



Norwegian University of  
Science and Technology

# Performance Analysis of the Dropping Scheme of DMP Network Nodes

Kaiyu Dai

Master of Science in Communication Technology

Submission date: June 2008

Supervisor: Leif Arne Rønningen, ITEM

Co-supervisor: Lena Wosinska, Royal Institute of Technonoly



# Problem Description

The Distributed Multimedia Play (DMP) is new generation of multimedia communication systems which provides near-natural virtual collaborations for long-distance users. To approach near-natural perceptions, DMP produces huge amount of traffics and requires very restrict delay of the Multimedia Content packets. If the traffic of a network node of DMP overloads, this node has to drop some packets selectively. To ensure the graceful degradation of quality, a special dropping scheme is designed particularly for the DMP network nodes.

The goal of this project is to analyze the performance of the DMP nodes with the special dropping scheme, give simulation and mathematic models to analyze the delay and loss rate of the packets passing through DMP network nodes, and give suggestions and proposals to improve the performance of DMP network nodes.

Assignment given: 15. January 2008  
Supervisor: Leif Arne Rønningen, ITEM



# Preface

This report is the result of my Master's thesis carried out at the Department of Telematics at The Norwegian University of Science and Technology (NTNU).

I would like to thank my main supervisor, Professor Leif Arne Rønningen for giving me guidelines on how to approach the problem statement and giving me all the helpful information of DMP system and motivation during the process. The structure of this thesis is also proposed by Professor Rønningen and thanks to his carefully review of my report.

I would also thank my supervisor in KTH, Lena Wosinska for the careful review of the analytic models part of the thesis and giving me many helpful comments and suggestions.

Mei Li in New Mexico State University deserves thanks for helping me dealing with the mathematics problems about the Non-homogeneous Poisson Process and Doc Arne Lie deserves thank for giving me suggestion about the simulation and mathematic tools.

Finally, I would like to thank my friends and fellow students at the NordSecMob program for great days and valuable discussions, and I thank my parents for giving me life and support my education.

*Trendheim, June 26, 2008*

*Kaiyu Dai*

# Abstract

The Distributed Multimedia Play (DMP) is new generation of multimedia communication systems which provides near-natural virtual collaborations for long-distance users. To approach near-natural perceptions, DMP produces huge amount of traffics and requires very restrict delay of the Multimedia Content packets. If the traffic of a network node of DMP overloads, this node has to drop some packets selectively. To ensure the graceful degradation of quality, a special dropping scheme is designed particularly for the DMP network nodes. This thesis analyzes the performance of the DMP nodes with the special dropping scheme.

This thesis focuses on building analytic models to calculate the delay and loss rate of the traffics of the DMP nodes. Two simulators are developed to verify the analytic models. Several experiments are carried out on the two simulators to test the performance of the network nodes in different conditions. More specifically, the thesis work will be aiming at the following objectives:

1. To provide a concise and clear introduction to DMP and its special dropping scheme.
2. To describe the simulation models of the two simulators, explaining their functions and the input and output of them.
3. To build analytic models to calculate the delay and the loss rate of the packets passing through one DMP network node or passing through several network nodes of a DMP collaboration.
4. To provide some suggestion about how to improve the performance of the DMP networks.

The main contributions of this Master's thesis are the two simulators and the analytic models. Three analytic models are proposed to calculate the delay and loss rate of one DMP network node. They are M/D/1 with infinite queue model, M/D/1 with dynamic queue model and M/B/1 with dynamic queue model. The calculation results of those models are consistent well with the simulation results in the experiments. The analytic models can also be applied to universal queuing systems in which dropping queue and normal queue converge.

---

# Contents

PREFACE .....	I
ABSTRACT.....	II
LIST OF FIGURES .....	VIII
LIST OF TABLES.....	X
LIST OF ABBREVIATIONS.....	XI
CHAPTER 1	
INTRODUCTION.....	1
1.1 Motivation.....	1
1.2 Scope.....	2
1.3 Research Methodology .....	2
1.4 Reference Comments .....	3
1.5 Related works.....	3
1.6 Report Outline.....	4
CHAPTER 2	
TECHNICAL BACKGROUND.....	5
2.1 Distributed Multimedia Plays .....	5
2.1.1 Quality Requirement of DMP .....	6
2.1.2 Quality Shaping .....	6
2.1.3 SceneProfiles and the NOC Scheme .....	7



2.2 AppTraNet protocol ..... 8  
2.2.1 The AppTraNet packet header ..... 9  
2.2.2 Packet types, priority and the dropping scheme ..... 10

CHAPTER 3

THE SIMULATORS ..... 13

3.1 System requirements ..... 13  
3.1.1 Hardware requirements ..... 13  
3.1.2 Software preparation ..... 14

3.2 Model Explanation ..... 14  
3.2.1 ENTITY and WAITQ ..... 14  
3.2.2 The model of simulator *NodeSim* ..... 15  
3.2.2.1 Packets Generators ..... 16  
3.2.2.2 Sel Box ..... 16  
3.2.2.3 Output component H ..... 17  
3.2.2.4 *packetCTRL* ..... 18  
3.2.2.5 *packetNOC* ..... 18  
3.2.3 The Model of Simulator *RouteSim* ..... 19  
3.2.3.1 *DMP node* ..... 21  
3.2.3.2 Packets Generators ..... 21  
3.2.3.3 *Router packetCTRL* ..... 22  
3.2.3.4 *Router packetNOC* ..... 22  
3.2.3.5 *Source packetCTRL* ..... 23  
3.2.3.6 *Source packetNOC* ..... 24

3.4 Output of the Simulators ..... 26  
3.4.1 Output of *NodeSim* ..... 26  
3.4.2 Output of *RouteSim* ..... 26

CHAPTER 4

ANALYTIC MODELS FOR ONE DMP NETWORK NODE ..... 27

4.1 Preliminaries ..... 27  
4.1.1 The Output link queuing system of the AppTraNet protocol ..... 28  
4.1.2. Definitions and Assumptions ..... 29

4.2 The Average Waiting Time of the Control Packets ..... 30  
4.2.1 Low Traffic Rate ..... 31  
4.2.2 High Traffic Rate ..... 31

4.3 The Average Waiting Time and Loss Rate of the NOC Packets ..... 32



4.3.1 The model M/D/1 with dynamic queue.....	33
4.3.2 The Model M/B/1 with Dynamic Queue.....	38
4.4 Verify the analytic models by <i>NodeSim</i> .....	40
CHAPTER 5	
ANALYTIC MODEL FOR A .....	47
MULTI-COLLABORATION.....	47
5.1 Establishment of a Multiparty DMP Collaboration .....	47
5.1.1 DMP Network Topologies .....	47
5.1.2 ServiceControl and Security of DMP collaborations .....	49
5.1.3 Establishment of a Multiparty DMP Collaboration.....	49
5.2 Analytic Models .....	50
5.2.1 Delay of the NOC packets .....	50
5.2.2 Delay of the Control packets .....	51
5.2.3 The Loss Rate of the NOC packets .....	51
5.3 Experiments and Discussion .....	52
5.3.1 Experiment 1: Delay and Loss Rate .....	52
CHAPTER 6	
GUARANTEED DELAY AND RELIABLE DELIVERY USING SUB-OBJ SEQ. NO 9 .....	57
6.1 Introduction of NOC .....	57
6.2 An Updated Version of Output Link Queuing System.....	58
6.3 Analytic Model.....	59
6.3.1 Delay Analysis.....	59
6.3.2 Loss Rate .....	60
CHAPTER 7	
IMPROVE THE PERFORMANCE.....	61
7.1 Overview of the Delays .....	61
7.2 Faster than Light .....	62
7.3 Parallel Processors .....	63



7.4 Powerful Routers .....	64
CHAPTER 8	
CONCLUSIONS AND THE FUTURE WORK .....	65
8.1 Conclusion .....	65
8.2 Future work.....	66
REFERENCE .....	67
APPENDIX A	
USING THE SIMULATOR.....	71
A.1 Compiling.....	71
A.2 Define the parameters of <i>NodeSim</i> .....	72
A.3 Define the parameters of <i>RouteSim</i> .....	72
A.4 Execute the simulators .....	73
A.5 Output of <i>NodeSim</i> .....	73
A.6 Output of <i>RouteSim</i> .....	75
APPENDIX B	
DISCUSSION OF A SPECIAL CASE OF NON-HOMOGENEOUS POISSON PROCESS .....	79
B.1 mathematical proof.....	79
B.2 Verify by simulation program.....	83
APPENDIX C.....	87
THE MATLAB SCRIPTS .....	87
C.1 <i>newm7_plot.m</i> .....	87
C.2 <i>newm7_plot_md1.m</i> .....	93
C.3 <i>select.m</i> .....	98

---

<i>C.4 plotm.m</i> .....	98
<i>C.5 plotm_lr.m</i> .....	98
<i>C.6 Poissonstest.m</i> .....	99
APPENDIX D	
THE SCRIPTS OF THE SIMULATORS .....	101

# List of Figures

Figure 2-1: A work lunch as DMP near natural collaboration .....	6
Figure 2-2: object oriented scenes with sub-objects.....	7
Figure 2-3: Sub-object allocation, 9 sub-objects .....	8
Figure 2-4: Three-layer architecture, Source Access Node, Network Node, and Destination Access Node .....	9
Figure 2-5: Dropping and prioritizing of packets in network nodes, AppTraNet layer. Hardware architecture .....	11
Figure 3-1: the idea of waiting in queue .....	15
Figure 3-2: The Dropping Scheme of WAITQ .....	15
Figure 3-3: the components of <i>NodeSim</i> simulator.....	16
Figure 3-4 Packets Generator generates packets periodically .....	16
Figure 3-5: <i>Sel Box</i> fetches packets from Q2 and Q1 and forwards them to Q0.....	17
Figure 3-6: <i>H</i> fetches packets from Q0 and send them out periodically.....	18
Figure 3-7: The lifecycle of a Control packet in <i>NodeSim</i> .....	18
Figure 3-8: the lifecycle of a NOC packet in <i>NodeSim</i> .....	19
Figure 3-9: Simulation model of <i>RouteSim</i> .....	20
Figure 3-10: the components of <i>RouteSim</i> .....	21
Figure 3-11: the components in <i>DMP node</i> .....	21
Figure 3-12: the lifecycle of a <i>Router packetCTRL</i> .....	22
Figure 3-13: the lifecycle of a <i>Router packetNOC</i> .....	23
Figure 3-14: lifecycle of a <i>Source packetCTRL</i> .....	24
Figure 3-15: the lifecycle of a <i>Source packetNOC</i> .....	25
Figure 4-1: the output link queuing system of DMP node [1].....	29
Figure 4-2: comparison of the average waiting time of M/D/1 with infinite queue model and the simulation result .....	41
Figure 4-3: comparison of the average waiting time from M/D/1 with dynamic queue model, M/B/1 with dynamic queue model, and from the simulation application, while $N_{Q0}=1000, p1=0.7$ . .....	42
Figure 4-4: Comparison of the loss rate from M/D/1 with dynamic queue model, M/B/1 with dynamic queue model, and from the simulation application. $N_{Q0}=1000, p1=0.7$ .....	42
Figure 4-5: comparison of the average waiting time from M/D/1 with dynamic queue model, M/B/1 with dynamic queue model, and from the simulation, while $N_{Q0}=1500, p1=0.7$ .....	43
Figure 4-6: Comparison of the loss rate from M/D/1 with dynamic queue model,	

M/B/1 with dynamic queue model, and from <i>NodeSim</i> , while $N_{Q0}=1500$ , $p1=0.7$ .....	44
Figure 4-7: Comparison of the average waiting time from M/D/1 with dynamic queue model, M/B/1 with dynamic queue model, and from the <i>NodeSim</i> , while $N_{Q0}=1000$ , $p1=0.5$ .....	44
Figure 4-8: Comparison of the loss rate from M/D/1 with dynamic queue model, M/B/1 with dynamic queue model, and from <i>NodeSim</i> , while $N_{Q0}=1000$ , $p1=0.5$ ; .....	45
Figure 5-1: Network between three European cities.....	48
Figure 5-2: a one-way collaboration with 5 network nodes .....	52
Figure 5-3: the mean delay of the NOC packets in different traffic interests.....	53
Figure 5-4: the loss rate of the NOC packets in different traffic interests.....	54
Figure 5-5: The mean delay against the link capacity when the traffic load is 100% ..55	
Figure 5-6: The loss rate against the link capacity when the traffic load is 100% .....	56
Figure 6-1: An updated version of output link queuing system of DMP node .....	59
Figure 7-1: parallel processors in the user equipment of end node .....	63
Figure 8-1: Queuing systems that the analytic models can be applied .....	66
Figure A-1: Compile the two simulators.....	71
Figure A-2: Input_ <i>NodeSim</i> .txt .....	72
Figure A-3: Input_ <i>NodeSim</i> .txt .....	72
Figure A-4: Executing <i>NodeSim</i> and <i>RouteSim</i> .....	73
Figure A-5: <i>NodeSim_delay_NOC.txt</i> .....	73
Figure A-6: <i>NodeSim_delay_NOC.txt</i> .....	74
Figure A-7: <i>NodeSim_overview.txt</i> .....	75
Figure A-8: <i>NodeSim_drop.txt</i> .....	75
Figure A-9: <i>RouteSim_delay_Ctrl.txt</i> .....	76
Figure A-10: <i>RouteSim_delay_NOC.txt</i> .....	76
Figure A-11: <i>RouteSim_total.txt</i> .....	77
Figure A-12: <i>RouteSim_node1_delay.txt</i> .....	77
Figure A-13: <i>RouteSim_node1_drop.txt</i> .....	78
Figure A-14: <i>RouteSim_ori_node1_drop.txt</i> .....	78

## List of tables

Table 5-1: The mean delay and drop rate in different traffic interest of network nodes .....	53
Table 5-2: The mean delay and drop rate in different link capacity .....	55
Table B-1: Comparison of results from Equation B-12 and the simulation 1.....	84
Table B-2: Comparison of results from Equation B-12 and the simulation 2.....	84
Table B-3: Comparison of results from Equation B-12 and the simulation 3.....	85

## List of Abbreviations

DMP: distributed multimedia play  
HMS: Multimedia Home Space  
DEMOS: Discrete Event Modeling On Simula  
NOC: near-natural coding  
AV: Audio-Visual  
PNG: Portable Network Graphics  
VASARI: the Visual Arts System for Archiving and Retrieval of Images  
NodeSim: Node Simulator  
RouteSim: Route Simulator  
AH: Authentication Header  
ESP: Encapsulating Security Payload  
IKE: Internet Key Exchange  
Mbps: millions of bits per second  
ASIC: Application Specific Integrate Circuit





# Chapter 1

## Introduction

### 1.1 Motivation

The Distributed Multimedia Play (DMP) is new generation of multimedia communication systems which provides near-natural virtual collaborations for long-distance users. [1]

The DMP architecture is designed to handle Multimedia Home Space (MHS) distributed services and other services that may be introduced in a fifteen years' time. MHS can be any room in a single family house or an apartment. It can be a living room or kitchen, or a specialized room built for near-natural virtual collaborations. From those spaces the users can participate in any networked or local collaboration, private or public, with other people, or with servers.

