit function type (3 bits following the queuing algorithm bits). The 8-bit TTL(Time To Live) field is used to carry the maximum benefit value since the original TTL function is not necessary in our LAN. These real-time

---

```
TOS: setsockopt(sock, IPPROTO_IP, IP_TOS, &BF_type, sizeof(BF_type))

TTL: setsockopt(sock, IPPROTO_IP, IP_TTL, &max_BF, sizeof(max_BF))

OPTION:

rtdlopt.type=233; /* Select a new type number only used by BPA */

rtdlopt.size=sizeof(rtdlopt)-1;

rtdlopt.pointer=5; /* Refer to RFC 791 for details */

rtdlopt.flag=0x70;

memcpy(rtdlopt.tm_stamp, &tm_stamp, 8);

memcpy(rtdlopt.relative_DL, &relative_DL, 4);

memcpy(rtdlopt.abs_DL, &abs_DL, 8);

Rtdlopt.endchar=IPOPT_END;

setsockopt(sock, IPPROTO_IP, IP_OPTIONS, (char *)&rtdlopt, sizeof(rtdlopt))
```

---

Figure 8.2: Real-Time Parameters Within IP Header

attributes can be carried in the `IP_OPTION` field if `TTL` and `TOS` fields are used in a system.

We set the time stamp, the relative deadline (32 bits), and the absolute deadline (64 bits) in the `IP_OPTION` field since this field is currently not used by the Linux kernel. Both time stamp and absolute deadline are of type structure `time_val` and the time stamp is obtained by the `gettimeofday` function call.

To set `TOS`, `TTL` and `OPTION` field in the IP header, we use the `setsockopt` function [39].

To facilitate the setting and retrieving of timing attribute-related operations, we defined a header file called `dlopt.h` in the `/include/net/sched` directory.

Figure 8.2 shows sample code that illustrates how `TOS`, `TTL`, and `OPTION` fields of the IP header are set.

### 8.3.2 Queuing Algorithm Implementation

In the Linux kernel, each network device has a doubly linked-list of type structure `sk_buff` [40]. Each element in the list corresponds to an IP packet. Therefore, we can access the IP packet header through the corresponding `sk_buff` element. Moreover, in the header file `/include/net/linux/skbuff.h`, all the necessary low-level `sk_buff` linked-list operations such as enqueue, dequeue, insert, and append are defined.

All queuing discipline-related operations such as enqueue and dequeue are defined in the structure `Qdisc_ops`. Based on different enqueue and dequeue operations, multiple queuing algorithms can be easily implemented.

Table 8.2: Implementation of Queuing (or Scheduling) Algorithms

| <i>Type</i>                           | <i>Enqueue</i>  | <i>Dequeue</i>  |
|---------------------------------------|---|---|
| <i>FIFO</i>                           | Append a new <i>skb</i> to end of <i>skb list</i>   | Dequeue from head of <i>skb list</i>  |
| <i>EDF_NDMC</i>                       | Insert a new <i>skb</i> to <i>skb list</i> based on the ascending order of the packet deadline. No deadline-miss check.   | Dequeue from head of the deadline-ordered <i>skb list</i> .   |
| <i>EDF_DMC</i>                        | Insert a new <i>skb</i> to <i>skb list</i> based on the ascending order of the packet deadline with deadline-miss check. If the new <i>skb</i> has already missed its deadline, it is dropped.  | Dequeue from head of the deadline-ordered <i>skb list</i> .   |
| <i>BPA</i><br><i>CMA</i><br><i>OQ</i> | If the new packet has missed its deadline, it is dropped. Otherwise, append it to the end of <i>skb list</i> and set the new packet arrival flag so that BPA, CMA and OQ can be triggered next. | If no new packet joins in after the last dequeue operation, just dequeue from head of the ordered <i>skb list</i> . Otherwise, reorder <i>skb list</i> based on BPA, CMA or OQ policy. For BPA, non-feasible packets are dropped during the reordering process. |

In order to do a thorough evaluation of BPA, we implemented six queuing algorithms: FIFO, EDF\_NDMC, EDF\_DMC, BPA, CMA, and OQ (for Optimal Queuing). Here, EDF\_NDMC means Earliest Deadline First without Deadline Miss Check and EDF\_DMC means Earliest Deadline First with Deadline Miss Check.

Our rationale for using EDF\_DMC and EDF\_NDMC in the experimental study is because of the feasibility test (or Deadline Miss Check) employed by BPA in Step (5.2.2) and Step (5.2.3) (see Figure 5.1). Thus, a comparison of BPA and EDF\_DMC will help us evaluate the impact of BPA's scheduling scheme toward BPA's performance since both BPA and EDF\_DMC employ the feasibility test. On the other hand, a comparison of EDF\_DMC and EDF\_NDMC will help us evaluate the impact of the feasibility test itself, since EDF\_DMC employs the test and EDF\_NDMC does not.

Table 8.2 summarizes the implementation of the six queuing algorithms.

# Chapter 9

## Test Bench Setup

We now discuss the test bench setup of the prototype implementation for conducting experiments and measuring actual performance. We discuss the experimental parameters of the set up and issues involved with emulating end-hosts, generating heavy message traffic, and selecting benefit functions and benefit values in the sections that follow.

### 9.1 Experimental Parameters for Generating Message Traffic

The attributes of a real-time message include message length, message deadline, maximum benefit value, and inter-arrival time. For measuring actual performance of BPA and other algorithms in the test bench set up, we generate the message traffic using the same distribution

functions as that used in our simulation studies.

The expressions that we used for generating real-time message attributes are shown in Table 9.1.

We set the minimum message size as 40, which is reasonable for real-time messages. Further, we set the deadline to be larger than the transmission time. Furthermore, we set the maximum benefit to be a normal distribution between 1 and 1000. The message inter-arrival time is also defined as a normal distribution.

In all the expressions shown in Table 9.1, the *max* function is used to avoid very small values that are abnormal and the *LevelKnob* variable is used to control the traffic density. The bigger the *LevelKnob* value, bigger the message size and shorter the message inter-arrival time.

Table 9.1: Expressions for Real-Time Message Attributes

| <i>Properties</i>            | <i>Expressions</i>  |
|------------------------------|---|
| Message Length               | $\text{Max}(\exp(500+\text{LevelKnob}*50),40)$                                      |
| Deadline (in $\mu\text{s}$ ) | $\text{Max}(\text{Transmit\_time}+30700, \exp(\text{Transmit\_time}+30700))$        |
| Maximum Benefit              | $\text{Norm}(1,1000)$   |
| Ia_time (in $\mu\text{s}$ )  | $\text{Max}(100,\text{norm}(1500-\text{LevelKnob}*100,31000-\text{LevelKnob}*500))$ |

In addition to the traffic configuration shown in Table 9.1, we also studied cases where message size is small and limited to a range so that low-level segmentation is avoided or rare.

To facilitate a comparative study, we must construct exactly the same traffic condition for

all tests. A simple way to do this is to read traffic attribute data from fixed files. Therefore, we generated 16 traffic files based on the expressions in Table 9.1 for 16 traffic levels. We repeated the experiments for multiple queuing algorithms and multiple benefit functions by reading exactly the same 16 traffic files.

## 9.2 Emulating Four End-Hosts With a Single PC

In our prototype, four sending “hosts” are emulated by four Ethernet ports. In order to send messages through a specific port, we configure each Ethernet port with a different IP address. More importantly, all four IP addresses belong to four different network addresses so that the routing table can be built by the Linux kernel exactly as what we want. However, on the receiver side, it has only one IP address. Therefore, we set three IP alias addresses for the receiver so that the receiver can be accepted by four different network address domains.

The IP address configuration is shown in Table 9.2.

Table 9.2: IP Address Configuration for Senders and the Receiver

| <i>Sender</i>      |                   | <i>Receiver</i>           |                         |
|--------------------|-------------------|---------------------------|-------------------------|
| <i>Port Number</i> | <i>IP address</i> | <i>Port: Alias number</i> | <i>IP alias address</i> |
| eth0               | 128.173.52.101    | eth0                      | 128.173.52.17           |
| eth1               | 128.183.52.101    | eth0:1                    | 128.183.52.17           |
| eth2               | 128.193.52.101    | eth0:2                    | 128.193.52.17           |
| eth3               | 128.203.52.101    | eth0:3                    | 128.203.52.17           |

Thus, for example, to send a message to the receiver through “host 3,” that is, `eth2`, we select the IP address `128.193.52.17` as the destination address. According to the routing table stored in the Linux kernel of the sender host, only those packets of the masked destination address `128.193.48.0` will be forwarded out through `eth2`. Since our network mask is `255.255.248.0`, the masked address for `128.193.52.17` is exactly `128.193.48.0`. Therefore, the message will be send out through “host 3.”

### 9.3 Generating Heavy Message Traffic

In order to evaluate the performance of a packet queuing algorithm, the packet arrival rate should be faster than the network link speed so that packets will be accumulated in the outgoing network queue.

For the sending “hosts,” we use the idea: “keep sending and then wait” to generate sufficient traffic on the outgoing link as well as to control the queue length. Furthermore, we ensure that the intervals between two consecutive send operations are sufficiently short so that significant number of packets queue-up in the Ethernet device queue.

To minimize the effect of the scheduling interval of the Linux process scheduler, we use a single process to send sufficiently large number of messages to all four ports during each message-sending cycle. With multiple processes, the scheduling interval can cause large delays between consecutive message-send operations for a port. This can negatively affect the packet queue of the port [41]. Once the messages are sent, the process waits for the next

cycle by a small interval value.

During each message-sending cycle, we assume that each port is collecting messages from multiple applications at almost the same time. Thus, messages will queue-up at the outgoing port. However, if messages are sent too fast, the kernel memory can be exhausted. Hence, after each message-sending cycle, the process waits for a short interval of time before starting the next cycle.

## 9.4 Benefit Functions and Benefit Assignment

In our test bench experiments, we consider four benefit functions. These include rectangular, soft-rectangular, linear, and quadratic. We do not consider exponential benefit functions for two reasons. First, it is computationally expensive to evaluate an exponential function inside the kernel since it requires math-emulation that is slow in a PC. Second, in case we need an exponential benefit function, we can compose quadratic functions and approximate. Thus, we focus on the four benefit functions described here.

The BPA algorithm schedules packets using the benefit function of packets with the goal of maximizing aggregate message-level benefit, where packets inherit the same benefit functions as that of their parent messages. To evaluate the effectiveness of this strategy and thus the relationship between packet-level and message-level scheduling, we study two cases: (1) “big” message sizes and (2) “small” message sizes. With big message sizes, message decomposition into packets clearly occurs. However, in the case of small message sizes, a message will not

or rarely be segmented into packets during its lifetime.

Maximum benefit is an important attribute of a real-time message. Similar to our simulation experiments, we consider a normal distribution function to obtain the value for maximum benefit in our implementation experiments. However, to evaluate the impact of maximum benefit values on the performance of the algorithms, we also consider a deterministic assignment of maximum benefits. For example, if an algorithm shows an advantage over others even when the ratio of the maximum benefit value to the minimum benefit value is small, this indicates that the algorithm is quite efficient because it can distinguish a small difference.

Thus, we consider two benefit assignments called,  $BA_L$  and  $BA_H$ . In  $BA_L$ , we repeatedly assign the maximum benefit values 1, 5, 15, 17, 20, 23, 25, 280, and 1000 to the messages. In  $BA_H$ , we repeatedly assign the maximum benefit values 1, 5, 15, 17, 20, 23, 25, 28, and 8000 to the messages. Thus, the ratio of maximum benefit value to minimum benefit value of  $BA_L$  is 1000, while that of  $BA_H$  is 8000. Furthermore, the average of the difference between successive maximum benefit values of  $BA_L$  is smaller than that of  $BA_H$ .

# Chapter 10

## Performance Measurement and Analysis

We conducted a number of experiments in the implementation test bed to measure the performance of BPA and other queueing algorithms. We first discuss the infeasibility of conducting implementation experiments with the Optimal Queueing algorithm and CMA in Section 10.1. We then report the performance results obtained with BPA, EDF\_DMC, EDF\_NDMC, and FIFO algorithms in Sections 10.2, 10.3, and 10.4.