The DMP system is expected to be implemented in ten or fifteen years. Nowadays, the researches on DMP mainly focus on its network infrastructure and multimedia coding schemes rather than the technical details of the implementation.

The AppTraNet protocol is the protocol specially deigned to handle the network traffics of the application layer and network layer of the DMP networks. To approach near-natural perceptions, DMP produces huge amount of traffics and requires very restrict delay of the multimedia content packets. If the traffic of a network node of DMP overloads, this node has to drop some packets selectively. To ensure the graceful degradation of quality, a special dropping scheme is designed particularly for the DMP network nodes implementing the AppTraNet protocol. This thesis analyzes the performance of the DMP nodes with the special dropping scheme. The research of this thesis focuses on analyzing the delay and loss rate of the traffics of the DMP nodes by analytic models and simulators.

## 1.2 Scope

The title of this Master's thesis is "Performance Analysis of the Dropping Scheme of DMP Network Nodes". This thesis focuses on research the delay and loss rate of the packets passing through the DMP network node. Delay and loss rate are important factors in evaluating the performance a network system.

This thesis analyzes the traffics of the DMP nodes in which the special dropping scheme is implemented. Three Analytic models are built to calculate the average waiting time and loss rate of the packets passing through a DMP network node. The delay and total loss rate of a DMP collaboration are also analyzed. Two simulators are programmed to measure the delay and loss rate in different situations. The output from the simulators will be compared with the calculation results from the analytic models.

The analytic models and simulators are all built under some simplified assumptions. The traffic is assumed to be generated from Poisson Process. Other interarrival distributions can be simulated but they are too complicated to be calculated from the mathematical models.

Some proposals to improve the performance of the network infrastructure are discussed in the Master's thesis. However, those are not the emphasis of this project, and the detailed exploration of those technologies is out of the scope of this thesis.

## 1.3 Research Methodology

The emphases of this thesis are the two simulators and three analytic models. The mathematical models are built mainly based on Queuing Theory. The calculations based on the analytic models are carried out on Matlab 7.0. The programming language SIMULA with the DEMOS (Discrete Event Modelling on Simula) is chosen as the programming language of the simulators in this thesis.

Queuing Theory is the mathematical study of waiting lines. [2]The theory enables mathematical analysis of several related processes, including arriving at the back of the queue, waiting in the queue (essentially a storage process), and being served by the server(s) at the front of the queue. The theory permits the derivation and calculation of several performance measures including the average waiting time in the queue or the system, the expected number waiting or receiving service and the probability of encountering the system in certain states, such as empty, full, having an available server or having to wait a certain time to be served. In the thesis, the analytic models are designed to calculate the average waiting time and loss rate of the packets entering a DMP network node, and Queuing Theory is the right tool to construct those

models. The books [19], [28], and [29], and the papers [18], [30], [31] and [32] give me a lot of instructions in the analysis part of this thesis and we utilize several conclusions from those books and papers.

Matlab is a high-level language and interactive environment that enables us to perform computationally intensive tasks faster than with traditional programming languages such as C, C++, and Fortran. [33] The analytic models consist of many equations. The calculations based on the analytic models are complicated and the environment Matlab 7.0 is chosen to perform the calculations.

SIMULA is a simpler and earlier version of object oriented programming languages. It was developed for discrete event simulation, but was later expanded and reimplemented as a full scale general purpose programming language. With the DEMOS add on package SIMULA becomes a good language for running discrete event models. [3]

DEMOS extends the standard context of SIMULA by a few concepts which provide a standardized approach to a wide range of discrete event problems. The basic concept is the ENTITY for mirroring major dynamic model components. In addition, DEMOS automates as much as possible and provides event tracing to help in model validation and debugging. [4]

## 1.4 Reference Comments

In Chapter 2, 5, 6, most of the DMP related information is cited from *The DMP System and Physical Architecture* [1] and *A protocol stack for futuristic multimedia* [5] by Leif Arne Rønningen, because they are the only materials we can find describing DMP systems. The analytic models are mainly based on the book *Introduction to Queuing Theory (third edition)* by Robert B. Cooper [19], and the paper *Analytical solution of finite capacity M/D/1 queues* by Olivier Brun and Jean-Marie [18].

## 1.5 Related works

Professor Leif Arne Rønningen proposed the concept of Distributed Multimedia Play and has done a lot of work on the development of DMP. The first paper on traffic shaping by Prof Rønningen was published in 1983 [2], which is now the basis of quality shaping for handling DMP networking traffics. The technical report *The DMP System and Physical Architecture* [1] and *A protocol stack for futuristic multimedia* [5] systematically addressed the DMP architecture, the AppTraNet protocol and the related technologies. This Master's thesis is mainly based on the adaptive network scheme presented by those reports.

The Hems Lab is a realization of the DMP architecture, and is part of the ongoing research at the Caruso Lab, Dept of Telematics of NTNU. The test facilities of the

Hems Lab is expected to be a 5-surface stereoscopic collaboration space for audio and video, optimizing the use of video and audio systems when applying DMP in virtual music and musical theatre education, design, rehearsal and performing arts. [1]

Dr Arne Lie contributed a lot on the research of the traffic control scheme of DMP. His researches mainly focus on the performance of traffic control scheme of DMP and its QoS guarantees, by Mathematics analysis and simulations. His work guides me on developing the analytic models and the simulators in this thesis. [6] [7] [8]

## **1.6 Report Outline**

The report is divided into eight chapters and an additional appendix. Technological background for understanding of Distributed Multimedia Play and its special dropping scheme is given in Chapter 2. The two simulators are described in Chapter 3. In Chapter 4, three analytic models are proposed to calculate the delay and loss rate of the packets in one DMP network node. Chapter 5 describes the set up of a DMP collaboration, analyze the delay and loss rate of the packet passing through several network nodes of a DMP collaboration, and present several simulation experiments. Chapter 6 proposes an updated version of dropping scheme to guarantee the delay and reliable delivery of some kind of special packets. Chapter 7 gives some suggestions to improve the performance of DMP systems. Finally, a conclusion is given in Chapter 8, together with some proposals for future work.

## Chapter 2

# Technical Background

This chapter presents the general background for the Distributed Multimedia Play system, its architecture and quality requirements. The AppTraNet protocol is also presented with the focus on the special packet dropping scheme.

It is a brief overview, explaining only the terms that are used in this thesis. This chapter provides background theory for the discussions in the following Chapters. The level of detail is sufficient to understand the further discussion without the need of pre-studies.

### 2.1 Distributed Multimedia Plays

All of the discussions in this Master Thesis are applied to Distributed Multimedia Plays (DMP) introduced by Prof Rønningen.

The concept of Distributed Multimedia Plays was introduced as a proposal for an extension to the coming digital TV system, Multimedia Home Platform, MHP, in a Telenor project in 1996-1999. The technical specification of MHP is available in its website [9]. Now, the intention of the DMP architecture is to present a system architecture that can handle Multimedia Home Space (MHS) distributed services and other services that may be introduced in a fifteen years' time. [1]

MHS can be any room in a single family house or an apartment. It can be a living room or kitchen, or a specialized room built for near-natural virtual collaboration. From those spaces the users can participate in any networked or local collaborations, private or public, with other people, or with servers.

The three-layer DMP systems architecture provides near-natural virtual networked stereoscopic multi-view video and multi-channel sound collaboration between long distance players. Figure 2-1 shows a near natural DMP collaboration that make the two women feel like they are in one room face to face, although they are actually in

different cities.



Figure 2-1: A work lunch as DMP near natural collaboration [1]

### 2.1.1 Quality Requirement of DMP

Three main quality goals of DMP are “near-natural virtual collaboration”, “simple-to-use”, and privacy. [1]

By definition, the near-natural virtual scene has a quality that approaches the natural scene, that is, users should not perceive any difference when experiencing a real scene and the corresponding virtual scene. To obtain the natural level of human perception, the data rates of content may increased to 1000-10000 higher than today's videoconferencing systems and the end-to-end time delay should be restricted in 10-20 milliseconds. [10]

To guarantee the minimum delays, the quality of audio-visual (AV) content is allowed to vary with the traffic in the network. When the traffic overloads the network or some system components fail, the degradation of quality should be graceful.

### 2.1.2 Quality Shaping

The concept of Quality Shaping in DMP was introduced to give graceful degradation of quality when traffic overloads the network or system components fail. The concept builds on controlled dropping of sub-objects (selected packets), and scaling of scene resolution, composition and coding parameters. The scheme guarantees a maximum user-to-user delay without any reservation of resources. [1]

The division of scenes into sub-scenes, objects and finally sub-objects, is of fundamental importance for DMP. This is the basis for making multimedia content

packets independent, so that dropping a content packet will not influence the receiver displaying the multimedia content of other content packets.

The main multi-media content of DMP is scenes. Scenes are composed of real and virtual sub-scenes. The smallest real entities are called objects. Examples are human being, a football, or a background. After being shot by a camera array, objects can be further subdivided into sub-objects. The number of sub-objects per object could be 4, 9, 25, or other. Figure 2-2 shows an example where object 1 is divided into 4 sub-objects, and moves from one position in space to another with a certain speed. The sub-objects can be coded as independent streams and transferred in independent packets. This scheme supports graceful degradation of quality when traffic overloads the network, or when network components fall.

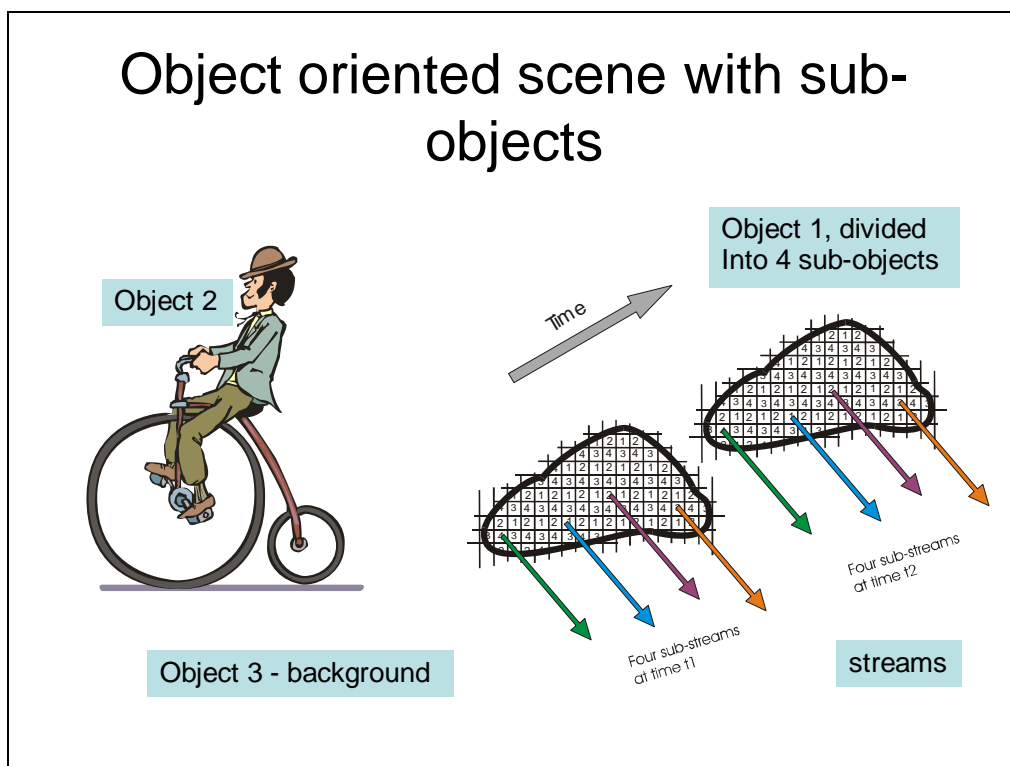


Figure 2-2: object oriented scenes with sub-objects. [1]

### 2.1.3 SceneProfiles and the NOC Scheme

SceneProfiles describe the characteristics and limits of the shooting and presentation space, scenes, sub-scenes, objects and sub-objects, and the movement of objects, and characteristics of each sub-object. During a collaboration the scene may vary fast, objects pop up and disappear, and the importance of objects may also change. The dynamics of the scene shall be within the limits described by the SceneProfile of the scene; otherwise events will not be shot and presented. These limits are determined by physical constraints as well as the technical specifications of the object shooting



equipments and display equipment.

Sub-objects are sorted out according to the SceneProfile associated with an actual service. Important packets have to be specially protected or prioritized. A flat coding structure is expected to give a simpler and smoother degradation of quality when packets are dropped or lost in the network.

Sub-objects are compressed independently using shape adaptation, either with a new scheme called NOC (Near-natural Coding), a slightly modified JPEG2000 [11], or other approaches based on Discrete Wavelet Transforms with Zero Tree Coding [12]. NOC supports the extreme quality requirements of DMP, the 'near-natural feel', and is inspired by the PNG compression standard [13] and the VASARI (the visual arts system for archiving and retrieval of images) project [14]. The NOC scheme has a flat priority structure.

The terms, definitions and concepts used for NOC are related to the PNG standard. Figure 2-3 shows some examples of sub-object division, where the distance between pixels of each sub-object is constant over the area. An object can have any shape. When a fast moving object is shot, the definition of the object can also include the background covered by the object in the previous picture. This is important when the background is updated at smaller rate than the object.

1	2	3	1	2	3
4	5	6	4	5	6
7	8	9	7	8	9
1	2	3	1	2	3
4	5	6	4	5	6
7	8	9	7	8	9

Figure 2-3: Sub-object allocation, 9 sub-objects [1]

In the Master Thesis, we assume that the Sub-Objects are compressed by NOC coding scheme, so the Multimedia Content packets in the DMP system are also called the NOC packets.

## 2.2 AppTraNet protocol

The DMP networks are hierarchical, built with combination of star and mesh topologies. It use fixed routes, and may have alternate routes which give the same number of hops so that it can guarantee minimum delays. [1]

Three layers are defined in DMP architecture, the Linksical layer, the AppTraNet layer and the Application layer. The AppTraNet layer is a combined layer reusing IPV6 functionality and introducing new functionality for DMP. The Protocol has one



combined header, the AppTraNet protocol header, and is the only protocol above the Linksical layer. The AppTraNet Protocol runs on top of the Linksical layer, a combined link and physical layer. This layer assumes optical fibers between network nodes, while wireless mobile connections are alternatives. The protocol data frame includes the AppTraNet Packet as payload, and a preamble for recognizing frame start and bit clock synchronization. Conversion between optical signals and electrical signals are made at each end of Linksical links. Figure 2-4 shows the three-layer architecture.

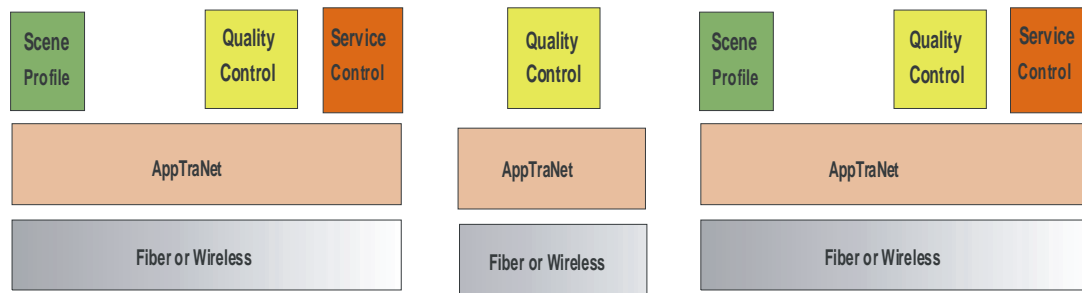


Figure 2-4: Three-layer architecture, Source Access Node, Network Node, and Destination Access Node [1]

### 2.2.1 The AppTraNet packet header

The AppTraNet layer is a combined application, transport and network protocol layer. The DMP architecture uses the IPv6 protocol and IPsec security, and defines a new part supporting specific DMP requirements. To enable efficient design of high-performance hardware, there is only one packet header called the AppTraNet header, where parameters can be processed in parallel by physically parallel hardware. And the packet length of an AppTraNet Protocol is fixed 1.5k Bytes. [1]

In addition to regular parameters defined by IPv6 [16] and IPsec [17] protocol, several parameters are introduced particularly for AppTraNet header:

- **PT**, Payload type, 8 bits
- **PacketRate**, 16 bits, the current packet rate from the sub-object, indicated by the user
- **Sequence number**, 8 bits, used for controlled dropping of progressive JPEG2000 compression layers, and for controlled dropping of sub-objects compressed by the NOC scheme.
- **Timestamp**, 32 bits, used for delay measurements from a network node to an access node, and from users to AccessNodes
- **ServiceID** 48 bits, used to identify on-going services
- **PixelAdrB**, 16 bits, addresses the start pixel of the sub-object represented in this packet (rectangle)

- **PixelAdrE**, 16 bits, addresses the end pixel of the sub-object represented in this packet (rectangle)
- **SPQSP**, 32 bits, reference to SceneProfile and QualityShapingProfile, used for collaborations
- **Reserve count**, 8 bits, counts the number of Reserve bytes
- **Reserve bytes**, 0-64 bytes, for future use [1]

### 2.2.2 Packet types, priority and the dropping scheme

The PT Payload type parameter reserves values for DMP use, defining the payload. Two main groups are defined, the Multimedia Content packets and the Control packets.

Some Control packets are used in typical request-response sequences. It is up to the application how to handle the situation if packets that need acknowledgement are not acknowledged. Since such requests normally do not have real-time requirements, and shall have an extremely low probability for being lost, the output link queuing system shown in Figure 2-5 is introduced as part of the AppTraNet protocol in all nodes after switching (routing). Control packets enter queue Q1, which holds packets on a very large store. If selected by *Sel*, with probability  $(1-p1)$ , the Control packet enters Q0 for link output. The module *H* is an abstraction of the output Linksical layer, sending the packets out at the maximum packet rate given by the link capacity. The transport delays of the Control packet are highly variable, depending on traffic patterns. The maximum length of Q1 should be so large that reaching the maximum should have an extremely low probability, before the admission control decreases the input traffic to the network.

The Multimedia Content packets, with a maximum guaranteed end-to-end delay, but that can be dropped selectively, are sent to buffer Q2 when they arrive. The selector *Sel* fetches packets from Q2 with probability  $p1$ , and forwards to Q0. Q0 has a limited length and determines maximum jitter of a transfer through the node. Q2 is a very short buffer used for dropping according to the sequence number of the packet. If Q0 is full, packets start queuing in Q2.

Assuming the paload of the Multimedia Content packet is coded by NOC scheme and every NOC packet has a sequence number 1, 2, 3, 4, 5, 6, 7, 8, or 9 to distinguish the type of its sub-object, Q2 drops arriving packets as follows:

If Q2.length 0-12 then join Q2 else  
If Q2.length 13-18, drop from sub-object 8 else  
If Q2.length 19-24, drop from sub-object 8 and 7  
If Q2.length 24-29, drop from sub-object 8, 7 and 6  
and so on (sub-object packets 9 are never dropped)

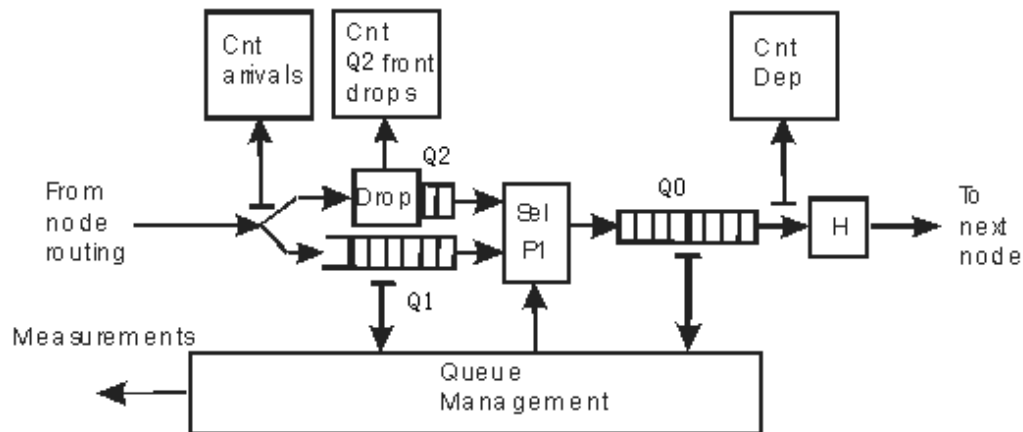


Figure 2-5: Dropping and prioritizing of packets in network nodes, AppTraNet layer. Hardware architecture [1]

The Sequence number is directly given by the sub-object number for NOC coding. This dropping scheme confines the maximum number of packets in Q2, thus restrict the delay of the NOC packets when the traffic load is high. When the traffic load is low, all of the packets will be forward to Q0 immediately when they arrive, so no NOC packets will be dropped when the traffic is low. Optimizing the maximum length of Q2 and Q0, the drop decision lengths of Q2, and the value of p1, is part of the Quality Shaping scheme design.

The DMP nodes with this special dropping scheme are simulated and analyzed in this Master's Thesis.



# Chapter 3

## The Simulators

In this chapter we describe and develop two simulators, *NodeSim* and *RouteSim*. We will first look at the simulators' system requirements. Then we will go through the simulation models and the components of each simulator.

The simulator names, *NodeSim* and *RouteSim* are abbreviations of “Node Simulator” and “Route Simulator”. *NodeSim* simulates an environment that a number of DMP packets passing through a DMP network node in which the special dropping scheme described in Section 2.2.2 is implemented. *RouteSim* simulates an environment that the DMP packets passing through a crowded multi-node DMP collaboration. We can get the delay of each packet, the average waiting time and the loss rate from the output of the two simulators. Several experiments will be carried out on the two simulators in the following chapters.

### 3.1 System requirements

#### 3.1.1 Hardware requirements

The simulators work on the following hardware and operating systems:

The two simulators can be compiled to run in any environment supported by the CIM compiler. During the thesis it has been run successfully in Windows XP and Ubuntu Linux environments. When simulating busy environments (1,000,000 packets per second) over a long time (100 seconds), the simulations are long lasting and memory hungry. Sometimes it will last for several hours. Shorter runs on the other hand take a couple of seconds to complete on a Laptop computer.

The used simulator environments:

1. Laptop Computer 1.83GHz Genuine Intel with 3 Gb memory, 80Gb ATA, Win XP SP2
2. HP server 2x Dual Core 3.4 GHz Xeon 4 GB memory 3x150 GB SCSI RAID 5

Ubuntu Breezy Linux

### 3.1.2 Software preparation

*NodeSim* and *RouteSim* have in both environments been precompiled with CIM. [15] In order to use a maximum of 100Mb of memory, we use the `-m100` parameter of `cim`.

*NodeSim* and *RouteSim* require some additional files to run. These are: *Input\_NodeSim.txt* and *Input\_RouteSim.txt*, containing the traffic rate of each kind of packets, the size of queue, the simulation runtime, and various other settings (See APPENDIX A: Using the simulators). In addition, the two simulators only require a platform which the `cim` compiler is compatible with. It can be: Windows, Linux and Mac OS, UNIX or GNU.

## 3.2 Model Explanation

### 3.2.1 ENTITY and WAITQ

ENTITY is introduced by DEMOS to mirror major dynamic model components in the simulator. In our simulators, the components of the queuing system and the packets are all modeled by ENTITY.

The simulation models are mainly based on the idea of waiting queue. There are three waiting queues in one output link of a network node, which are Q2, Q1, and Q0, see Figure 2-5. Every packet will queue in two waiting queues in one node. The NOC packets queue in Q2 and Q1, while the Content packets queue in Q2 and Q1. Those queues are dedicated objects of type "WAITQ" in Simula with DEMOS.