## 10.1 Infeasibility of Optimal Queuing Algorithm and CMA

From our experiments, we observed that both CMA and Optimal Queuing algorithms are not practical as packet scheduling algorithms due to their huge computational costs in terms of either time or space.

For the Optimal Queuing algorithm, we observed that when the queue size became larger than 13 packets, the algorithm would make the program execute for 10 hours to get the (optimal) result on a Pentium PC with a processor speed of 450 MHZ. Thus, clearly, the Optimal Queuing algorithm is not practical.

The huge demand for kernel memory of CMA makes it practically infeasible when the queue size is several thousands of packets long. CMA uses a precedence matrix to store real-time attribute values of each message (see Chapter 4). Thus, if the queue size is 4K, and each (queuing) message requires 14 bytes to store real-time attributes, the total demand for kernel memory would be  $4K \times 4K \times 14 = 224\text{Mbytes}$ . This will exhaust the entire kernel space.

However, for BPA, we only need a 1-dimensional array to store the real-time attributes of each (queuing) message. Thus, for BPA, we only need 56K of kernel memory for auxiliary storage space. For the two EDF algorithms, there is no need for similar storage. Therefore, only BPA, EDF\_DMC, EDF\_NDMC are practical and can be compared with FIFO.

## 10.2 Performance Comparison by Message Size and Maximum Benefit Value

In order to thoroughly evaluate the performance of the algorithms, we conducted experiments for all four benefit functions under all five traffic conditions. Experiments were repeated 93 times for some cases and 48 times for some other cases.

The results are illustrated in Table 10.1 through Table 10.5. From the tables, we observe that in each traffic type, BPA always achieves the largest final benefit among all four algorithms except in some cases of soft rectangular benefit functions, where EDF\_DMC performs slightly better than BPA. We believe that the exceptional cases where EDF\_DMC performs slightly better than BPA is due to the difference that exists between the pseudo-slope approximation done in BPA and the specific soft rectangular benefit functions considered in the experiments.

With different benefit functions, we observe that the advantage of BPA is not that obvious for rectangular benefit functions. However, it is quite obvious for the other three benefit functions where BPA achieves final benefits that are 6 to 38 times as that achieved by FIFO.

From Table 10.1 through Table 10.5, we also observe that BPA has the minimum deadline-miss ratio among all four algorithms. The difference between BPA and other algorithms for the deadline-miss ratio metric is quite significant.

Thus, from the results, we observe that BPA performs very well even in a software-switching platform. We believe that this is due to the average-case complexity of BPA, which is far

less than the worst-case cost of  $O(n^2)$ .

The dominating step of the BPA algorithm is the inner for-loop, which can iterate a maximum of  $n$  times. (The outer for-loop iterates for  $n$  times.) However, it turns out that the pseudo-order that BPA determines prior to the nested for-loop is very close to the final order. Thus, the exact number of iterations performed during the inner for-loop is far less than  $n$ . This significantly reduces the average-case complexity of BPA. Such results were frequently observed during our simulation studies.

Table 10.1: T0: Big Message Size and Normal Distribution Benefit

|                 | <i>Benefit Normalized to FIFO</i> |                |            |                 | <i>Average Deadline Miss Ratio</i> |                |            |                 |
|-----------------|-----------------------------------|----------------|------------|-----------------|------------------------------------|----------------|------------|-----------------|
|                 | <i>FIFO</i>                       | <i>EDF_DMC</i> | <i>BPA</i> | <i>EDF_NDMC</i> | <i>FIFO</i>                        | <i>EDF_DMC</i> | <i>BPA</i> | <i>EDF_NDMC</i> |
| <i>Rect</i>     | 1                                 | 1.34           | 1.61       | 0.86            | 0.56                               | 0.50           | 0.5        | 0.62            |
| <i>SoftRect</i> | 1                                 | 6.35           | 7.50       | 1.30            | 0.58                               | 0.52           | 0.48       | 0.54            |
| <i>Linear</i>   | 1                                 | 2.73           | 5.32       | 1.22            | 0.60                               | 0.50           | 0.44       | 0.52            |
| <i>Quad</i>     | 1                                 | 4.80           | 6.35       | 1.57            | 0.58                               | 0.51           | 0.5        | 0.48            |

Table 10.2: T1: Small Message Size and Normal Distribution Benefit

|                 | <i>Benefit Normalized to FIFO</i> |                |            |                 | <i>Average Deadline Miss Ratio</i> |                |            |                 |
|-----------------|-----------------------------------|----------------|------------|-----------------|------------------------------------|----------------|------------|-----------------|
|                 | <i>FIFO</i>                       | <i>EDF_DMC</i> | <i>BPA</i> | <i>EDF_NDMC</i> | <i>FIFO</i>                        | <i>EDF_DMC</i> | <i>BPA</i> | <i>EDF_NDMC</i> |
| <i>Rect</i>     | 1                                 | 1.63           | 1.84       | 0.83            | 0.44                               | 0.28           | 0.21       | 0.55            |
| <i>SoftRect</i> | 1                                 | 8.45           | 5.96       | 1.66            | 0.47                               | 0.36           | 0.19       | 0.42            |
| <i>Linear</i>   | 1                                 | 29.18          | 38.39      | 5.89            | 0.40                               | 0.34           | 0.17       | 0.47            |
| <i>Quad</i>     | 1                                 | 4.13           | 5.53       | 0.88            | 0.54                               | 0.51           | 0.47       | 0.56            |

From Table 10.1 and Table 10.2, we also observe that BPA achieves similar advantages over FIFO between message-level scheduling and packet-level scheduling except for linear benefit functions. This means that although BPA is doing packet-level scheduling, the final result at the message-level is also good.

On the other hand, since BPA is doing packet-level scheduling and its pseudo-slope for linear benefit functions is actually the real slope, it is not surprising that for linear benefit functions and small message sizes, BPA can achieve the best final benefit, which is 38 times as that achieved by FIFO.

Another interesting observation is that EDF\_NDMC presents almost no advantage over that of FIFO. This is reasonable, since during overloaded network situations, EDF\_NDMC does no deadline-miss checks. This wastes network bandwidth similar to that of FIFO. But the EDF-ordering wastes more network bandwidth than that of FIFO due to EDF's domino effect [33]. This explains why EDF\_NDMC performs worse than FIFO. Thus, clearly, a deadline-feasibility test is important during overloaded situations.

This hypothesis is further validated by the performance of EDF\_DMC, which to some extent, is close to that of BPA. Although EDF\_DMC only does deadline-ordering, it has a better complexity than that of BPA. This advantage of EDF\_DMC becomes significant when the pseudo-slope of BPA is not close to the real slope or when the benefit-ordering is close to the deadline-ordering.

Table 10.3 and Table 10.4 show results for small message size and deterministic  $BA_L$  and

Table 10.3: T2: Small Message Size and  $BA_L$  Maximum Benefit

|                 | <i>Benefit Normalized to FIFO</i> |                |            |                 | <i>Average Deadline Miss Ratio</i> |                |            |                 |
|-----------------|-----------------------------------|----------------|------------|-----------------|------------------------------------|----------------|------------|-----------------|
|                 | <i>FIFO</i>                       | <i>EDF_DMC</i> | <i>BPA</i> | <i>EDF_NDMC</i> | <i>FIFO</i>                        | <i>EDF_DMC</i> | <i>BPA</i> | <i>EDF_NDMC</i> |
| <i>Rect</i>     | 1                                 | 1.59           | 3.00       | 1.12            | 0.50                               | 0.35           | 0.31       | 0.45            |
| <i>SoftRect</i> | 1                                 | 12.64          | 20.43      | 1.46            | 0.51                               | 0.31           | 0.26       | 0.50            |
| <i>Linear</i>   | 1                                 | 3.35           | 3.52       | 0.93            | 0.55                               | 0.29           | 0.14       | 0.57            |
| <i>Quad</i>     | 1                                 | 7.31           | 6.24       | 1.11            | 0.56                               | 0.28           | 0.14       | 0.57            |

Table 10.4: T3: Small Message Size and  $BA_H$  Maximum Benefit

|                 | <i>Benefit Normalized to FIFO</i> |                |            |                 | <i>Average Deadline Miss Ratio</i> |                |            |                 |
|-----------------|-----------------------------------|----------------|------------|-----------------|------------------------------------|----------------|------------|-----------------|
|                 | <i>FIFO</i>                       | <i>EDF_DMC</i> | <i>BPA</i> | <i>EDF_NDMC</i> | <i>FIFO</i>                        | <i>EDF_DMC</i> | <i>BPA</i> | <i>EDF_NDMC</i> |
| <i>Rect</i>     | 1                                 | 1.96           | 3.03       | 1.03            | 0.51                               | 0.31           | 0.26       | 0.50            |
| <i>SoftRect</i> | 1                                 | 6.57           | 5.90       | 0.91            | 0.52                               | 0.31           | 0.21       | 0.55            |
| <i>Linear</i>   | 1                                 | 3.40           | 4.22       | 0.95            | 0.53                               | 0.30           | 0.19       | 0.57            |
| <i>Quad</i>     | 1                                 | 7.82           | 6.87       | 1.26            | 0.55                               | 0.29           | 0.14       | 0.56            |

$BA_H$  maximum benefit assignments for rectangular, soft rectangular, and quadratic benefit functions. The final benefit results are better than small message size and normal distribution benefit cases, while for linear benefit functions, the latter is much better than the former.

Table 10.5: T4: Big Message Size and  $BA_H$  Maximum Benefit

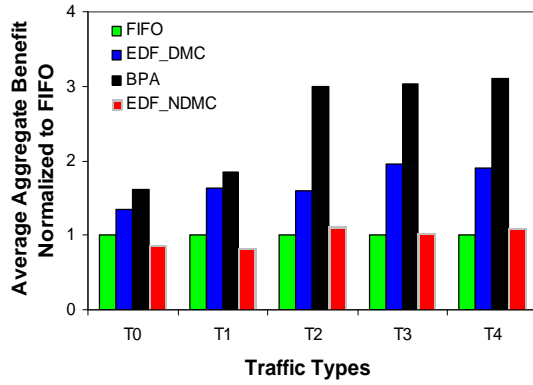
|                 | <i>Benefit Normalized to FIFO</i> |                |            |                 | <i>Average Deadline Miss Ratio</i> |                |            |                 |
|-----------------|-----------------------------------|----------------|------------|-----------------|------------------------------------|----------------|------------|-----------------|
|                 | <i>FIFO</i>                       | <i>EDF_DMC</i> | <i>BPA</i> | <i>EDF_NDMC</i> | <i>FIFO</i>                        | <i>EDF_DMC</i> | <i>BPA</i> | <i>EDF_NDMC</i> |
| <i>Rect</i>     | 1                                 | 1.90           | 3.11       | 1.09            | 0.68                               | 0.66           | 0.53       | 0.67            |
| <i>SoftRect</i> | 1                                 | 4.60           | 6.57       | 0.90            | 0.59                               | 0.50           | 0.52       | 0.66            |
| <i>Linear</i>   | 1                                 | 1.65           | 3.41       | 1.24            | 0.65                               | 0.61           | 0.48       | 0.57            |
| <i>Quad</i>     | 1                                 | 3.47           | 10.28      | 1.55            | 0.65                               | 0.47           | 0.32       | 0.55            |

Table 10.5 shows the results for big message size and  $BA_H$  benefit assignment. The performance here is very close to that observed in Table 10.4. This further demonstrates that packet-level benefit scheduling favors message-level scheduling as a whole.

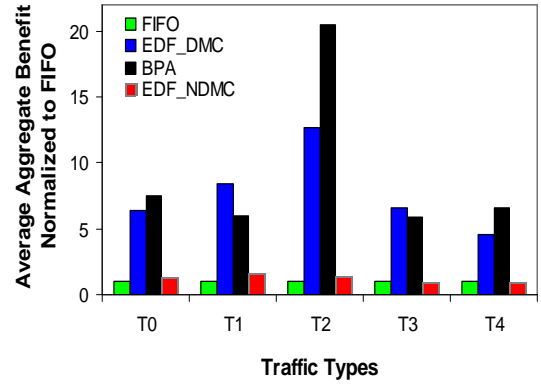
### 10.3 Performance Comparison by Benefit Function Types

The comparison of the four queuing algorithms based on different benefit functions are shown in Figures 10.1, 10.2, 10.3, and 10.4. While Figures 10.1 and 10.2 show the average aggregate benefit that is normalized with respect to FIFO, Figures 10.3 and 10.4 show the average deadline-miss ratio.

From Figures 10.1 and 10.2, we observe that the benefit function order where BPA performs

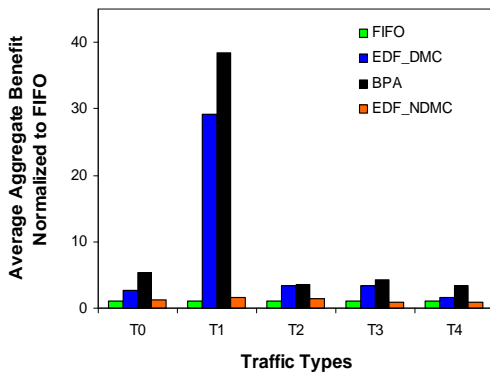


(a) Rectangular Function

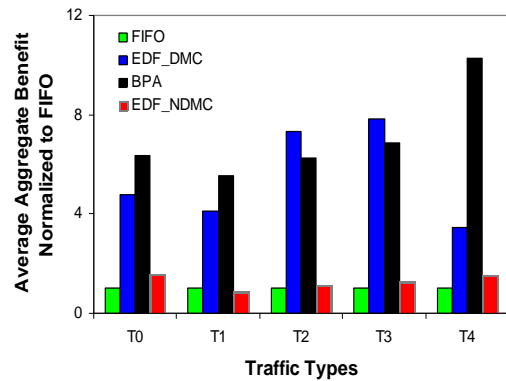


(b) Soft-Rectangular Function

Figure 10.1: Average Aggregate Benefit With Respect to FIFO Under 5 Traffic Types for Rectangular and Soft-Rectangular Functions



(a) Linear Function



(b) Quadratic Function

Figure 10.2: Average Aggregate Benefit With Respect to FIFO Under 5 Traffic Types for Linear and Quadratic Functions

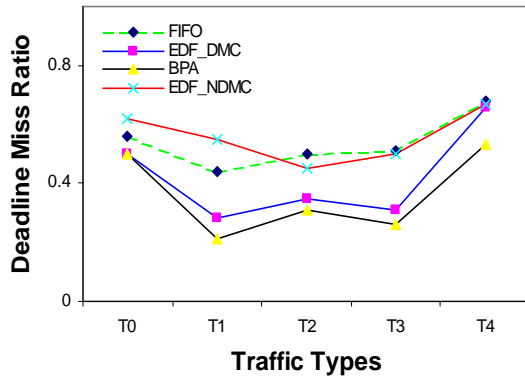
the best to the worst when compared with FIFO is given by: Linear  $>$  Soft Rectangular  $>$  Quadratic  $>$  Rectangular. From this order, we conclude that closer the pseudo-slope of BPA to the real benefit function slope, greater the advantages BPA has over that of FIFO.

Table 10.6: BPA Favored Traffic Types

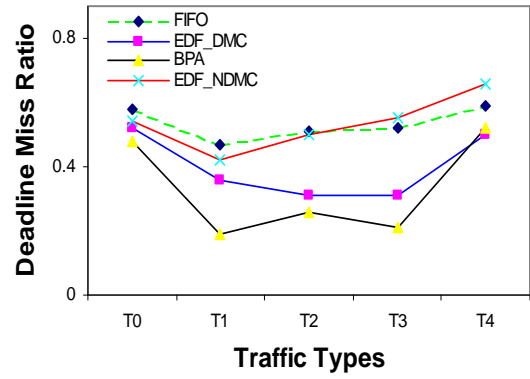
| <i>Benefit function</i> | <i>Favored Traffic Types</i> |
|-------------------------|------------------------------|
| Linear                  | Type 1                       |
| Soft-Rect               | Type 2                       |
| Quad                    | Type 4                       |
| Rect                    | Type 2, Type 3, Type 4       |

Another conclusion we draw is about the most favored traffic types of BPA in each benefit function. For example, we see from Figure 10.1(a) that when rectangular function is used, BPA is good at traffic Type 2, Type 3, and Type 4. The corresponding results are listed in Table 10.6 according to the previously listed order.

Figures 10.3 and 10.4 illustrate that BPA always generates the smallest deadline-miss ratio. Among all benefit functions for BPA, linear benefit function gives the smallest deadline-miss ratio. This is reasonable since pseudo-slope of linear benefit function is actually the real slope.

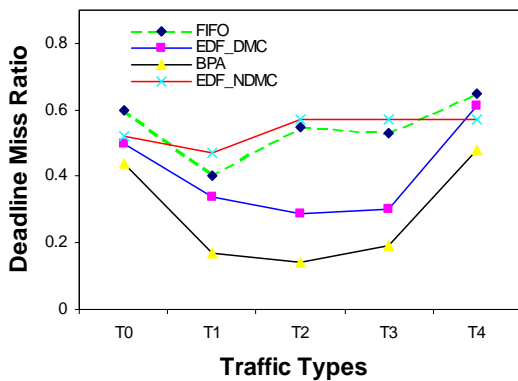


(a) Rectangular Function

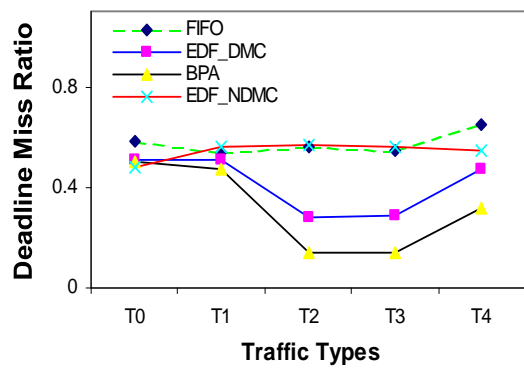


(b) Soft-Rectangular Function

Figure 10.3: Average Deadline Miss Ratio Under 5 Traffic Types for Rectangular and Soft-Rectangular Functions



(a) Linear Function



(b) Quadratic Function

Figure 10.4: Average Deadline Miss Ratio Under 5 Traffic Types for Linear and Quadratic Functions

## 10.4 Performance Comparison Based on Increasing Traffic Arrival Density

All the performance comparisons described so far are based on averaged results from tens of experiments. In order to compare all four algorithms at a much finer level, we plot all individual experiment result points according to the ascending order of traffic density.

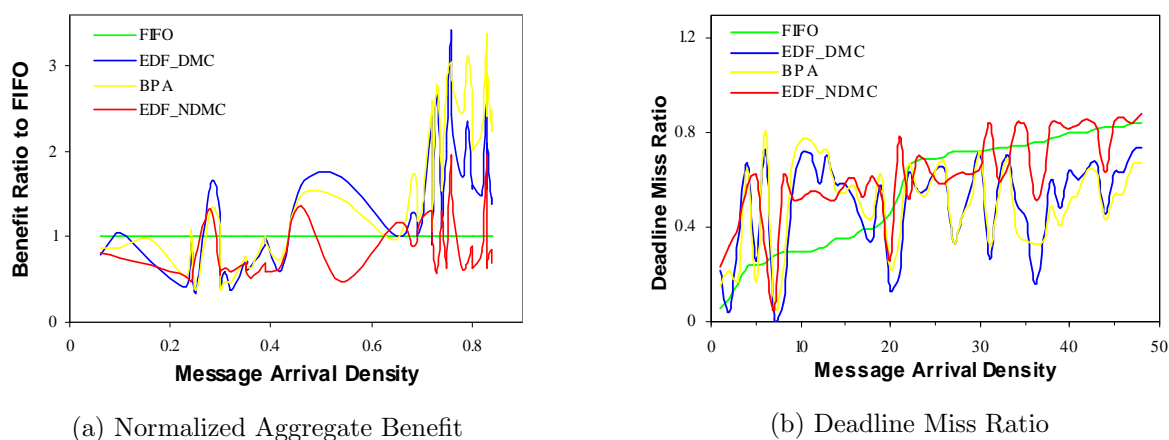
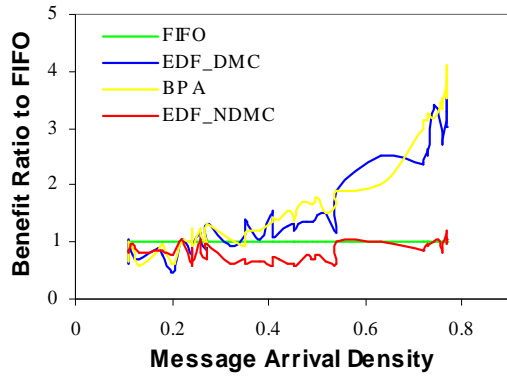


Figure 10.5: Performance Under Traffic Type T0 for Rectangular Benefit Function

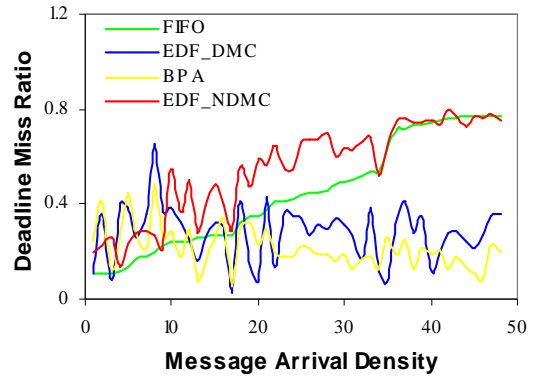
Figures 10.5, 10.6, 10.7, 10.8, and 10.9 illustrate the results for rectangular benefit function under all five traffic types. Results for soft-rectangular, linear, and quadratic benefit functions under all five traffic types are shown in Appendices B.1, B.2, and B.3, respectively.

We draw the following conclusions from Figures 10.5, 10.6, 10.7, 10.8, and 10.9:

First, we conclude that higher the traffic density, the more advantages BPA has over that of FIFO. This conclusion is exactly the same as that obtained from our simulation results. This also means that BPA is good at handling overloaded network situations.

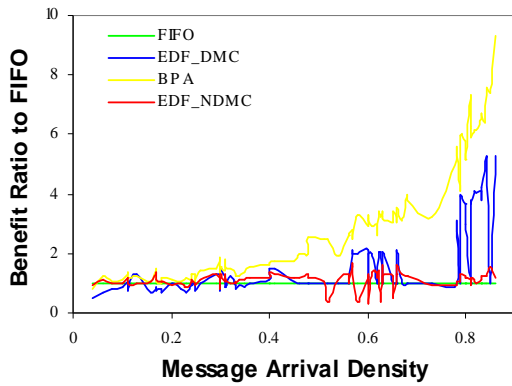


(a) Normalized Aggregate Benefit

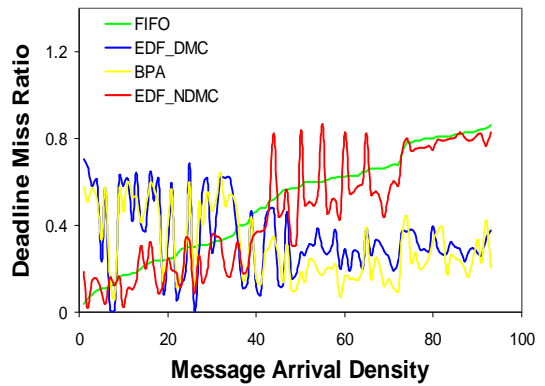


(b) Deadline Miss Ratio

Figure 10.6: Performance Under Traffic Type T1 for Rectangular Benefit Function



(a) Normalized Aggregate Benefit



(b) Deadline Miss Ratio

Figure 10.7: Performance Under Traffic Type T2 for Rectangular Benefit Function

Second, we conclude that when FIFO's deadline-miss ratio is small, BPA's and EDF's deadline-miss ratios are larger (than that of FIFO) due to their higher computational cost. However, when FIFO's deadline-miss ratio is large, the computational overhead of BPA and EDF is completely traded off; thus both BPA and EDF achieve lower deadline-miss ratio than that of FIFO, although the advantage of EDF\_NDMC over FIFO is quite small.