When an ENTITY arrives in a "WAITQ", it will wait at the end of the queue. When an ENTITY in the head of the queue leaves the "WAITQ", every ENTITY in the queue will move ahead to the next position. If the ENTITY reaches the head of the queue, it can be fetched and leaves the queue. See Figure 3-1:

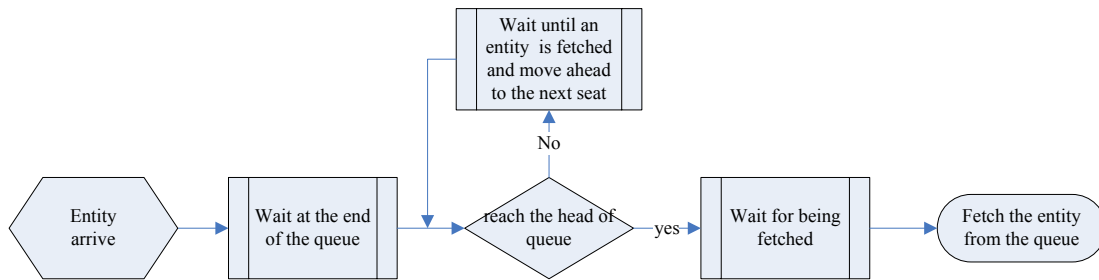


Figure 3-1: the idea of waiting in queue

In our simulators, a dropping scheme is inserted before the ENTITY entering the queue. The dropping scheme drops ENTITIES selectively, thus confines the maximum number of ENTITIES in the waiting queue. When an ENTITY arrives, and if the state of queue combined with some properties of the ENTITY satisfies the dropping condition, the ENTITY will be dropped instead of being put into the queue. See Figure 3-2:

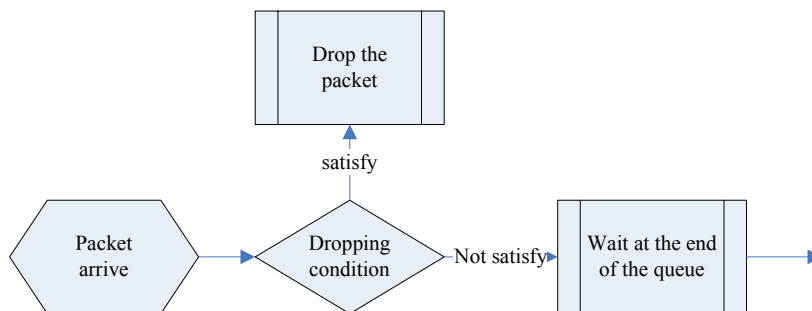


Figure 3-2: The Dropping Scheme of WAITQ

The dropping condition can be defined according to our demands. For example, in our simulator *NodeSim*, Q2 is a waiting queue and Q2.length is the number of packets which have already existed in Q2, then Q2 drops some incoming packets as follows:

If Q2.length 0-12 then join Q2 else  
 If Q2.length 13-18, drop from sub-object 8 else  
 If Q2.length 19-24, drop from sub-object 8 and 7  
 If Q2.length 24-29, drop from sub-object 8, 7 and 6  
 And so on.

### 3.2.2 The model of simulator *NodeSim*

The simulator *NodeSim* simulates the network traffic passing through a DMP network node. The parameters of this simulator are given by *Input\_NodeSim.txt*. These parameters are: the link capacity of the output link, the value of p1, the size of Q0, the length of a simulation period, the interarrival distributions of the NOC packets, the interarrival distributions of the Control packets and the parameters of those

distributions. We can get the delay of each NOC and Control packet, the mean delay and the loss rate of the NOC packets from the output files.

There are three WAITQs and six ENTITIES involved in this simulation program. Q0, Q1 and Q2 are WAITQs. ENTITIES *packetNOC* and *packetCTRL* are the packets passing through the node, while ENTITIES *NOC Generator*, *Control Generator*, *Sel Box*, and *H* are four components of the system.

The component *NOC Generator* generates *packetNOC* which joins Q2 one by one, while *Control Generator* generates *packetCTRL* which joins Q1. *packetNOC* might be lost according to its sequence number and the state of Q2. *Sel Box* fetches packets from Q2 and Q1 and forwards them to Q0, and *H* fetches packets from Q0 and sends them out of the node one by one. Figure 3-3:

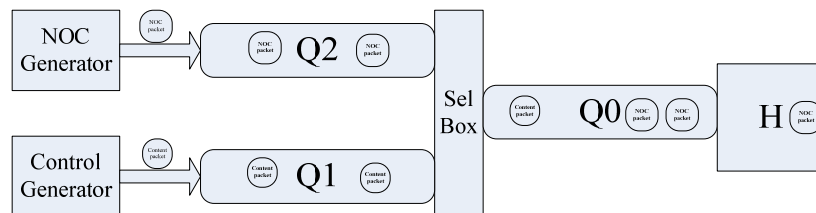


Figure 3-3: the components of *NodeSim* simulator

### 3.2.2.1 Packets Generators

*NOC Generator* and *Control Generator* generate NOC or Control packets periodically. The distributions of the interarrival time are read from *Input\_NodeSim.txt*. Normally, the packets are generated from Poisson Process, while we will also do some experiment to try other distributions. Figure 3-4 shows the working process of either *NOC Generator* or *Control Generator*.

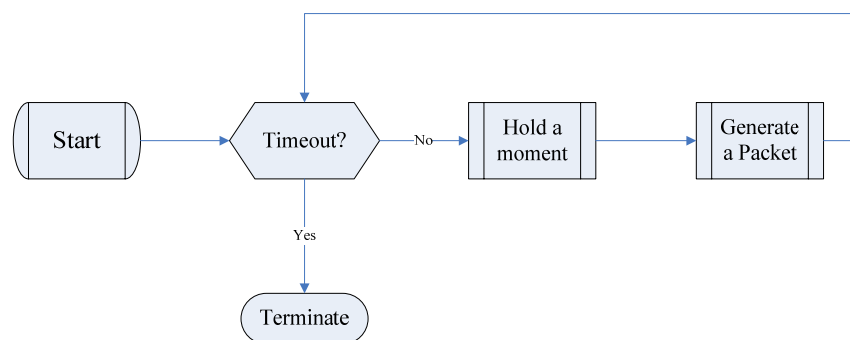


Figure 3-4 Packets Generator generates packets periodically

### 3.2.2.2 Sel Box

*Sel Box* fetches packets from Q2 and Q1 and forwards them to Q0. *Sel Box* generates a random number  $p$  which follows a standard uniform distribution every time before it fetches a packet. If  $p$  is larger than  $p_1$ , *Sel Box* will fetch the packet from Q1; while if



$p$  is smaller than  $p_1$ , it will fetch the packet from Q2. See Figure 3-5:

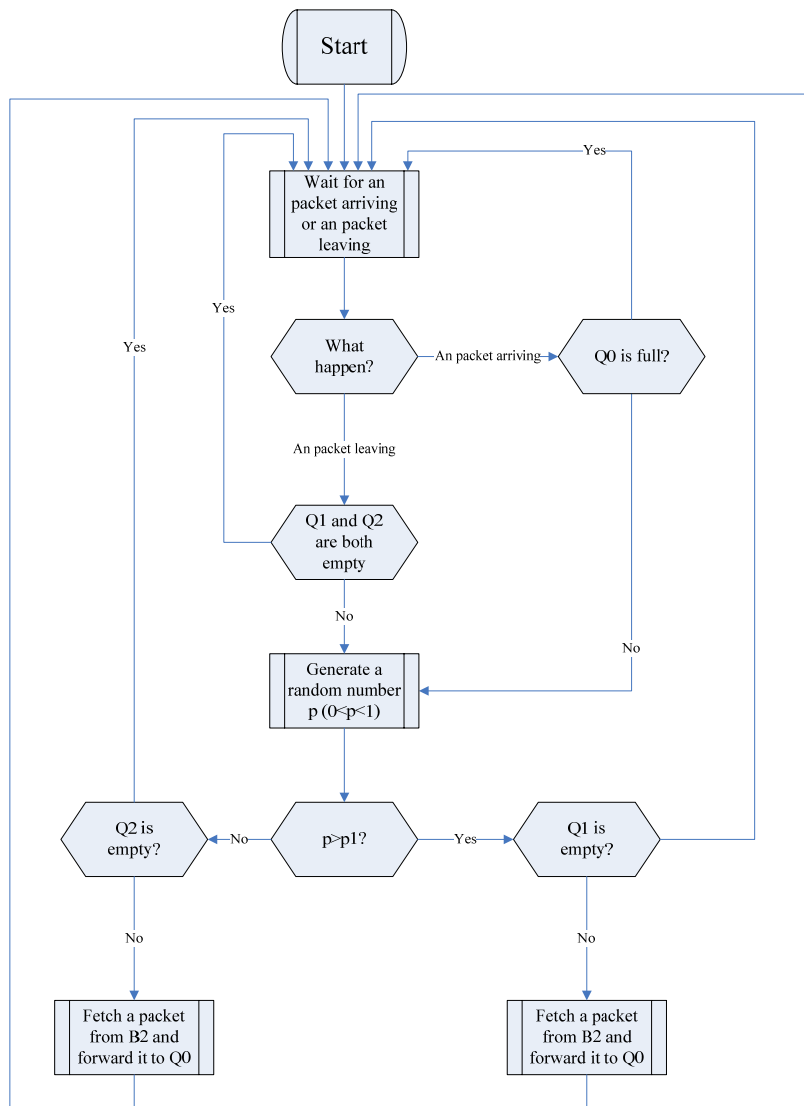


Figure 3-5: *Sel Box* fetches packets from Q2 and Q1 and forwards them to Q0.

### 3.2.2.3 Output component H

*H* fetches packets from Q0 periodically. Each packet, either a Control packet or a NOC packet, stays in *H* for a fixed time, then *H* will send it out of the node and fetch the next packet from Q0. See Figure 3-6.

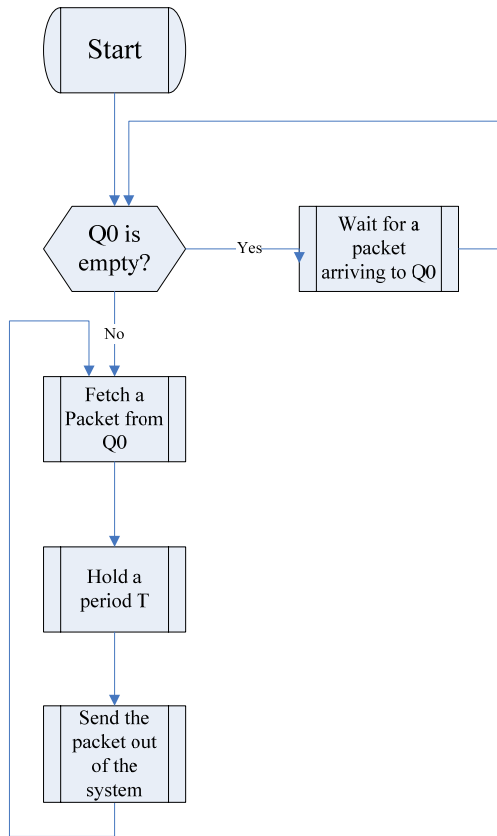


Figure 3-6: *H* fetches packets from *Q0* and send them out periodically.

### 3.2.2.4 *packetCTRL*

The ENTITY *packetCTRL* represents the Control packets, which is generated by *Control Generator*. It will wait in *Q1* at first, until it is fetched by *Sel Box* and wait in *Q0*. Then it will be fetched by *H* and finally be sent out of the node. Figure 3-7 shows the life cycle of a *packetCTRL*.

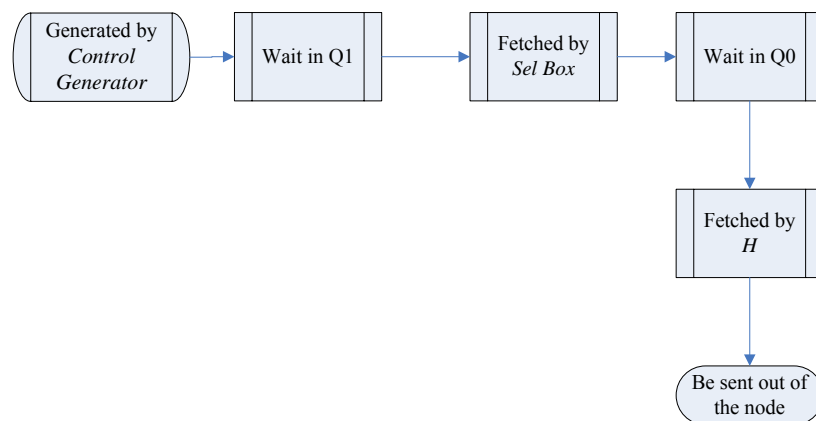


Figure 3-7: The lifecycle of a Control packet in *NodeSim*

### 3.2.2.5 *packetNOC*

The ENTITY *packetNOC* represents Multimedia Content packets, which is also called

NOC packets. *packetNOC* is generated by *NOC Generator*. The dropping scheme of queue Q2 is written in the code of *packetNOC*. After a *packetNOC* is generated, the sequence number of the packet and the number of packets in Q2 will be checked. If the dropping condition is satisfied, for example, the sequence number of the packet is 8 while 15 packets staying in Q2 at that time, this packet will be dropped. Otherwise, it will enter Q2 and wait, until it is fetched by *Sel Box* and forward to Q0. It will be fetched by *H* and finally be sent out of the node. Figure 3-8 shows the life cycle of a *packetNOC*.

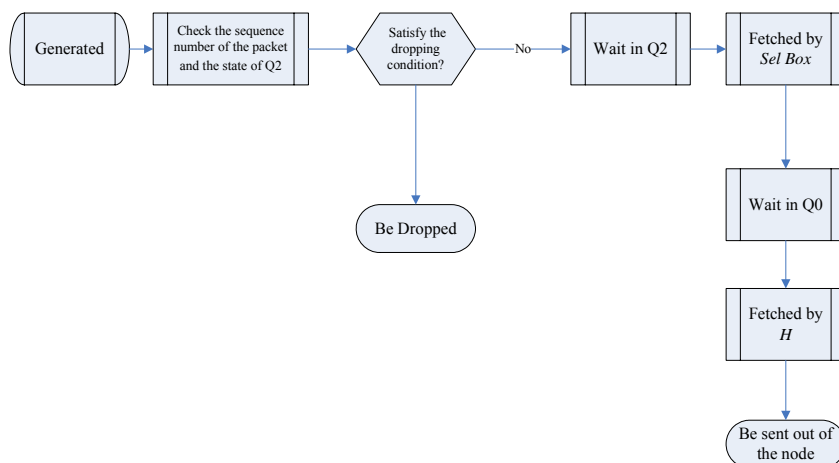


Figure 3-8: the lifecycle of a NOC packet in *NodeSim*

### 3.2.3 The Model of Simulator *RouteSim*

The simulator *RouteSim* is designed to simulate the network traffics of a specific route of DMP collaboration.