Third, we conclude that BPA achieves the largest final aggregate benefit except in very few special cases. Furthermore, BPA achieves the smallest deadline-miss ratio in all cases.

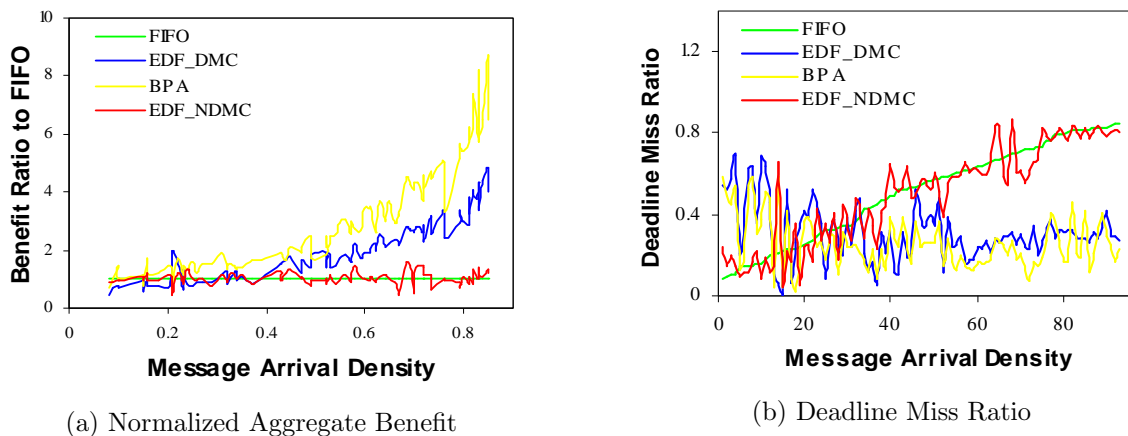
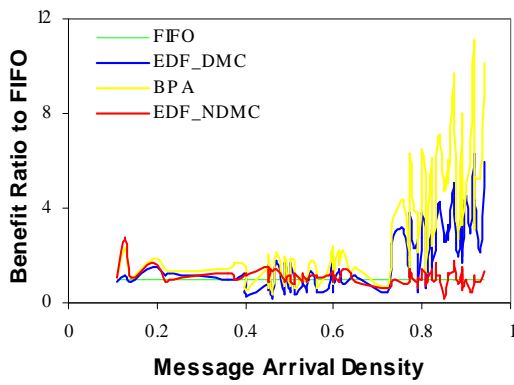


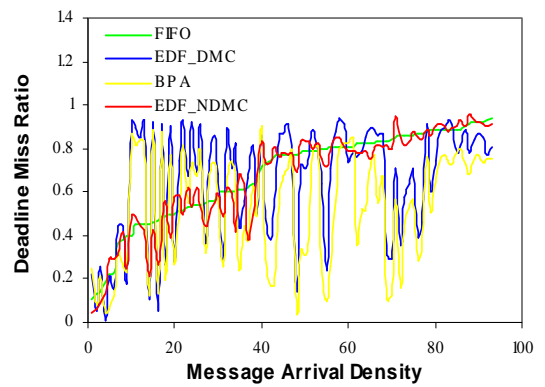
Figure 10.8: Performance Under Traffic Type T3 for Rectangular Benefit Function

Finally, we conclude that when comparing with EDF\_DMC, the final benefit advantage of BPA over EDF\_DMC is significant in some cases such as the ones shown in Figures 10.7, 10.8, and 10.9. However, in some other cases, this advantage is not too significant and in yet other few cases, the advantage even disappears. (We discuss the reasons for this in Section 10.2.)

Thus, the actual performance measurements clearly demonstrate the superiority of the BPA algorithm over the other three algorithms and its strong effectiveness.



(a) Normalized Aggregate Benefit



(b) Deadline Miss Ratio

Figure 10.9: Performance Under Traffic Type T4 for Rectangular Benefit Function

# Chapter 11

## Timeliness Feasibility Conditions

In this chapter, we derive timeliness feasibility conditions of real-time switched Ethernet system solutions to real-time application problems that use the BPA algorithm. We construct timeliness feasibility conditions by schedulability analysis and express them as *non-valued, benefit-accrual predicates*. Such predicates will embody all possible scenarios that can be deployed by “adversaries” that are considered in the design models of a given application problem. The quantification of the feasibility conditions will produce the specification of a system solution that can be certified to exhibit the desired soft timeliness properties. Thus, the timeliness feasibility conditions will facilitate the design of system solutions that guarantee the desired soft timeliness.

To construct the feasibility conditions, we first discuss a generic design problem called the Soft Real-time Packet Transmission (or SRPT) problem. We then consider a solution to this

problem that uses a single-segment real-time switched Ethernet with BPA as the underlying packet scheduling algorithm. We then derive the timeliness feasibility conditions for this solution.

We discuss the SRPT problem and the construction of the feasibility conditions in the sections that follow.

## 11.1 The Soft Real-time Packet Transmission Design Problem

We discuss SRPT by simply sketching the models and properties of the problem. Models are the set of assumptions regarding future operational conditions of the system solution that solves the problem. Properties are the desired services that the system solution that solves the problem must provide to end-users. Models and properties are invariants of the problem.

The SRPT problem is specified as follows:

### 11.1.1 Sketch of Problem Invariant: Models of SRPT

The set of application packets is denoted as  $p_i \in P, i \in [1, n]$ . The size  $n$  of the set  $P$  is unrestricted.

The bit length of a packet  $p_i \in P$  at the data link layer is denoted as  $b(p_i)$ . The physical

framing overheads increase this size into an actual bit length  $b'(p_i) > b(p_i)$  for transmission. Thus, the transmission latency of a packet  $p_i$  is given by  $l_i = b'(p_i)/\psi$ , where  $\psi$  denotes the nominal throughput of the underlying network medium (e.g.,  $10^9$  bits/s for Gigabit Ethernet).

Packets are generated from a set of sources,  $s_i \in S, i \in [1, z]$ . The number of sources  $z$  is unrestricted. The packet subset  $P_j \subseteq P$  is mapped onto source  $s_j \in S, j \in [1, z]$ . The mapping is unrestricted i.e., any packet  $p_j \in P$  can be mapped onto any source  $s_j \in S$ .

The arrival law for packets is the *multimodal arbitrary arrival model* [42]. The multimodal arrival model is a generalization of the *unimodal arbitrary arrival model*. For a packet  $p_i$ , the unimodal arrival model defines the size of a sliding time window  $w(p_i)$  and the maximum number of arrivals  $a(p_i)$  that can occur during any window  $w(p_i)$ . Multimodal arrival model is a finite, ordered sequence of unimodal arrivals. Thus, the sequence  $\langle (a_\pi(p_i), w_\pi(p_i)), \pi \in [1, k] \rangle, i \in [1, n]$  is defined, where the number of sequences  $k$  is unrestricted.

Note that the “adversary” embodied in the multimodal model is stronger—less restricted—than that in the unimodal model, which is stronger than the aperiodic, sporadic, and periodic arrival models. [42]

Each source  $s_i \in S$  is equipped with a clock synchronization module, denoted  $C_i$ . The clock synchronization module generates clock synchronization packets periodically with a period  $\theta$ . Such packets are always inserted at the head of the packet queues and thus transmitted before application packets are transmitted.

The bit length of a clock synchronization packet at the data link layer is a constant and is denoted as  $b_c$ . The physical framing overheads increase this size into an actual bit length  $b'_c > b_c$  for transmission.

### 11.1.2 Sketch of Problem Invariant: Properties of SRPT

The only property of interest is timeliness. The timeliness property is defined using benefit functions and benefit accrual predicates as follows:

Each packet  $p_i \in P$  has a benefit function, denoted as  $B_i$ . Thus, the arrival of a packet  $p_i$  at its destination at a time  $t'$  relative to the release of the packet at its source at an absolute time  $t$  will yield a benefit  $B_i(t + t')$ . All benefit functions  $B_i, i \in [1, n]$  are unimodal and have an initial time, a terminal time, and a deadline time denoted as  $I_i, T_i$ , and  $D_i$ , respectively. Initial times and terminal times are the earliest and latest times for which the benefit function is defined, respectively. A deadline time is the time at which the function drops to a zero value. Furthermore,  $B_i(t) \geq 0, \forall t \in [I_i, D_i], i \in [1, n]$  and  $B_i(t) = 0, \forall t \notin [I_i, D_i], i \in [1, n]$ .

The timeliness property is a desired lower bound on system-wide accrued timeliness benefit denoted  $ATB_l$ , where the system-wide accrued timeliness benefit denoted  $ATB$ , is defined as the sum of the individual benefit accrued by the arrival of packets at their destinations. Thus,  $ATB = \sum_{i=1}^n B_i(t_i) \geq ATB_l$ , where  $t_i$  is the absolute time at which packet  $p_i$  arrives at its destination. The functions  $B_i, i \in [1, n]$  and the value of  $ATB_l$  are unknown.

Thus, the desired timeliness property is a soft timeliness property. Hence, the problem is

called the “Soft Real-time Packet Transmission problem.”

## 11.2 A Solution Using BPA: Construction of Timeliness Feasibility Conditions

To establish the desired timeliness property, we need to determine the worst-case lower bound on the system-wide accrued timeliness benefit that is possible under the models of SRPT. This can be determined by first determining the lower bound on the individual benefit that is accrued by the arrival of any given packet at its destination. The aggregation of the individual benefit lower bounds will yield the lower bound on the system-wide benefit.

To determine the lower bound on individual benefit, we need to determine an upper bound on the packet delay. We thus seek to construct a computable function denoted  $R(s_i, p)$ , that gives an upper bound on the delay incurred by any packet  $p$  to arrive at its destination, since its arrival at the MAC-layer of any source  $s_i, i \in [1, z]$  for outbound transmission. The construction of  $R(s_i, p)$  is similar to that shown in [12].

In a single-segment switched network, a packet will experience contention for *two* network resources once it arrives at the MAC-layer of its source. The resources include (1) the network segment from the source to the switch and (2) the network segment from the switch to the destination. Thus, we define  $R(s_i, p) = R_1(s_i, p) + R_2(p)$ , where  $R_1(s_i, p)$  is the upper bound on the delay incurred by any packet  $p$  to arrive at the switch, since its arrival at the MAC-

layer of any source  $s_i, i \in [1, z]$  and  $R_2(p)$  is the upper bound on the delay incurred by any packet  $p$  to arrive at its destination, since its arrival at the switch.

### 11.2.1 Construction of $R_1(s_i, p)$

Consider a packet  $p$  that arrives at the MAC-layer of a source  $s_i$ . Let  $A(p)$  denote the arrival time of the packet  $p$  at the MAC-layer of the source  $s_i$  and let  $d(p)$  denote its relative deadline (i.e., relative to the arrival time  $A(p)$ ). Let  $I(p)$  denote the time interval  $[A(p), A(p) + d(p)]$ . To determine  $R_1(s_i, p)$ , we need to determine an upper bound on the number of packets belonging to subset  $P_i$  that will be scheduled for outbound transmission (by BPA) on source  $s_i$ , over any interval  $I(p)$ , *before* packet  $p$  is transmitted. We denote this upper bound as  $u_1^i(p)$ .

The bound  $u_1^i(p)$  must be established considering the strongest possible “adversary” that is embodied in the arrival model i.e., assuming that packet arrivals occur at their bounded densities over all time windows over  $I(p)$ .

### 11.2.2 Upper bound $u_1^i(p)$

Upper bound  $u_1^i(p)$  is established by observing that any packet  $q$  will be scheduled by BPA on source  $s_i$  before packet  $p$ , only if packet  $q$  arrives no sooner than  $A(p) - d(q)$  and no later than  $A(p) + d(p) - \frac{b'(p)}{\psi}$ .

This is because, if packet  $q$  were to arrive before  $A(p) - d(q)$ , then its absolute deadline will

occur before the arrival of packet  $p$ . Thus, when packet  $p$  arrives, BPA has either already scheduled packet  $q$  for transmission or has dropped packet  $q$  because it has become infeasible.