In *RouteSim*, a DMP user (The Source Host) sends his video streams to another DMP user (Destination Host). The video streams along with some Control packets are delivered through five network nodes in the path towards the destination. The Special dropping scheme is placed in each network node. There are other traffics compete for the buffer of the output port in each network node with the packets from The Source Host.

In Figure 3-9, we can see that there are a Source Host and a Destination Host representing the two users who are communicating to each other. Network Node 1, 2, 3, 4, 5 are the five network nodes in the path between The Source Host and The Destination Host. The black straight arrows from The Source Host to The Destination Host through five network nodes represent the video stream (the NOC packets) and the Control packets which are transmitted from the Source Host to The Destination Host. Five Packet Generators generates packets to each network node to compete for the buffer resource of output port of the network node with the packets from the Source Host. Those traffics also contain two kinds of packets, the NOC packets and the Control packets. The packets from Router Generator 1 are switched to the output

port connected with Network Node 2, and then they are switched to Router Dest 1 through other output port. The packets from Router Generator 2, 3, 4 are routed to Router Dest 2, 3, 4. The packets from Router Generator 5 are switched to the access node of the Destination Host, but they are accepted by other users except the Destination Host.

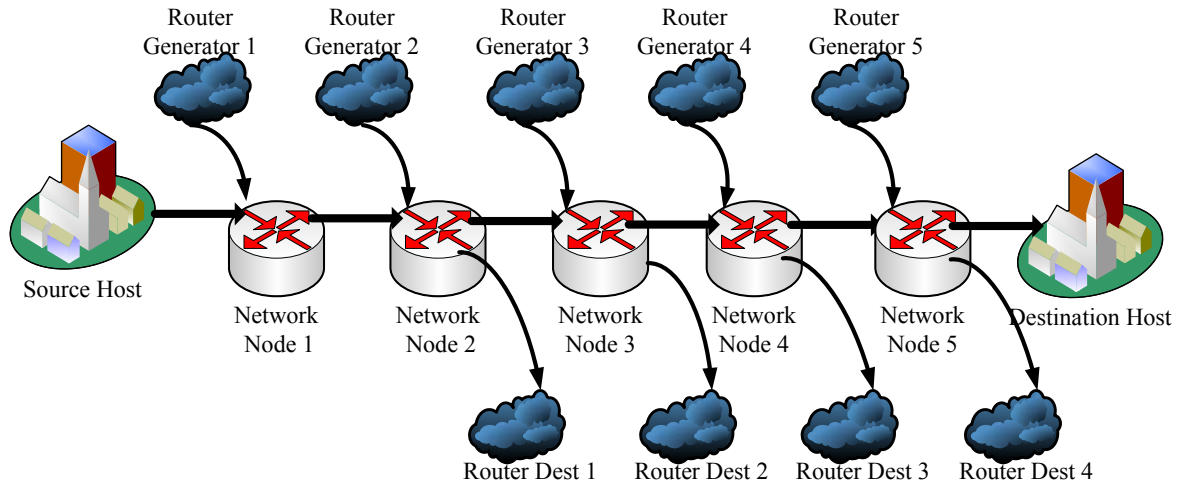


Figure 3-9: Simulation model of *RouteSim*

The parameters of *RouteSim* are given by *Input\_RouteSim.txt*. These parameters are: the link capacity of the output link of each network node, the value of  $p_1$  of each network node, the size of  $Q_0$  of each network node, the length of a simulation period, the interarrival distributions of the NOC packets from The Source Host, the interarrival distributions of the Control packets from the Source Host, the interarrival distributions of the NOC packets from each Router Generator, interarrival distributions of the Control packets from each Router Generator and the parameters of those distributions. We can get from the output files the delay of each NOC and Control packet from the Source Host to the Destination Host, the mean delay and the loss rate of the NOC packets from The Source Host to the Destination Host, and the mean delay and the loss rate of the NOC packets in different network nodes.

Figure 3-10 shows the components in *RouteSim*. There are four kinds of packet generators in *RouteSim*. *Source NOC Generator* and *Source Control Generator* generate the NOC and the Control packets from the Source Host, while *Router NOC Generator* and *Router Control Generator* generate the NOC and the Control packets to compete with the packets from The Source Host for the buffer resource in each network node. *DMP node* is a group of ENTITIES which constitute a network node in the path between The Source Host and The Destination Host. There are five *DMP nodes* in *RouteSim* which represent five network nodes in the path. ENTITIES *Source packetNOC* and *Source packetCTRL* are the packets transmitted through the whole route which is generated by *Source NOC Generator* and *Source Control Generator*,

while *Router packetNOC* and *Router packetCTRL* are the packets who only pass through one network node in *RouteSim* and are switched to other network nodes beyond *RouteSim*.

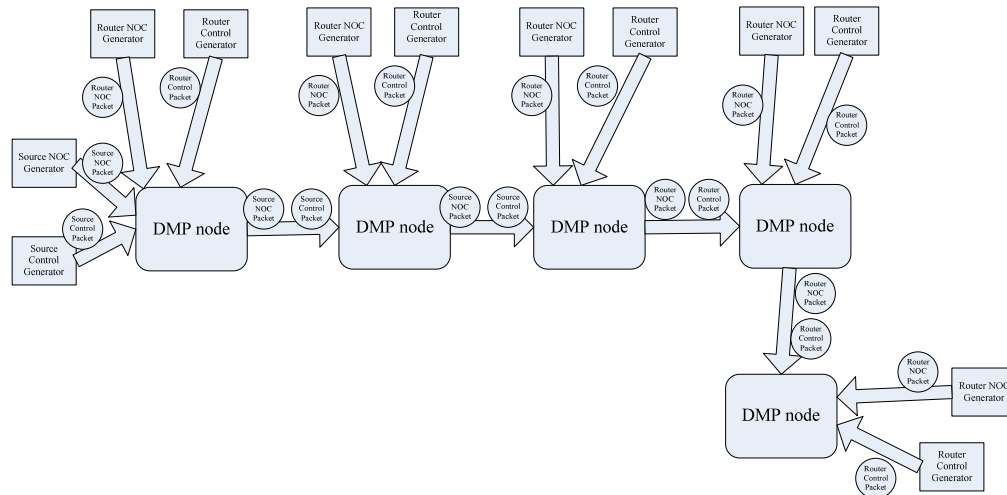


Figure 3-10: the components of *RouteSim*

### 3.2.3.1 DMP node

A *DMP node* is a group of components. Those components constitute a simplified *NodeSim* simulator which has no packet generators and packets. See Figure 3-11:

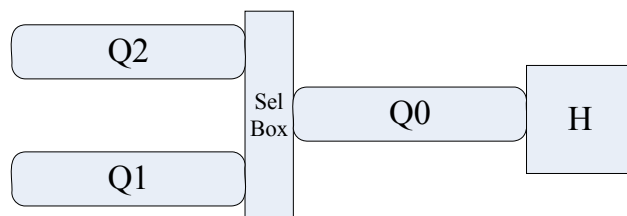


Figure 3-11: the components in *DMP node*

The three waiting queues Q2, Q0 and Q1 are introduced in Section 3.2.1, while the components *Sel Box* and *H* are introduced in Sections 3.2.2.2 and 3.2.2.3.

### 3.2.3.2 Packets Generators

The four Packets Generators *Source NOC Generator*, *Source Control Generator*, *Router NOC Generator* and *Router Control Generator* generate the NOC or Control packets periodically. They work in the same way with the packet Generators in Simulator *NodeSim* which are introduced in 3.2.2.1. The distribution of interarrival time is read from *Input\_RouteSim.txt*. Normally, the packets are generated from Poisson Process, while we can also do some experiment trying other distributions.

### 3.2.3.3 Router packetCTRL

The ENTITY *Router packetCTRL* represents the Control packets which are generated by *Router Control Generator*. *Router packetCTRL* passes through only one network node of *RouteSim* and then is switched to other network nodes beyond *RouteSim*. It will be put in Q1 of one network node at first, until it is fetched by *Sel Box* and wait in Q0 of that node. Then it will be fetched by *H* and finally be sent out of the node to other nodes beyond the simulator. Figure 3-12 shows the life cycle of a *Router packetCTRL*.

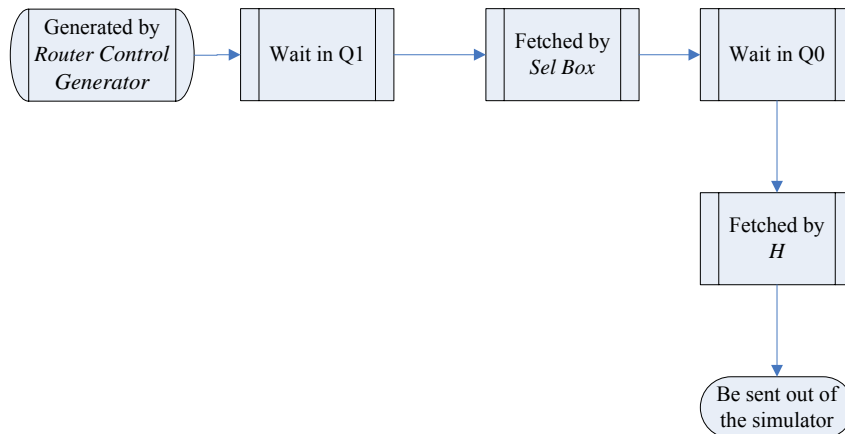


Figure 3-12: the lifecycle of a *Router packetCTRL*

### 3.2.3.4 Router packetNOC

The ENTITY *Router packetNOC* represents Multimedia Content packets, which is also called NOC packets. *Router packetNOC* is generated by *Router NOC Generator*, pass through only one network node of Simulator *RouteSim* and then is switched to other network nodes beyond *RouteSim*. The dropping scheme of queue Q2 is written in the code of *Router packetNOC*. After a *Router packetNOC* is generated, the sequence number of the packet and the number of packets in Q2 will be checked. If the dropping condition is satisfied, for example, the sequence number of the packet is 8 while 15 packets staying in Q2 at that time, this packet will be dropped. Otherwise, it will be put in Q2 of that network node, until it is fetched by *Sel Box* and forward to Q0. It will be fetched by *H* and finally be sent out of the node. Figure 3-13 shows the life cycle of a *Router packetNOC*.

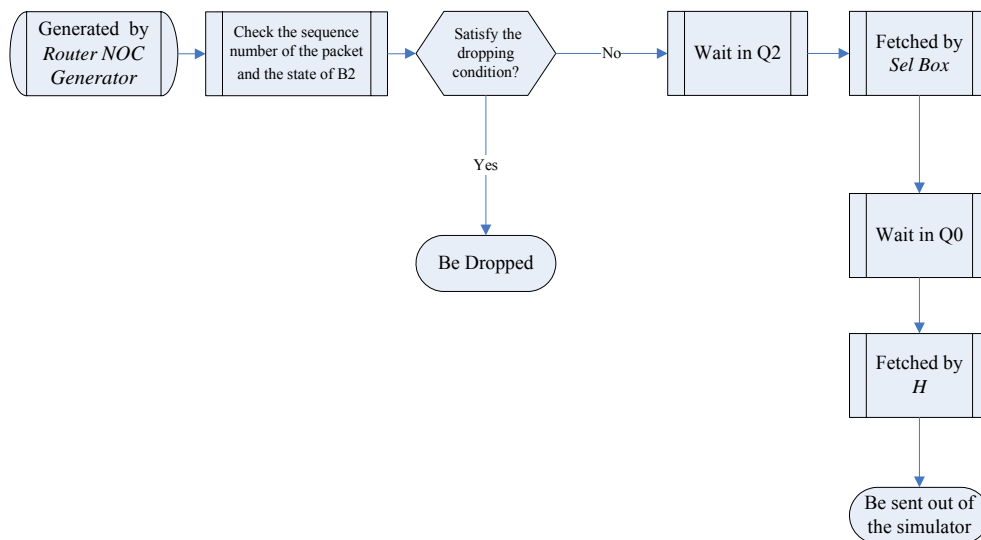


Figure 3-13: the lifecycle of a *Router packetNOC*

### 3.2.3.5 Source packetCTRL

The ENTITY *Source packetCTRL* represents the Control packets which are generated by *Source Control Generator*. *Source packetCTRL* passes through all the five *DMP nodes* in *RouteSim* one by one. In each *DMP node*, it will queue in Q1 of this node at first, until it is fetched by *Sel Box* and wait in Q0 of the node. Then it will be fetched by *H* and be sent to the next *DMP node*, waiting in Q0 of that node. When it comes to the last *DMP node* and fetched by *H* of the last *DMP node*, it will be sent out of the simulator which means it has arrived at The Destination Host and the end of its lifecycle in *RouteSim*. Figure 3-14 shows the life cycle of a *Source packetCTRL*.

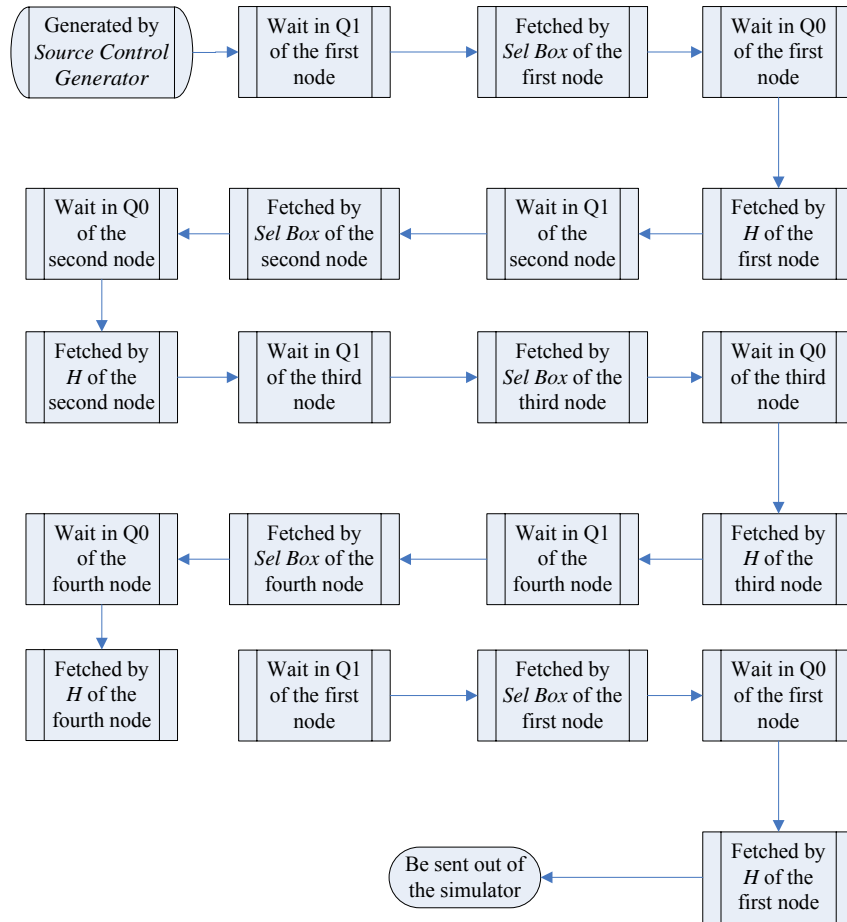


Figure 3-14: lifecycle of a *Source packetCTRL*

### 3.2.3.6 *Source packetNOC*

The ENTITY *Source packetNOC* represents the NOC packets which are generated by *Source NOC Generator* and pass through all the five network nodes in the route of Simulator *RouteSim*. The dropping scheme of Q2 is written in the code of *Source packetNOC*. Every time when a *Source packetNOC* comes into a *DMP node*, the sequence number of the packet and the number of packets in Q2 will be checked. If the dropping condition is satisfied, for example, the sequence number of the packet is 8 while 15 packets are staying in Q2, this packet will be dropped and its lifecycle is ended. Otherwise, it will enter and wait in Q2 of that *DMP node*, until it is fetched by *Sel Box* and forward to Q0 of that node. Then it will be fetched by *H* and be sent to the next *DMP node* in the path. When it comes to the last *DMP node* and fetched by *H* of the last *DMP node*, it will be sent out of the simulator which means it has arrived at The Destination Host and the end of its lifecycle. Figure 3-15 shows the lifecycle of a *Source packetNOC*.



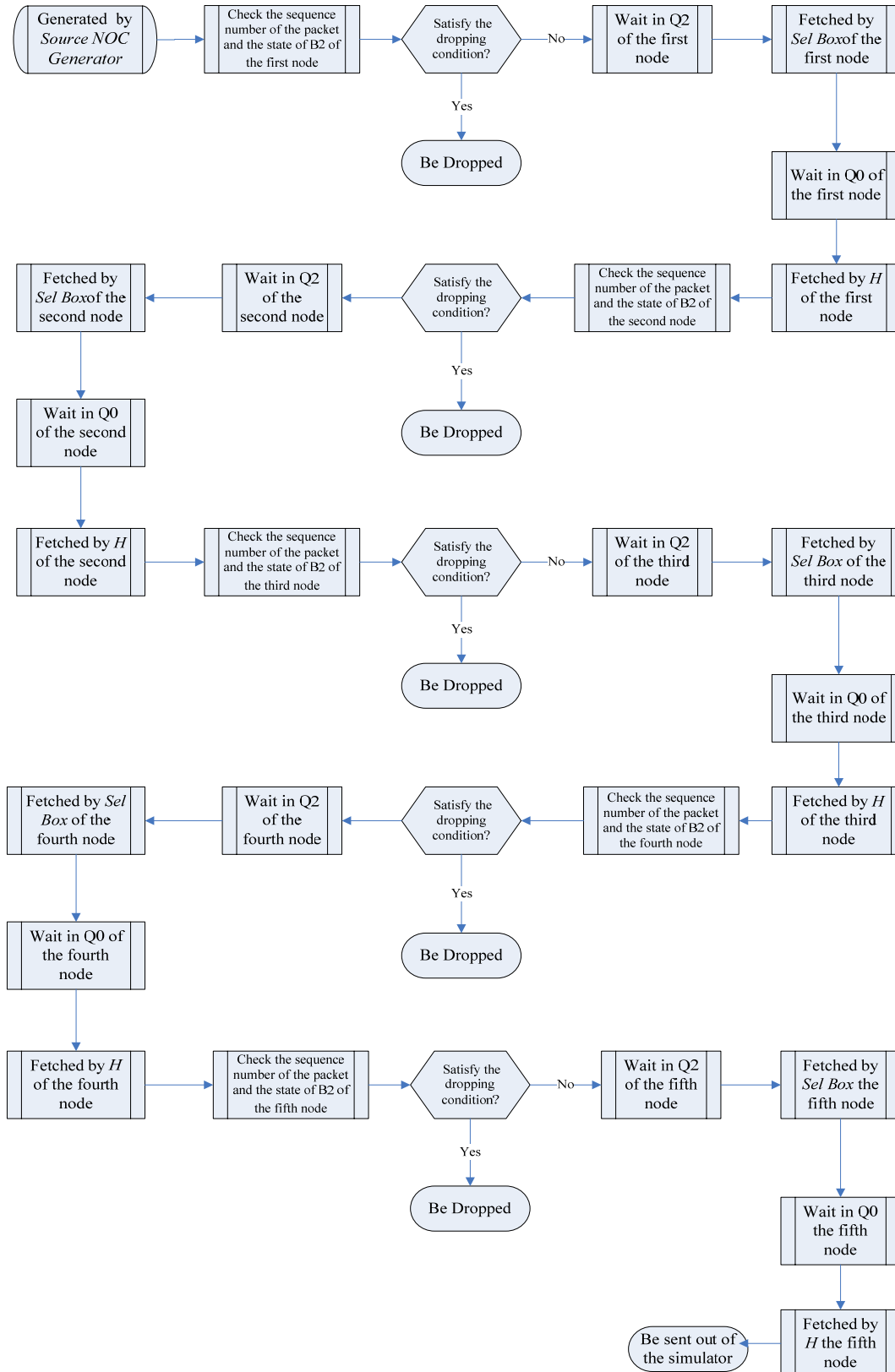


Figure 3-15: the lifecycle of a *Source packet NOC*

## 3.4 Output of the Simulators

The output of *NodeSim* and *RouteSim* will be saved in text files.

### 3.4.1 Output of *NodeSim*

*NodeSim* simulates the traffics of one network node. Given the distribution of interarrival time of the NOC packets, the distribution of interarrival time of Control packets, the output link capacity, the length of Q0, the value of  $p1$ , and the duration of one simulation period by the input file, we can get the delay of each packet, the average waiting time and the loss rate from the output of the simulator.

When we execute *NodeSim.exe*, *NOC Generator* and *Control Generator* in the simulator will start generating packets and the packets will pass through the node one by one. The delay of each packet will be recorded in *NodeSim\_delay\_NOC.txt* and *NodeSim\_delay\_NOC.txt*. The average waiting time and the number of packets in the queues will be measured six times, that is, the total running time is divided into six periods and the registers for calculating average delay and loss rate will be cleared at the beginning of each period. The average delay and loss rate are reported by text files, See APPENDIX A: Using the simulators for the detailed description of those output files.

### 3.4.2 Output of *RouteSim*

*RouteSim* simulates the traffic of a DMP collaboration of five network nodes. Given the distribution of interarrival time of the NOC and Control packets from the Source, the distribution of interarrival time of the NOC and Control packets from other nodes, the output link capacity, the length of Q0 and the value of  $p1$  of each node, and the duration of one simulation period by the input file, we can get the delay of each packet, the average waiting time and the loss rate from the output of the simulator.

When we execute *RouteSim.exe*, the *Source NOC Generator* and *Source Control Generator* will start generating packets from the source and those packets will pass through the five network nodes one after one. Packets from other nodes are also be generated by *Router NOC Generator* and *Router Control Generator*. The delay of each packet from the Source Host to the Destination Host will be recorded in two text files: *RouteSim\_delay\_NOC.txt* and *RouteSim\_delay\_Ctrl.txt*. The average waiting time, the number of packets dropped and the number of packets in the queues will be measured six times, that is, the total running time is divided into six periods and the registers for calculating average delay and loss rate will be cleared at the beginning of each period. The average delay and loss rate are reported by text files, See APPENDIX A: Using the simulators for the detailed description of those output files.

## Chapter 4

# Analytic Models for one DMP network Node

The Distributed Multimedia Play architecture includes in all network nodes a special output link queuing system to support the graceful degradation of quality when traffic overloads the network or system failures happen. In this Chapter, we use three mathematic models to compute the average waiting time and loss rate of the packets passing through a DMP network node.

M/D/1 with infinite queue model is used to compute the average waiting time of Control packets when the total traffic load is low or high, and the average waiting time of the NOC packets when the total traffic load is low; M/D/1 with dynamic queue model and M/B/1 with dynamic queue model are used to compute the average waiting time and loss rate of the NOC packets when the traffic load is high.

### 4.1 Preliminaries

As what we have discussed in Chapter 2, adaptive QoS and coding schemes are technically designed and included in every DMP network node so that the effect from packet drop is minimized. The AppTraNet protocol is designed to encapsulate DMP packets on top of the linksical layer. The AppTraNet protocol is a combination of the application, transport and network layer in one protocol, with one combined header. To support efficient hardware design, and to reduce the complexity of the network, the packet length of AppTraNet protocol is fixed 1.5k bytes.

There are two types of packets in DMP networks, Control packets and Multimedia Content packets. The Control packets normally do not have real-time requirements, and shall have an extremely low probability for being lost. The Multimedia Content

packets, also called NOC packets, shall be guaranteed a maximum end-to-end delay, but can be dropped selectively by identifying the sub-object sequence number which is marked 1, 2... or 9 in AppTraNet header. The control or NOC packets that shall not be dropped and shall have a guaranteed delay are marked sequence number 9.

#### 4.1.1 The Output link queuing system of the AppTraNet protocol

The output link queuing system shown in Figure 4-1 is introduced as part of the AppTraNet protocol in all network nodes. If Q0 is not full, both Control and NOC packets enter Q0 directly. If Q0 is full, the Control packets enter queue Q1, and the NOC packets start queuing in Q2. Q1 holds packets on a very large store, so the delays of the Control packets are highly variable, depending on traffic patterns. The size of Q1 should be so large that reaching the maximum should have an extremely low probability. Q2 is a very short buffer used for storing NOC packets and dropping some packets according to their sequence numbers. The selector *Sel* fetches packets from Q2 with probability  $p_1$ , or fetches packets from Q1 with probability  $(1-p_1)$ , and forwards them to Q0. Q0 has a limited length, but is much longer than Q2. The module *H* is an abstraction of the output Linkical layer, sending the packets out at the maximum packet rate given by the link capacity.

We call Q2 a dynamic queue because it may drop some incoming packets. The Multimedia Content packets are coded by NOC coding scheme and there are nine types of sub-objects, Q2 drops arriving packets as follows:

Q2.length is the number of packets waiting in Q2.  
If Q2.length 0-12 then join Q2 else  
If Q2.length 13-18, drop from sub-object 8 else  
If Q2.length 19-24, drop from sub-object 8 and 7 else  
If Q2.length 25-29, drop from sub-object 8, 7 and 6 else  
If Q2.length 30-34, drop from sub-object 8, 7, 6 and 5 else  
If Q2.length 35-39, drop from sub-object 8, 7, 6, 5 and 4 else  
If Q2.length 40-45, drop from sub-object 8, 7, 6, 5, 4 and 3 else  
If Q2.length 46-49, drop from sub-object 8, 7, 6, 5, 3, and 2 else  
If Q2.length  $\geq 50$ , drop from sub-object 8, 7, 6, 5, 3, 2 and 1;  
(Sub-object packets 9 are never dropped)

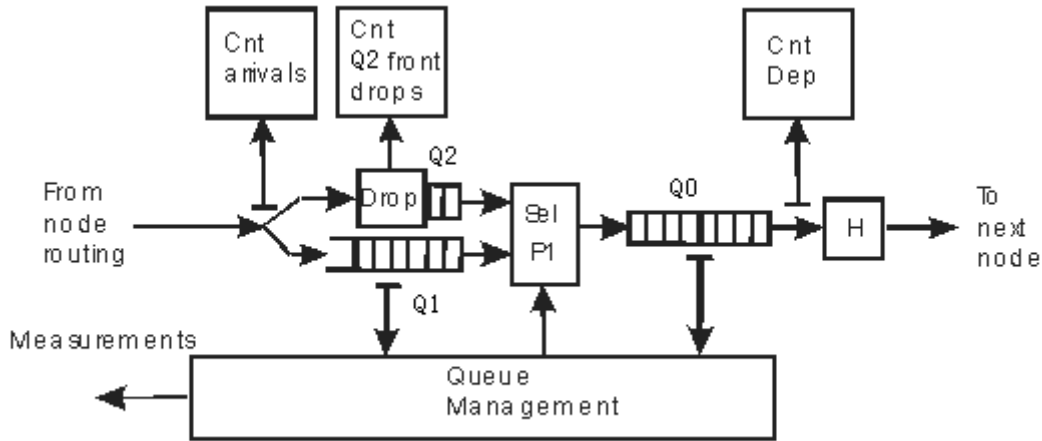


Figure 4-1: the output link queuing system of DMP node [1]

### 4.1.2. Definitions and Assumptions

In this Chapter, we will build mathematical models to compute the average waiting time of the Control or NOC packets and the loss rate of the NOC packets when they pass through a DMP network node. There are some definitions and assumptions before our calculation.