Similarly, if packet  $q$  were to arrive after  $A(p) + d(p) - \frac{b'(p)}{\psi}$ , then at that time, BPA would have either already scheduled packet  $p$  for transmission or has dropped  $p$  because it has become infeasible. Note that  $q$  cannot be scheduled before  $p$  once  $p$  has been scheduled and is in transmission (even if  $q$  were to arrive before  $A(p) + d(p)$ ), since packet transmission is non-preemptive.

Note that if packet  $q$  were to arrive after  $A(p) + d(p) - d(q)$  but before  $A(p) + d(p) - \frac{b'(p)}{\psi}$ , the absolute deadline of packet  $q$  will occur after that of  $p$ . Under an EDF packet scheduler, this will cause packet  $q$  to be scheduled after packet  $p$ , since packet  $q$  has a longer absolute deadline than that of  $p$ . However, under BPA, it is quite possible that packet  $q$  can be scheduled before packet  $p$ . Thus, with EDF, the latest arrival time of  $q$  after which  $q$  cannot be scheduled before  $p$  will occur at  $A(p) + d(p) - d(q)$ , whereas with BPA, this will occur at  $A(p) + d(p) - \frac{b'(p)}{\psi}$ .

It follows that:

$$\begin{aligned} u_1^i(p) &= \sum_{q \in P_i} \sum_{\pi=1}^k \left\lceil \frac{\left[ A(p) + d(p) - \frac{b'(p)}{\psi} \right] - [A(p) - d(q)]}{w_\pi(q)} \right\rceil a_\pi(q) + \left\lceil \frac{d(p) - \frac{b'(p)}{\psi}}{\theta} \right\rceil \\ &= \sum_{q \in P_i} \sum_{\pi=1}^k \left\lceil \frac{\left[ d(p) + d(q) - \frac{b'(p)}{\psi} \right]}{w_\pi(q)} \right\rceil a_\pi(q) + \left\lceil \frac{d(p) - \frac{b'(p)}{\psi}}{\theta} \right\rceil. \end{aligned}$$

Now,  $R_1(s_i, p)$  is given by the sum of (1) the time needed to physically transmit  $u_1^i(p)$  packets at throughput  $\psi$  and (2) the upper bounds on aggregate worst-case execution times for BPA

to make scheduling decisions for  $u_1^i(p)$  packets. The worst-case execution time of BPA will depend upon factors such as processor speed and memory access latencies. Given a COTS product that guarantees a worst-case execution time  $\delta_h$  for BPA at an end-host,  $R_1(s_i, p)$  is given by:

$$R_1(s_i, p) = \sum_{q \in P_i} \sum_{\pi=1}^k \left\lceil \frac{[d(p) + d(q) - \frac{b'(p)}{\psi}]}{w_\pi(q)} \right\rceil a_\pi(q) \left[ \frac{b'(q)}{\psi} + \delta_h \right] + \left\lceil \frac{d(p) - \frac{b'(p)}{\psi}}{\theta} \right\rceil \left[ \frac{b'_c}{\psi} + \delta_h \right].$$

### 11.2.3 Construction of $R_2(p)$

$R_2(p)$  is the upper bound on the delay incurred by any packet  $p$  to arrive at its destination, since its arrival at the switch. This upper bound can be established in a manner similar to that of  $R_1(s_i, p)$ . The only difference is that we now need to consider all packets  $q$  that can arrive from all sources  $s_i, i \in [1, z]$  such that they will contend for the same outgoing network segment at the switch as that of packet  $p$ . Since the destination of packets is not specified in  $\langle m.SRPT \rangle$ , we will consider all packets  $q \in P$ .

Consider a packet  $p$  that arrives at the MAC-layer of the switch. Let  $A(p)$  denote the arrival time of  $p$  at the MAC-layer of the switch,  $d(p)$  denote its relative deadline, and  $I(p)$  denote the time interval  $[A(p), A(p) + d(p)]$ . Similar to  $R_1(s_i, p)$ , to determine  $R_2(p)$ , we need to determine an upper bound on the number of packets belonging to set  $P$  that will be scheduled for outbound transmission by BPA on the switch over any interval  $I(p)$ . We denote this upper bound as  $u_2(p)$ .

### 11.2.4 Upper bound $u_2(p)$

Upper bound  $u_2(p)$  is established by observing that any packet  $q$  will be scheduled by BPA on the switch before packet  $p$ , only if packet  $q$  arrives no sooner than  $A(p) - d(q)$  and no later than  $A(p) + d(p) - \frac{b'(p)}{\psi}$ . The rationale for this is exactly the same as that for establishing the bound  $u_1^i(p)$ . It follows that:

$$\begin{aligned} u_2(p) &= \sum_{q \in P} \sum_{\pi=1}^k \left\lceil \frac{[A(p) + d(p) - \frac{b'(p)}{\psi}] - [A(p) - d(q)]}{w_\pi(q)} \right\rceil a_\pi(q) + \left\lceil \frac{d(p) - \frac{b'(p)}{\psi}}{\theta} \right\rceil \\ &= \sum_{q \in P} \sum_{\pi=1}^k \left\lceil \frac{[d(p) + d(q) - \frac{b'(p)}{\psi}]}{w_\pi(q)} \right\rceil a_\pi(q) + \left\lceil \frac{d(p) - \frac{b'(p)}{\psi}}{\theta} \right\rceil. \end{aligned}$$

Now,  $R_2(p)$  is given by the sum of (1) the time needed to physically transmit  $u_2(p)$  packets at throughput  $\psi$  and (2) the upper bounds on aggregate worst-case execution times for BPA to make scheduling decisions for  $u_2(p)$  packets. Again, given a COTS product that guarantees a worst-case execution time  $\delta_s$  for BPA at the switch,  $R_2(p)$  is given by:

$$R_2(p) = \sum_{q \in P} \sum_{\pi=1}^k \left\lceil \frac{[d(p) + d(q) - \frac{b'(p)}{\psi}]}{w_\pi(q)} \right\rceil a_\pi(q) \left[ \frac{b'(q)}{\psi} + \delta_s \right] + \left\lceil \frac{d(p) - \frac{b'(p)}{\psi}}{\theta} \right\rceil \left[ \frac{b'_c}{\psi} + \delta_s \right].$$

### 11.2.5 Feasibility Conditions

The feasibility conditions are therefore given as:

$$ATB = \sum_{i=1}^z \sum_{p_j \in P_i} B_j(R(s_i, p_j)) \geq ATB_l, \text{ where } R(s_i, p_j) = R_1(s_i, p_j) + R_2(p_j).$$

Now, to dimension a system solution to the SRPT problem, an assignment of values to unvalued variables in models of SRPT including number of packets  $n$ , packet sizes  $b(p_i)$ ,  $i \in$

$[1, n]$ , number of sources  $z$ , mappings  $P_j, j \in [1, z]$ , number of arrival sequences  $k$ , and arrival densities  $\langle (a_\pi(p_i), w_\pi(p_i)), \pi \in [1, k] \rangle, i \in [1, n]$  and properties of SRPT including benefit functions  $B_i, i \in [1, n]$  and lower bound on accrued timeliness benefit  $ATB_l$  must be made. The resulting quantified instance of the SRPT problem must then be subject to feasibility analysis using the feasibility conditions presented here. If a feasible solution exists (to the problem instance), a quantified system that will have values assigned to unvalued variables in the solution including network throughput  $\psi$  and BPA's worst-case execution times  $\delta_h$  and  $\delta_s$  can be obtained (using the feasibility conditions).

# Chapter 12

## Conclusions, Limitations and Future Work

In this thesis, we present a MAC-layer packet scheduling algorithm called BPA, for real-time switched Ethernets. BPA considers a message model where application messages have trans-node timeliness requirements that are specified using Jensen's benefit functions. The algorithm seeks to maximize aggregate message benefit by allowing message packets to inherit benefit functions of their parent messages and scheduling packets to maximize aggregate packet-level benefit. Since the packet scheduling problem is NP-hard, BPA heuristically computes schedules with a worst-case cost of  $O(n^2)$ , faster than the  $O(n^3)$  cost of the best known CMA algorithm for the same problem. Our simulation studies show that BPA performs the same or significantly better than CMA for a broad set of benefit functions and significantly outperforms CMA when message arrival density increases.

We also construct a real-time switched Ethernet by implementing BPA in the network protocol stack of the Linux kernel for packet scheduling. Our actual performance measurements using the network implementation reveal the strong effectiveness of the algorithm.

Finally, we derive timeliness feasibility conditions of real-time switched Ethernet systems that use the BPA algorithm. The feasibility conditions allow real-time distributed systems to be constructed (using BPA networks) with guaranteed soft timeliness properties.

Thus, the contribution of the thesis is threefold: (1) the BPA packet scheduling algorithm that seeks to maximize aggregate message benefit in real-time distributed systems; (2) the construction of a soft real-time switched Ethernet network using BPA; and (3) timeliness feasibility conditions for constructing real-time distributed systems using BPA with guaranteed timeliness.

## 12.1 Limitations

As all other packet scheduling algorithms, the effectiveness of BPA is limited to handle the overload situations in all outgoing network queues through an efficient way i.e, to avoid the worst case of system performance like an "insurance policy". It is not meant to guarantee the best network performance which may be achieved by improving the whole network throughput such as increasing bandwidth and switching speed and decreasing processing overhead.

Another limitation of BPA is that queueing is just one part of the entire real-time message delivery process, especially on the host side. Therefore, in order to achieve the best real-time

message delivery performance, we need support from all other real-time mechanisms as well. For instance, process or thread scheduling may need to be based on the real-time constraints of messages associated with each process or thread.

Due to the limitation of experiment conditions, we implemented BPA in a layer under IP and above device driver and checked IP head during packet switching. For practical switch implementations, we believe that it is best to implement BPA at the MAC layer.

## **12.2 Future Work**

Several aspects of this work are directions for interesting future research. For example, the switching speed of the Ethernet switch can be improved by using real-time kernels. This will allow the multi-port Ethernet device driver to work in polling mode, instead of in interrupt-driven mode, which suffers from the interrupt response latency [15].

Extending BPA for multi-segment switched Ethernet networks is a topic that can be immediately explored.

Finally, developing scheduling algorithms for arbitrary unimodal functions and multimodal functions is a challenging research problem.

# Bibliography

- [1] IEEE 802.3ae Task Force (Sub-committee of IEEE 802.3 Ethernet Working Group), “Draft of the 10 gbps ethernet standard,” Draft available from the IEEE, March 2001.
- [2] 10 Gigabit Ethernet Alliance (10GEA), “10 gigabit ethernet technology overview white paper,” <http://www.10gea.org>, September 2001.
- [3] M. Alves, E. Tovar, and F. Vasques, “Ethernet goes real-time: a survey on research and technological developments,” Tech. Rep., The Polytechnic Institute of Porto, Portugal, January 2000.
- [4] C. Venkatramani, *Design, Implementation and Evaluation of RETHER: A Real-Time Ethernet Protocol*, Ph.D. thesis, State University of New York at Stony Brook, December 1996.
- [5] D. Kim, Y. Doh, and Y. Lee, “Table driven proportional access based real-time ethernet for safety-critical real-time systems,” in *Proceedings of IEEE Pacific Rim International Symp. on Dependable Computing*, 2001, pp. 356–363.

- [6] B. Plagemann, “Real time ethernet - live,” [http://ethernet.industrial-networking.com/articles/i01\\_beck\\_rtether.asp](http://ethernet.industrial-networking.com/articles/i01_beck_rtether.asp), 2001.
- [7] J. Kerkes, “Real-time ethernet,” <http://www.embedded.com/story/OEG20010221S0086>, February 2001.
- [8] D. W. Pritty, J. R. Malone, D. N. Smeed, S. K. Banerjee, and N. L. Lawrie, “A real-time upgrade for ethernet based factory networking,” in *Proceedings of 21st IEEE/IECON International Conference on Industrial Electronics, Control, and Instrumentation*, 1995, pp. 1631 –1637.
- [9] SIXNET, “The sixnet industrial ethernet switch,” <http://www.sixnetio.com/>.
- [10] S. K. Kweon and K. G. Shin, “Achieving real-time communication over ethernet with adaptive traffic smoothing,” in *Proceedings of IEEE Real-Time Technology and Applications Symposium*, 2000, pp. 90 –100.
- [11] W. Zhao, J. A. Stankovic, and K. Ramamritham, “A window protocol for transmission of time-constrained messages,” *IEEE Transactions on Computers*, vol. 39, no. 9, pp. 1186 –1203, September 1990.
- [12] J.-F. Hermant and G. Le Lann, “A protocol and correctness proofs for real-time high-performance broadcast networks,” in *Proceedings of the 18th IEEE International Conference on Distributed Computing Systems*, 1998, pp. 360–369.