In Figure 4-1, the output component  $H$  is regarded the server in our models. Because the sizes of Control packets and NOC packets are the same and the link capacity of the linksical layer is a constant, so the service time of both types of packets is a constant. We define  $T$  as the service time for one packet.

In our calculation, we assume that the total input streams of the NOC packets and the Control packets both approach Poisson Process, that is, the interarrival time distribution approaches the negative exponential distribution when a large number of small, independent streams merge in one output link. We also assume that the arrival rates of both types of packets are stable. We define  $\lambda_1$  as the arrival rate of the NOC packets, and define  $\lambda_2$  as the arrival rate of the Control packets. Given service time  $T$ ,  $\rho_1$  is defined as the traffic load of the NOC packets and  $\rho_2$  is defined as the traffic load of Control packets, then we have

$$\rho_1 = \lambda_1 \cdot T \quad (4-1)$$

$$\text{And } \rho_2 = \lambda_2 \cdot T. \quad (4-2)$$

Every NOC packet is marked by a sequence number 1, 2, 3, 4, 6, 7, 8, or 9 according to its priority in the dropping scheme. The packets which have sequence number 9 shall never be dropped and have a guaranteed delay, and the number of those packets is very low. To simplify our calculation, we neglect packets with sequence number 9

in the analytic models of this Chapter, (those packets have guaranteed delay will be analyzed in Chapter 6: Guaranteed delay and reliable delivery using sub-obj seq. no.9), so we assume that the packets with sequence number 9 never appear in the system. And we assume that the packets with sequence number 1, 2, 3, 4, 6, 7 or 8 are coming under the same probability. That is, in our calculation, the number of packets with sequence number 1 is one eighth of the total number of packets arriving during a period, which is equal to the number of incoming packets with sequence number 2, 3, 4, 6, 7 or 8 in that period. So dropping packets with sequence number 8 means dropping one eighth of the incoming NOC packets, and dropping packets with sequence number 8 and 7 means dropping one fourth of the incoming NOC packets, and so on.

The buffer Q0 is a finite queue with  $N_{Q0}$  seats, Q1 is an infinite queue, and Q2 is a dynamic queue whose loss rate increases when the number of waiting packets in this queue increases, and if the number of packets reaches the maximum size of Q2, all of the incoming NOC packets will be dropped. Dropping some packets means the decreasing of arrival rate. So if the original arrival rate of the NOC packets is  $\lambda_2$ , because some of them arriving are dropped according to their sequence numbers and the number of packets in Q2, the de facto arrival rates of the NOC packets entering Q2, denoted by  $\lambda_2'$  is as follows:

If Q2.length 0-12 then  $\lambda_2' = \lambda_2$  else  
 If Q2.length 13-18, then  $\lambda_2' = \frac{7}{8} \lambda_2$  else  
 If Q2.length 19-24, then  $\lambda_2' = \frac{6}{8} \lambda_2$  else  
 If Q2.length 25-29, then  $\lambda_2' = \frac{5}{8} \lambda_2$  else  
 If Q2.length 30-34, then  $\lambda_2' = \frac{4}{8} \lambda_2$  else  
 If Q2.length 35-39, then  $\lambda_2' = \frac{3}{8} \lambda_2$  else  
 If Q2.length 40-45, then  $\lambda_2' = \frac{2}{8} \lambda_2$  else  
 If Q2.length 46-49, then  $\lambda_2' = \frac{1}{8} \lambda_2$  else  
 If Q2.length  $\geq 50$ , then  $\lambda_2' = 0$ .

## 4.2 The Average Waiting Time of the Control Packets

The Control Packets are never lost because the size of Q1 is very large and the service time of a packet is a constant, so we use M/D/1 with infinite queue model to calculate

the average waiting time of the Control packets.

### 4.2.1 Low Traffic Rate

When the traffic load is low, the average number of packets in Q0 is very small, so all of the incoming packets will be forward to Q0 immediately when they arrive and no packets will wait in Q2 (NOC packets) and Q1 (Control packets). Therefore, Q0 is considered infinite. Because the service times of the two types of packets are the same, their arrivals are both characterized by Poisson process and they enter the same queue Q0, the average waiting time of both types of packets are the same.

If the arrival rate of the NOC packets is  $\lambda_2$  and the arrival rate of the Control packets is  $\lambda_1$ , then the total arrival rate of both types of packets  $\lambda = \lambda_1 + \lambda_2$ . The service time of both types of packets is a constant T. Then the total traffic load, denoted by  $\rho$ , is

$$\rho = (\lambda_1 + \lambda_2) \cdot T \quad (4-3)$$

From the formulas of M/D/1 with infinite queue model [28], the average number of packets in Q0, denoted by  $N_{Q0}$ , is

$$N_{Q0} = \frac{\rho^2}{2(1-\rho)} \quad (4-4)$$

The average number of packets in H is the offered load, denoted by  $\rho$ , from equation 4-3. So the average waiting time of an arriving packet, denoted by  $T_{W-LowLoad}$ , is

$$T_{W-LowLoad} = (N_{Q0} + \rho) \cdot T \quad (4-5)$$

The equations 4-4 and 4-5 give the average waiting time (delay) of both the NOC and Control packets when the traffic load is low.

### 4.2.2 High Traffic Rate

When the arrival rate of total packets rises, the number of packets in Q0 will reach its limit and the Control packets will start to queue in Q1. The Selection scheme *Sel* goes into effect in this condition to fetch packets from Q1 to Q0.

The arriving of NOC packets and Control packets are characterized by Poisson Processes with different arrival rate, and the arrival rates are stable.

When the total traffic rate of the NOC and Control packets are higher than 100%, Q0 is supposed always full. When a packet departs from Q0, no matter a NOC or Control packet it is, *Sel* will fetch a packet from Q1 or Q2. The probability of *Sel* fetching a NOC packet from Q2 is  $p_1$ , and the probability of *Sel* fetching a Control packet from

Q1 is  $(1-p1)$ .

Because the probability of *Sel* fetching a Control packet from Q1 is  $(1-p1)$ , and *Sel* always waits until there is at least one place available in Q0 so the average service rate of *Sel* for the Control packets is  $(1-p1)/T$ . To ensure the number of packets waiting in Q1 never goes to infinity, the arrival rate  $\lambda1$  should not be larger than the service rate, that is  $\lambda1 \leq (1-p1)/T$ . According to the service rate  $(1-p1)/T$ , the traffic load of *Sel* for Control packets, denoted by  $\rho1'$  is

$$\rho1' = \frac{\lambda1}{T / (1-p1)} = \frac{\rho1}{1-p1} \quad (4-6).$$

So we can get the average number of packets in Q2 from the formula of M/D/1 with infinite queue model:

$$N_{Q2} = \frac{\rho1'^2}{2(1-\rho1')} \quad (4-7)$$

Because Q0 is always full in this condition, the number of packets in Q0, denoted by  $N_{Q0}$  is  $N_{Q0} = N_{Q0}$ , and the number of packets in *H* is 1, the average waiting time for Control packets is:

$$T_{W\_Ctrl\_HighLoad} = (N_{Q2} + N_{Q0} + 1) \cdot T \quad (4-8)$$

The equations 4-6, 4-7 and 4-8 give the average waiting time of the Control packets when the total traffic load  $\rho$  is higher than 100%. However, the traffic load of the Control packets in *Sel*  $\rho1'$  must be lower than 100%, which can be guaranteed by adjusting the parameter  $p1$ , otherwise, the number of packets in Q1 will go to infinity.

### 4.3 The Average Waiting Time and Loss Rate of the NOC Packets

As we have discussed in Section 4.2.1, when the total traffic rate is low, the average waiting time of both types of packets are the same. When the total traffic rate is low, we can also get the average waiting time of NOC packets from the equations 4-4 and 4-5. And in this condition, no packets will be dropped, so the loss rate of the NOC packets is always 0 when the total traffic rate is low.

In this section, we try to calculate the average waiting time and loss rate of the NOC



packets when the traffic load is high. Two mathematical models are proposed to solve the problem.

### 4.3.1 The model M/D/1 with dynamic queue

We build this mathematic model to calculate the delay and loss rate only for the NOC packets. The arriving of the NOC packets and the Control packets are characterized by Poisson Processes with different arrival rate. When the total traffic rate of the both types of packets is higher than 100%, some of the NOC packets will be dropped selectively.

When *Sel* fetches a packet, the probability of *Sel* fetching a NOC packet from Q2 is  $p_1$ , and the probability of *Sel* fetching a Control packet from Q1 is  $(1-p_1)$ . The packets in Q0 are all fetched by *Sel*. When the traffic load is high, Q2 and Q1 are never empty and there are always packets in Q2 and Q1 waiting for being fetched by *Sel*. Since the size of Q0 is relatively large, the fraction of the numbers of two types of packets in Q0 is equal to the fraction of the numbers of two types of packets fetched by *Sel* during a period, which is approximately  $p_1 : (1-p_1)$ .

The size of Q0 is denoted by  $N_{Q0}$ . When Q0 is full, the number of NOC packets in Q0, denoted by  $N_{Q0\_NOC}$  is approximately

$$N_{Q0\_NOC} = N_{Q0} \cdot p_1 \quad (4-9),$$

and the number of the Control packets in Q0, denoted by  $N_{Q0\_Ctrl}$  is approximately

$$N_{Q0\_Ctrl} = N_{Q0} \cdot (1 - p_1) \quad (4-10).$$

The service time of a NOC or Control packet in *H* is a constant  $T$ . When the traffic rate of the total packets is high, *Sel* fetches a packet from Q1 or Q2 in every interval  $T$ , and the probability of *Sel* fetching a NOC packet from Q2 is  $p_1$ , so the service rate of *Sel* for the NOC packets is  $p_1/T$ . Then we can get the average service time of *Sel* for the NOC packets, denoted by  $T_{NOC}$ , is:

$$T_{NOC} = T/p_1; \quad (4-11)$$

The original arrival rate of the NOC packets is  $\lambda_2$ . When Q0 is not full, no packets will be dropped. When Q0 is full, the NOC packets start to queue in Q2 and some of the packets will be dropped according to the dropping scheme. Dropping some packets can be considered as the decreasing of arrival rates. As a result, the arrival rate of the NOC packets  $\lambda_2'$  changes according to the number of packets in Q2, which is

described in Section 4.1.2. So that the offered load of  $Sel$  for the NOC packets, denoted by  $\rho_2'(i)$ ,  $i$  is the number of number of NOC packets in the system, is:

$$\rho_2'(i) = \begin{cases} \lambda_2 \cdot T_{NOC} = \lambda_2 \cdot T / p_1 \cdots \cdots i < N_{Q0} \cdot p_1 \\ \lambda_2 \cdot T_{NOC} = \lambda_2 \cdot T / p_1 \cdots \cdots N_{Q0} \cdot p_1 \leq i \leq N_{Q0} \cdot p_1 + N_{Q2} \end{cases} \quad (4-12);$$

So we can summarize this model. It is an M/D/1 queuing system. Service time is a constant  $T_{NOC}=T/p_1$ . The queue consists of two parts. The size of first part of the queue is the number of NOC packets in  $Q_0$  when  $Q_0$  is full, that is  $N_{Q0} \cdot p_1$ , from equation 4-9. When this part of queue is not full, no packets will be dropped. The second part of the queue is  $Q_2$  which is a dynamic queue and its size is  $N_{Q2}$ . If packets arrive and queue in this part of queue, some of the packets will be dropped according to the dropping scheme. The size of the entire waiting queue is  $N_{Q0} \cdot p_1 + N_{Q2}$ . The offered load of the traffic is from Equation 4-12. Taking the packet in the server  $H$  into account, the maximum number of packets in the system, denoted by  $N$ , is

$$N = N_{Q0} \cdot p_1 + N_{Q2} + 1 \quad (4-13).$$

To compute the average waiting time and loss rate from this model, we utilized the calculation methods from the algorithm of finite capacity M/D/1 queues by Olivier Brun and Jean-Marie Garcia. [18]

Let  $X(t)$  be the number of packets in the system at time  $t$ . Let  $t_n$  be the time of the  $n$ th packet departure.  $N$  is the maximum number of packets in the system, and at the time of one packet departure, the maximum number of packets in the system is  $N-1$ , so the stochastic process  $\{X(t_n)\}$  ( $n \geq 0$ ) can be described as a finite state machine with state space  $\{0, 1, 2, \dots, N-1\}$ . Let  $Q = \{q_0, q_1, \dots, q_{N-1}\}$  represent the stationary distribution of the probability that 0, 1...  $N-1$  packets are left in the system when a service is finished. And  $\alpha(i, j)$  denote the probability of  $i$  packets arriving during a packet service period while  $j$  customers have already existed in the queue.