- [13] W. Kim and J. Srivastava, “New virtual time csma/cd protocols for real-time communication,” in *Proceedings of IEEE Conference on Communications for Distributed Applications and Systems*, 1991, pp. 11–22.
- [14] W. Zhao and K. Ramamritham, “A virtual time csma/cd protocol for hard real-time communication,” in *Proceedings of IEEE Real-Time Systems Symposium*, December 1986, pp. 120–127.
- [15] S. Varadarajan and T. Chiueh, “Ethereal: A host-transparent real-time fast ethernet switch,” in *Proceedings of IEEE International Conference on Network Protocols*, October 1998.
- [16] H. Hoang, M. Jonsson, U. Hagstrom, and A. Kallerdahl, “Switched real-time ethernet and earliest deadline first scheduling-protocols and traffic handling,” in *Proceedings of 10th International Workshop on Parallel and Distributed Real-Time Systems*, April 2002.
- [17] C. Baek-Young, S. Sejun, N. Birch, and J. Huang, “Probabilistic approach to switched ethernet for real-time control applications,” in *Proceedings of 7th IEEE International Conference on Real-Time Computing Systems and Applications*, 2000, pp. 384–388.
- [18] ZNYX Networks, “Multiport ethernet adaptors product zx340q series,” <http://www.znyx.com/products/netblaster/zx340q.htm>.
- [19] H. Zhang, “Service disciplines for guaranteed performance service in packet switching network,” *Proceedings of The IEEE*, vol. 83, no. 10, pp. 1374–1396, October 1995.

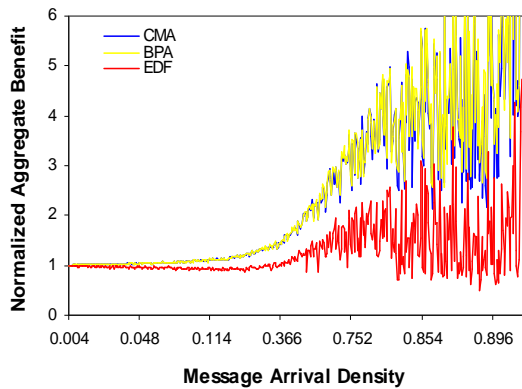
- [20] E. D. Jensen, “Asynchronous decentralized real-time computer systems,” in *Real-Time Computing*, W. A. Halang and A. D. Stoyenko, Eds., Proceedings of the NATO Advanced Study Institute. Springer Verlag, October 1992.
- [21] E. D. Jensen and B. Ravindran, “Guest editor’s introduction to special issue on asynchronous real-time distributed systems,” *IEEE Transactions on Computers*, vol. 51, no. 8, pp. 881–882, August 2002.
- [22] G. Koob, “Quorum,” in *Proceedings of Darpa ITO General PI Meeting*, October 1996, pp. A-59–A-87.
- [23] L. R. Welch, B. Ravindran, B. Shirazi, and C. Bruggeman, “Specification and modeling of dynamic, distributed real-time systems,” in *Proceedings of The IEEE Real-Time Systems Symposium*, December 1998, pp. 72–81.
- [24] R. Clark, E. D. Jensen, A. Kanevsky, J. Maurer, P. Wallace, T. Wheeler, Y. Zhang, D. Wells, T. Lawrence, and P. Hurley, “An adaptive, distributed airborne tracking system,” in *Proceedings of The Seventh IEEE International Workshop on Parallel and Distributed Real-Time Systems*. April 1999, vol. 1586 of *Lecture Notes in Computer Science*, pp. 353–362, Springer-Verlag.
- [25] G. C. Buttazzo, “Chapter 5: Fixed priority servers,” in *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*. Kluwer Academic Publishers, 1997.

- [26] G. C. Buttazzo, “Chapter 6: Dynamic priority servers,” in *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*. Kluwer Academic Publishers, 1997.
- [27] K. Chen and P. Muhlethaler, “A scheduling algorithm for tasks described by time value function,” *Journal of Real-Time Systems*, vol. 10, no. 3, pp. 293–312, May 1996.
- [28] D. L. Mills, “Improved algorithms for synchronizing computer network clocks,” *IEEE/ACM Transactions on Networking*, vol. 3, pp. 245–254, June 1995.
- [29] M. R. Garey, R. E. Tarjan, and G. T. Wilfong, “One-processor scheduling with symmetric earliness and tardiness penalties,” *Math. Oper. Res.*, vol. 13, no. 2, pp. 330–348, May 1988.
- [30] M. Held and R. M. Karp, “A dynamic programming approach to sequencing problems,” *Journal of SIAM*, vol. 10, no. 1, pp. 196–210, March 1962.
- [31] R. K. Clark, *Scheduling Dependent Real-Time Activities*, Ph.D. thesis, Carnegie Mellon University, 1990, CMU-CS-90-155.
- [32] A. Varga, “The objective modular network testbed in c++,” <http://www.hit.bme.hu/phd/vargaa/omnetpp.htm>.
- [33] J. W. S. Liu, *Real-Time Systems*, Prentice Hall, New Jersey, 2000.
- [34] D. Mills, “xntpd - network time protocol (ntp) daemon,” [http://www.eecis.udel.edu/~ntp/database/html\\\_xntp3-5.90/xntpd.html](http://www.eecis.udel.edu/~ntp/database/html\_xntp3-5.90/xntpd.html).

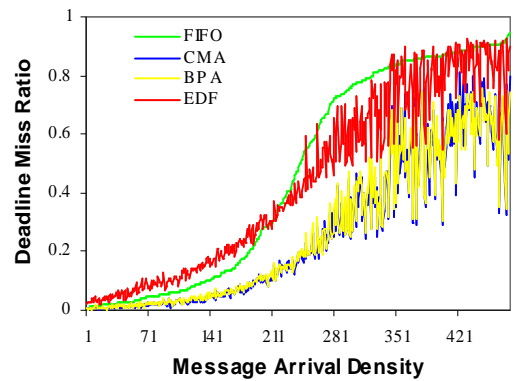
- [35] S. Radhakrishnan, “Linux – advanced networking overview version 1,” <http://qos.ittc.ukans.edu/howto/howto.html>.
- [36] W. Almesberger, “Linux network traffic control-implementation overview,” <http://diffserv.sourceforge.net/#doc>, April 1999.
- [37] U. Böhme and L. Buytenhenk, “Linux bridge-stp-howto,” <http://www.tldp.org/HOWTO/BRIDGE-STP-HOWTO>, January 2001.
- [38] U. Böhme and L. Buytenhenk, “Linux bridge source code,” <http://bridge.sourceforge.net/>, Nov. 2001.
- [39] W. R. Stevens, *UNIX Network Programming Networking APIs: Sockets and XTI*, Prentice Hall, Upper Saddle River, New Jersey 07458, 1998.
- [40] M. Beck, H. Bohme, M. Dziaszka, U. Kunitz, R. Magnus, and D. Verworner, *Linux Kernel Internals*, Addison Wesley Longman, 1998.
- [41] A. Rubini, Ed., *Linux Device Drivers*, O’Reilly, 1998.
- [42] G. Le Lann, “Proof-based system engineering and embedded systems,” in *Lecture Notes in Computer Science*, G. Rozenberg and F. Vaandrager, Eds., vol. 1494, pp. 208–248. Springer-Verlag, October 1998.

# Appendix A

## Simulation Results for Different Benefit Functions

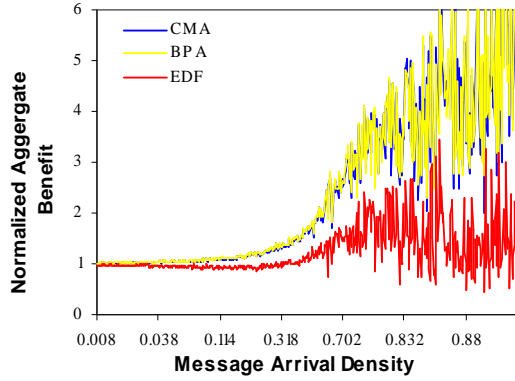


(a) Normalized Aggregate Benefit

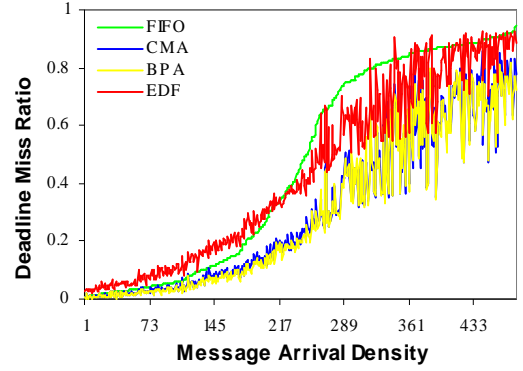


(b) Deadline Miss Ratio

Figure A.1: Performance Under Increasing Message Arrival Density and *Composite* Benefit Functions

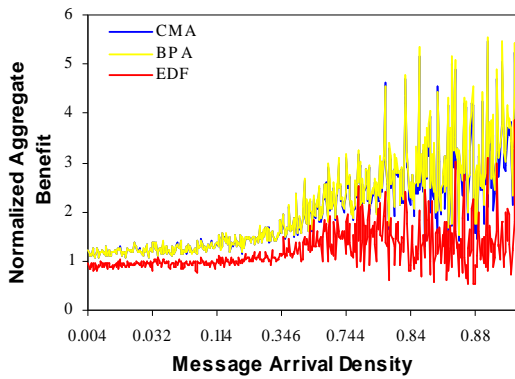


(a) Normalized Aggregate Benefit

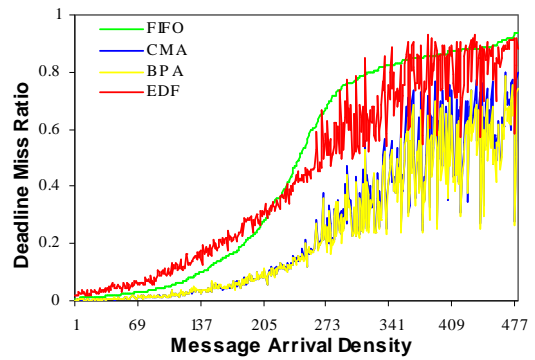


(b) Deadline Miss Ratio

Figure A.2: Performance Under Increasing Message Arrival Density and *Exponential* Benefit Functions

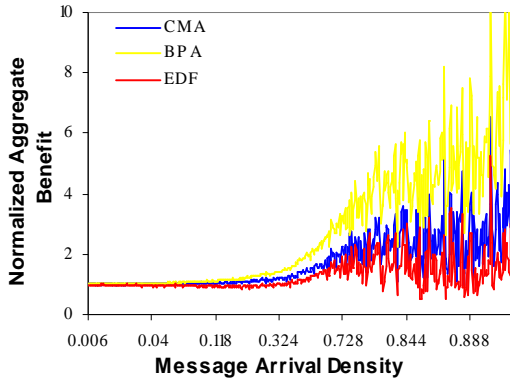


(a) Normalized Aggregate Benefit

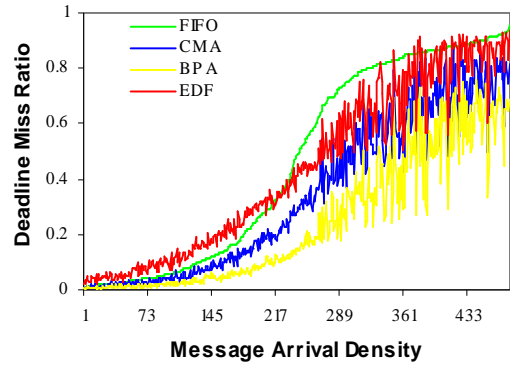


(b) Deadline Miss Ratio

Figure A.3: Performance Under Increasing Message Arrival Density and *Linear* Benefit Functions

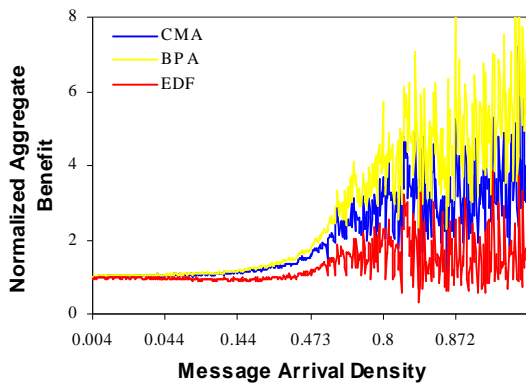


(a) Normalized Aggregate Benefit

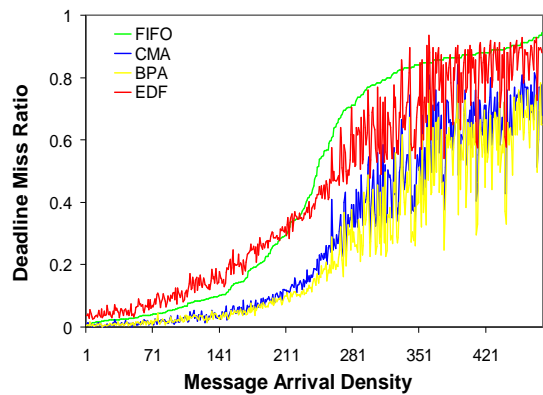


(b) Deadline Miss Ratio

Figure A.4: Performance Under Increasing Message Arrival Density and *Rectangular* Benefit Functions



(a) Normalized Aggregate Benefit



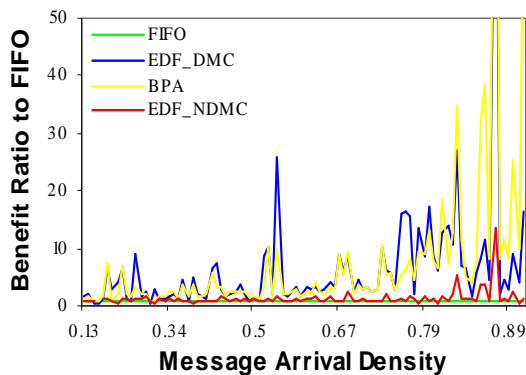
(b) Deadline Miss Ratio

Figure A.5: Performance Under Increasing Message Arrival Density and *Soft-Rectangular* Benefit Functions

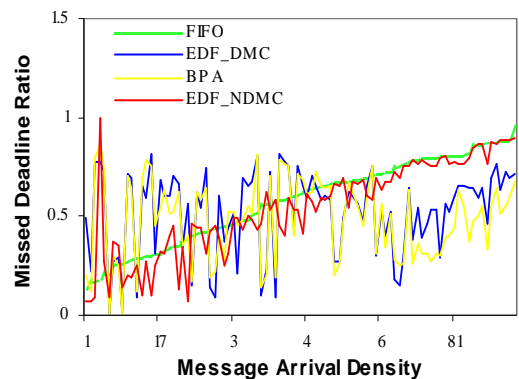
# Appendix B

## Implementation Results for Different Benefit Functions

### B.1 *Soft-Rectangular* Benefit Function

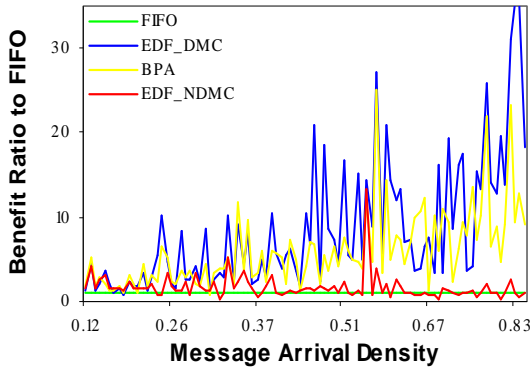


(a) Normalized Aggregate Benefit

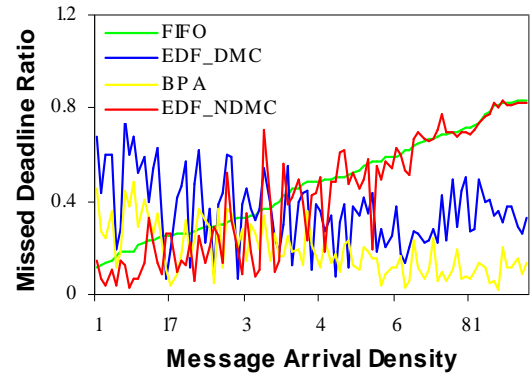


(b) Deadline Miss Ratio

Figure B.1: Performance Under Traffic Type T0

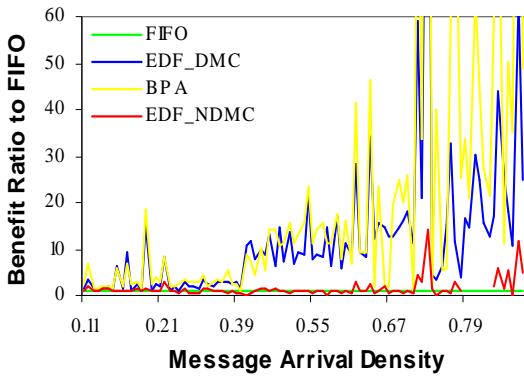


(a) Normalized Aggregate Benefit

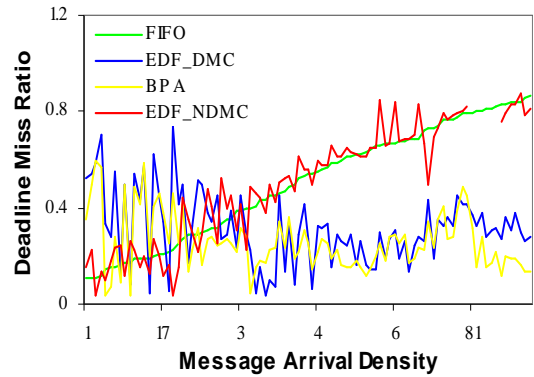


(b) Deadline Miss Ratio

Figure B.2: Performance Under Traffic Type T1

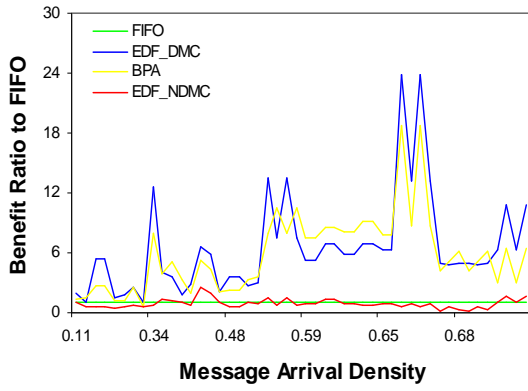


(a) Normalized Aggregate Benefit

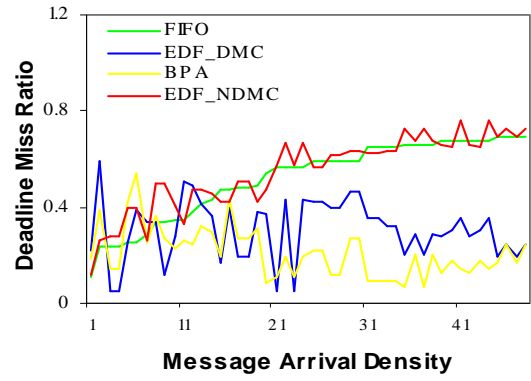


(b) Deadline Miss Ratio

Figure B.3: Performance Under Traffic Type T2

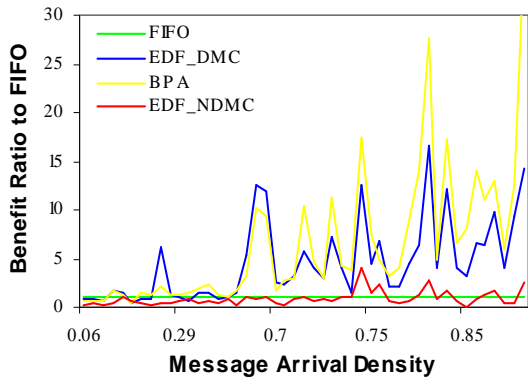


(a) Normalized Aggregate Benefit

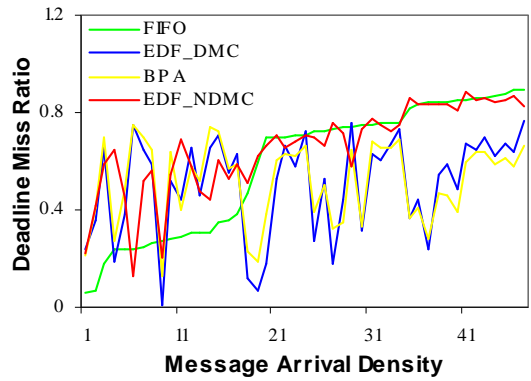


(b) Deadline Miss Ratio

Figure B.4: Performance Under Traffic Type T3



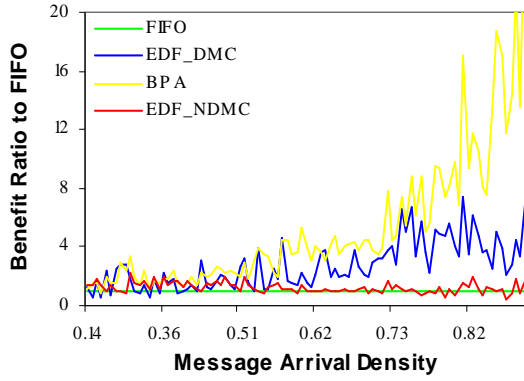
(a) Normalized Aggregate Benefit



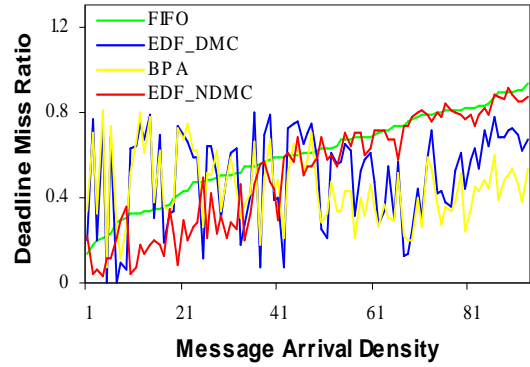
(b) Deadline Miss Ratio

Figure B.5: Performance Under Traffic Type T4

## B.2 Linear Benefit Function

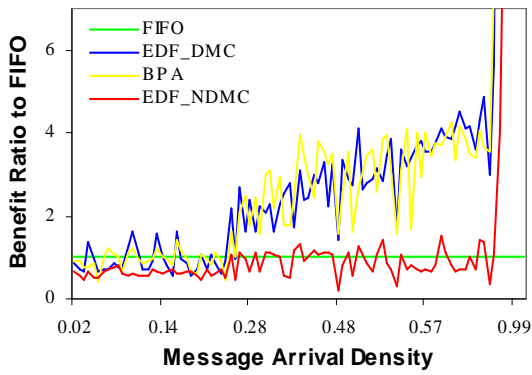


(a) Normalized Aggregate Benefit

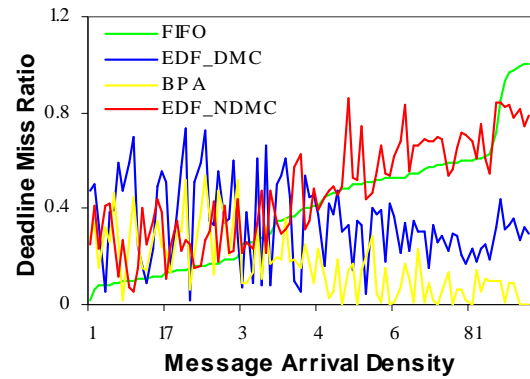


(b) Deadline Miss Ratio

Figure B.6: Performance Under Traffic Type T0

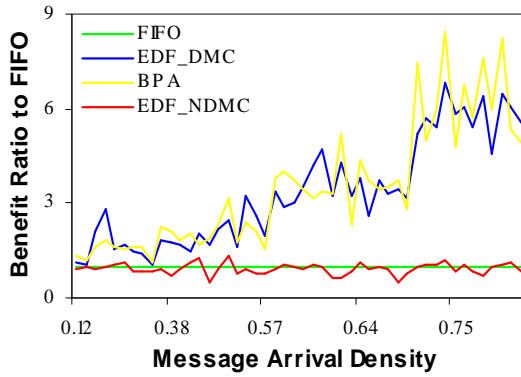


(a) Normalized Aggregate Benefit

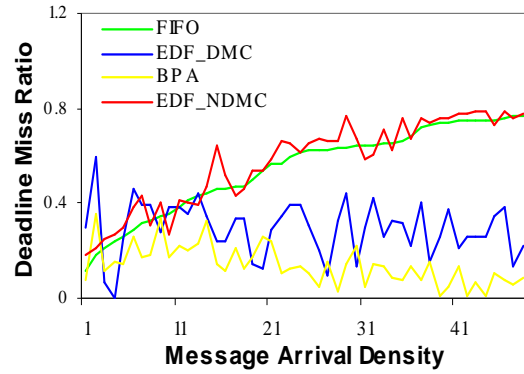


(b) Deadline Miss Ratio

Figure B.7: Performance Under Traffic Type T1

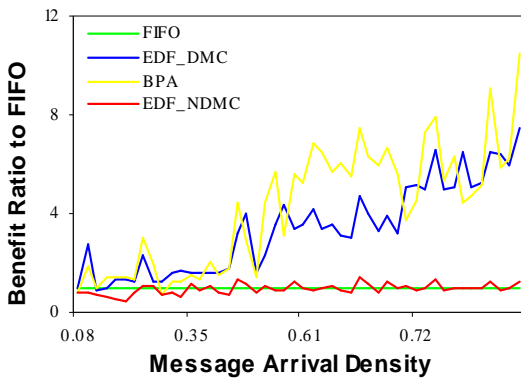


(a) Normalized Aggregate Benefit

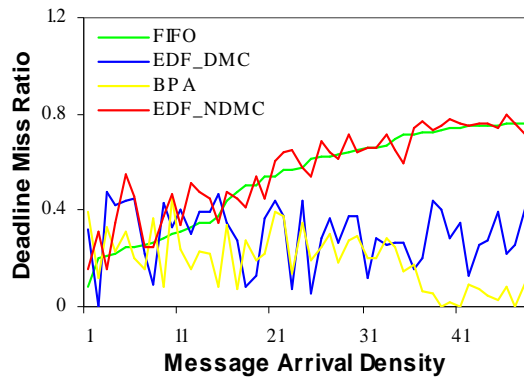


(b) Deadline Miss Ratio

Figure B.8: Performance Under Traffic Type T2

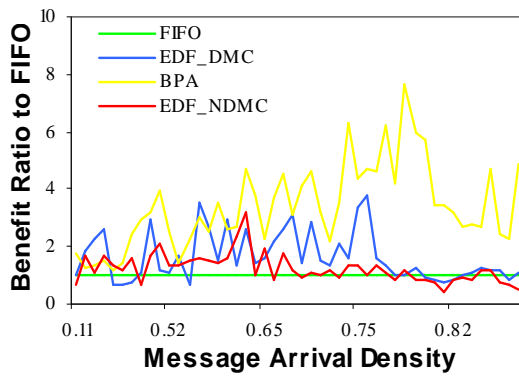


(a) Normalized Aggregate Benefit

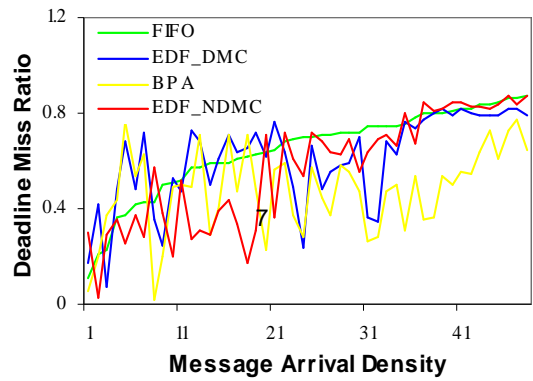


(b) Deadline Miss Ratio

Figure B.9: Performance Under Traffic Type T3



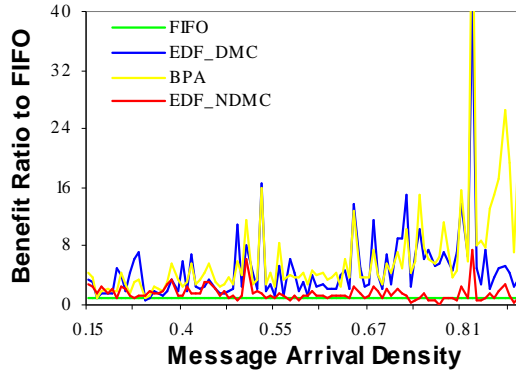
(a) Normalized Aggregate Benefit



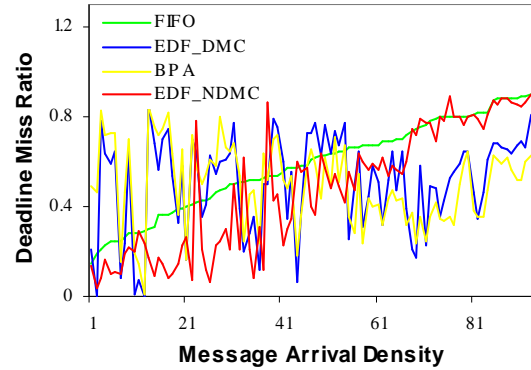
(b) Deadline Miss Ratio

Figure B.10: Performance Under Traffic Type T4

### B.3 Quadratic Benefit Function

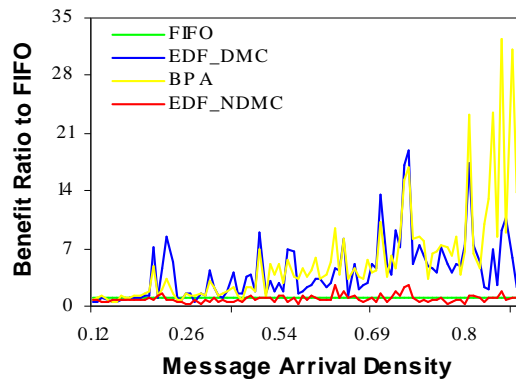


(a) Normalized Aggregate Benefit

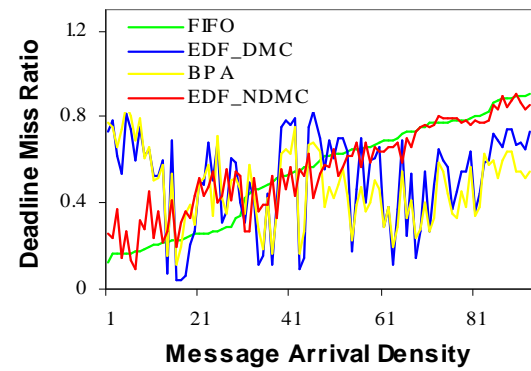


(b) Deadline Miss Ratio

Figure B.11: Performance Under Traffic Type T0

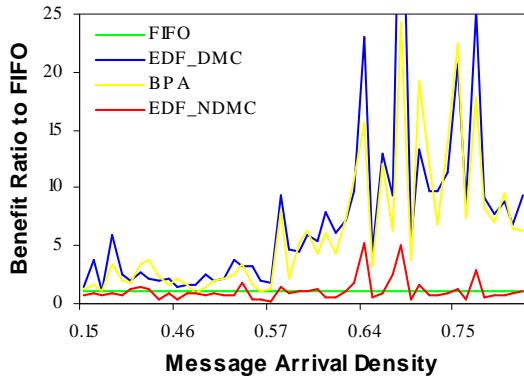


(a) Normalized Aggregate Benefit

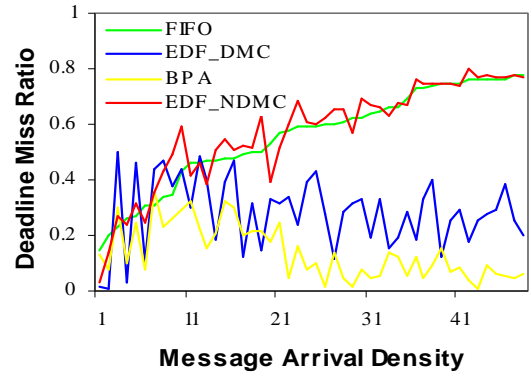


(b) Deadline Miss Ratio

Figure B.12: Performance Under Traffic Type T1

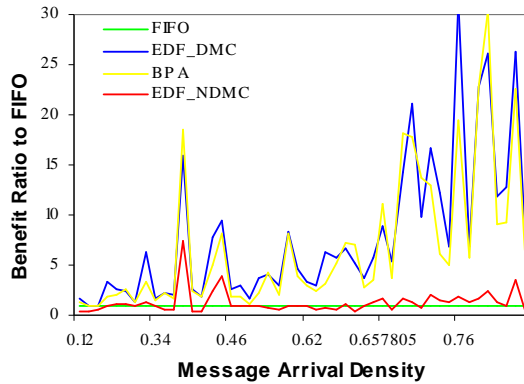


(a) Normalized Aggregate Benefit

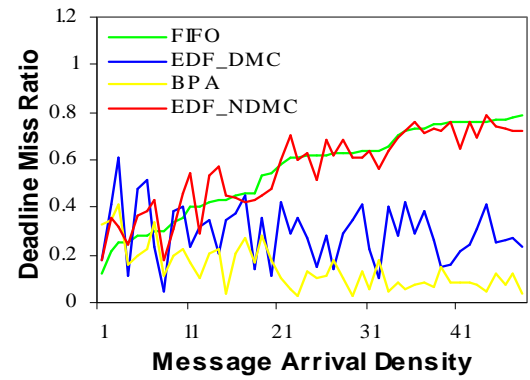


(b) Deadline Miss Ratio

Figure B.13: Performance Under Traffic Type T2

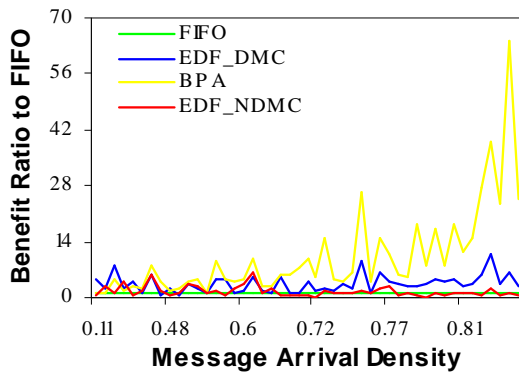


(a) Normalized Aggregate Benefit

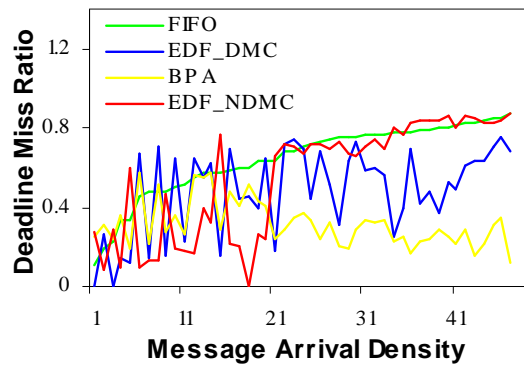


(b) Deadline Miss Ratio

Figure B.14: Performance Under Traffic Type T3



(a) Normalized Aggregate Benefit



(b) Deadline Miss Ratio

Figure B.15: Performance Under Traffic Type T4

\*

# Appendix C

## Vita

Mr.Wang was born in Qiyang County, Hunan Province, China. In 1992, he got his Bachelor of Science degree in Electrical Engineering from Beijing University of Aeronautics and Astronautics (BUAA) in Beijing, China. Then he served as a computer engineer in Juko Electrical Co. and Industrial and Commercial Bank of China, Huizhou Branch for several years. His major responsibilities included designing and maintaining computer hardware, designing application software for credit card business management, POS, ATM machines in banking services. His research interests are in real-time distributed operating system design, power management in hardware/software codesign, VLSI testing and verification. His other interests include soccer, travel and making more friends all over the world.