So we can get stationary equations:

$$\begin{aligned} \alpha(0,0) \cdot q_0 + \alpha(0,1) \cdot q_1 &= q_0 \\ \alpha(1,0) \cdot q_0 + \alpha(1,1) \cdot q_1 + \alpha(0,2) \cdot q_2 &= q_1 \\ \alpha(2,0) \cdot q_0 + \alpha(2,1) \cdot q_1 + \alpha(1,2) \cdot q_2 + \alpha(0,3) \cdot q_3 &= q_2 \\ &\vdots \\ &\vdots \\ &\vdots \end{aligned}$$

$$\alpha(N-2,0) \cdot q_0 + \alpha(N-2,1) \cdot q_1 + \dots + \alpha(0,N-1) \cdot q_{N-1} = q_{N-2} \quad (4-14)$$

If we know the expression of  $\alpha(i, j)$ , this is a linear system of  $N-1$  equations involving the  $N$  unknowns  $q_0, q_1, \dots, q_{N-1}$ . Hence, it allows us to express the probabilities  $q_1, q_2, \dots, q_{N-1}$  in terms of  $q_0$ . Let  $\{a_0, a_1, \dots, a_{N-1}\}$  be such that

$$q_n = a_n \cdot q_0 \quad \text{for all } n \geq 0. \quad (4-15)$$

It is easy to see that  $a_0 = 1$  ( $q_0 = a_0 \cdot q_0 \Rightarrow a_0 = 1$ ),  $a_1 = [1 - \alpha(0,0)] / \alpha(0,1)$  and that  $a_2, a_3, \dots, a_{N-1}$  obey the following recursion:

$$a_n = (a_{n-1} - \sum_{i=1}^{n-1} \alpha(i, n-i) \cdot a_{n-i} - \alpha(n-1,0)a_0) / \alpha(0,i) \quad (4-16)$$

So if we know the mathematical expression of  $\alpha(i, j)$ , we can easily get the  $a_2, a_3, \dots, a_{N-1}$  from the recursion equation (4-16).

The NOC packets are arriving according to a Poisson Process and wait in a queue with two parts. When the number of packets in the queue is less than  $N\_Q0\_NOC$  (the number of NOC packets when  $Q0$  is full), the arrival rate is  $\lambda_2$ . The traffic load of  $Sel$  is  $\rho_2'$ . From the formula of Poisson Process, we get if  $i + j \leq N\_Q0 \cdot p_1$

$$\alpha(i, j) = \frac{\rho_2'^i}{i!} e^{-\rho_2'} \quad (4-17)$$

If  $i + j > N\_Q0 \cdot p_1$ , the incoming packets will wait in  $Q2$  first, and some of them might be dropped. This is considered as the arrival rate of the NOC packets varies according to the number of packets already existed in  $Q2$ . This situation is a non-homogenous Poisson Process. The calculation is complicated, and we can not give the explicit mathematical expression to find  $\alpha(i, j)$  in this condition. However, I suggest an algorithm to estimate  $\alpha(i, j)$  when packets queue in  $Q2$ . The discussion of this estimation formula is given in Appendix B: Discussion of a Special Case of Non-Homogeneous Poisson Process.

Suppose  $i$  packets arrive in different rates, and the traffic load when each packet arrive is  $\rho_1, \rho_2, \dots, \rho_i$ , that is, the first packet arrives at the traffic load  $\rho_1$ , the second packet arrives at the traffic load  $\rho_2$ , and the  $i$ th packet arrives at the traffic load  $\rho_i$ . We calculate the Geometric mean of those  $i$  traffic loads

$$\rho_g = \sqrt[i]{\rho_1 \cdot \rho_2 \cdot \dots \cdot \rho_i}; \quad (4-18)$$

Assuming  $\rho_{i+1}$  is the traffic load when the  $(i+1)$ th packet is supposed arriving, we

calculate the Arithmetic mean of those  $i+1$  traffic loads,

$$\rho_a = (\rho_1 + \rho_2 + \dots + \rho_{i+1}) / (i+1) \quad (4-19)$$

Then we can get an estimation of  $\alpha(i, j)$  when  $i + j > N - Q_0 \cdot p_1$ :

$$\alpha(i, j) = \frac{\rho_a^i}{i!} e^{-\rho_a} \quad (4-20)$$

From equations (4-16), (4-17), (4-18), (4-19), and (4-20), using Matlab, we can calculate the value of  $a_0, a_1, \dots, a_{N-1}$ , and it is known that

$$q_0 + q_1 + \dots + q_{N-1} = 1 \quad (4-21)$$

So given the values of  $a_0, a_1, \dots, a_{N-1}$ , we can get  $\{q_0, q_1, \dots, q_{N-1}\}$  from the Equations 4-15 and 4-21.

We are not only interested in the state probabilities of the number of packets in the system when a packet departs from the system. In order to calculate the average waiting time of the NOC packets and the loss rate, we should also calculate the state probabilities of the number of packets in the system when a packet arrives at the system, which is denoted by  $\{p_0, p_1, p_2, \dots, p_N\}$ .

By the theorem given by Robert B. Cooper in Section 5.3 and 5.9 of Introduction to Queuing Theory (third edition) [19],  $p_i$  and  $q_i$  are proportional when  $i < N$ .

That is, when  $i < N$ ,

$$p_i = c \cdot q_i \quad (4-22)$$

where  $c$  is a constant.

It remains to find the equation that permits the calculation of the probability  $p_N$  that an incoming packet finds all the  $N$  waiting positions occupied. The constant of proportionality  $c$  is then calculated from the normalization equation

$$p_0 + p_1 + \dots + p_{N-1} + p_N = 1 \quad (4-23)$$

Combination of (4-21), (4-22) and (4-23) yields:

$$p_N = 1 - c \quad (4-24)$$

And the carried load  $\rho_c$  (the mean number of busy servers) is

$$\rho_c = p_1 + \dots + p_{N-1} + p_N = 1 - p_0; \quad (4-25)$$

The loss rate of the system can be calculated by carried load and offered load:

$$L_r = 1 - \rho_c / \rho 2' = 1 - (1 - p_0) / \rho 2' = 1 - (1 - c \cdot q_0) / \rho 2'; \quad (4-26)$$

We can calculate  $L_r$  in other way. The size of Q2 is  $N\_Q2$ , when a packet arrives in the system state  $p_{N-N\_Q2}, p_{N-N\_Q2+1}, \dots, p_N$ , it will be dropped with some probability. The drop rate of each state can be calculated according to the dropping scheme in Section 4.1.2. If a NOC packet arrives when the system state is  $p_N$ , it will be lost. Assuming the drop rate in system state  $p_i$  is  $d_i$ , and the loss rate of the system  $L_r$  can be calculated from:

$$L_r = p_N + \sum_{i=N-N\_Q2}^{N-1} p_i \cdot d_i = p_N + \sum_{i=N-N\_Q2}^{N-1} c \cdot q_i \cdot d_i \quad (4-27)$$

Combination of Equations 4-24, 4-26 and 4-27 yields

$$c = \frac{1}{q_0 + \rho 2' - \rho 2' \cdot \sum_{i=N-N\_Q2}^{N-1} q_i d_i} \quad (4-28)$$

From Equations 4-22, 4-24, and 4-28, we can get  $\{p_0, p_1, p_2, \dots, p_N\}$ .

Finally we calculate the mean number of packets in the queue, average waiting time of the NOC packets and system loss rate from state probabilities  $\{p_0, p_1, p_2, \dots, p_N\}$  which denote the probabilities of different number of packets in the system when a packet arrives at the node.

Loss rate is calculated from Equation 4-26.

The Mean number of the NOC packets in the system, denoted by  $N_{NOC}$ , is:

$$N_{NOC} = \sum_{k=1}^N p_k \cdot k \quad (4-29)$$

The average waiting time of the NOC packets, denoted by  $T_w$  is:

$$T_w = N_{NOC} \cdot T\_NOC = N_{NOC} \cdot T / p1 = \left( \sum_{k=1}^N p_k \cdot k \right) \cdot T / p1 \quad (4-30)$$

The average waiting time of the NOC packets is the average delay of a NOC packet in this network node.

### 4.3.2 The Model M/B/1 with Dynamic Queue

This model is developed on basis of M/D/1 with dynamic queue model in Section 4.3.1. It seems more precise to describe the service process than M/D/1 with dynamic queue model when the traffic load is high and the queue Q0 is always full.

In this model, we do not assume that the service time of *Sel* for the NOC packets is a constant, but we regard the time interval between two NOC packets being fetched by the server *H* as the service time of the first NOC packet. Because the time interval of two NOC packets being fetched is determined by how many Control packets being fetched by *Sel* between the two NOC packets and the service time of one packet. Because the traffic rate is high and the service time in *H* for every packet is a constant *T*, *Sel* fetches a packet every interval *T*. If there is no Control packet being fetched by *Sel* between two NOC packets, the service time of the first NOC packet is *T*; if there is 1 Control packet being fetched by *Sel* between two NOC packets, the service time of the first NOC packet is 2*T*, and if there are *i* Control packets being fetched by *Sel* between two NOC packets, the service time of the first NOC packet is (*i*+1)*T*.

Because the selector *Sel* fetches packets from Q2 with probability *p*<sub>1</sub>, and fetches packets from Q1 with probability 1-*p*<sub>1</sub>, when a NOC packet is fetched by *Sel*, the probability of *Sel* fetching a NOC packet next is *p*<sub>1</sub>, in which case the service time of the first NOC packet is *T*; the probability of *Sel* fetching a Control packet next and then fetching a NOC packet is (1-*p*<sub>1</sub>)·*p*<sub>1</sub>, in which case the service time of the first NOC packet is 2*T*; the probability of *Sel* fetching continuously *k* Control packets next and then fetching a NOC packet is (1-*p*<sub>1</sub>)<sup>*k*</sup>·*p*<sub>1</sub>, in which case the service time of the first NOC packet is (*k*+1)·*T*. The probability distribution of the random event *X*(*k*) that there are continuously *k* Control packets being fetched by *Sel* during a period and then a NOC packet being fetched is a Bernoulli distribution, so we call it M/B/1 model. The mean value of service time of the NOC packets in this model is

$$T_{NOC} = \sum_{k=0}^{+\infty} (1-p_1)^k \cdot p_1 \cdot (k+1) \cdot T = T / p_1,$$

which is equal to the service time of the NOC packets in M/D/1 with dynamic queue model.

Similar with the M/D/1 with dynamic queue model, the queue consists of two parts. The size of first part of the queue is the number of NOC packets in Q0 when Q0 is full, that is *N*<sub>Q0</sub>·*p*<sub>1</sub>, and the maximum number of NOC packets in the system, denoted by *N* is *N* = *N*<sub>Q0</sub>·*p*<sub>1</sub> + *N*<sub>Q2</sub> + 1.

In M/B/1 with dynamic queue model, the offered load of  $Sel$  for the NOC packets, denoted by  $\rho_2'(i)$ ,  $i$  is the number of packets already in the system, is also

$$\rho_2'(i) = \begin{cases} \lambda_2 \cdot T_{NOC} = \lambda_2 \cdot T / p_1 \cdots \cdots i < N_{Q0} \cdot p_1 \\ \lambda_2' \cdot T_{NOC} = \lambda_2' \cdot T / p_1 \cdots \cdots N_{Q0} \cdot p_1 \leq i \leq N_{Q0} \cdot p_1 + N_{Q2} \end{cases}$$

The offered load of the NOC packets in  $H$ , denoted by  $\rho_2$  is also useful in the calculation of  $\alpha(i, j)$  which denote the probability of  $i$  arrivals during a packet service period while  $j$  packets have already been in the system.

$$\rho_2(i) = \begin{cases} \lambda_2 \cdot T = \lambda_2 \cdot T \cdots \cdots i < N_{Q0} \cdot p_1 \\ \lambda_2' \cdot T = \lambda_2' \cdot T \cdots \cdots N_{Q0} \cdot p_1 \leq i \leq N_{Q0} \cdot p_1 + N_{Q2} \end{cases} \quad (4-31)$$

The mathematical calculation in the M/D/1 with dynamic queue model and the M/B/1 with dynamic queue model are almost the same except the calculation of  $\alpha(i, j)$

$$\text{In the M/D/1 model, } \alpha(i, j) = \begin{cases} \frac{\rho_2^i}{i!} e^{-\rho_2'} \cdots \cdots i + j < N_{Q0} \cdot p_1 \\ \frac{\rho_2'_g}{i!} e^{-\rho_2'_a} \cdots \cdots i + j \geq N_{Q0} \cdot p_1 \end{cases}$$

In the M/B/1 model, the service time of the NOC packets is not constant but follows Bernoulli distribution. The probability of continuously  $k$  Control packets are fetched by  $Sel$  between two NOC packets is  $(1-p_1)^k \cdot p_1$ , in which case the service time of the first NOC packet is  $(k+1) \cdot T$ . The probability of  $i$  packets have arrived during that period is  $[\rho_2 \cdot (k+1)]^i \cdot e^{-\rho_2 \cdot (k+1)} / i!$ , so we calculate  $\alpha(i, j)$  like this:

$$\alpha(i, j) = \begin{cases} \sum_{k=0}^{+\infty} \frac{[\rho_2 \cdot (k+1)]^i}{i!} e^{-\rho_2 \cdot (k+1)} \cdot (1-p_1)^k \cdot p_1 \cdots \cdots i + j < N_{Q0} \cdot p_1 \\ \sum_{k=0}^{+\infty} \frac{[\rho_2'_g \cdot (k+1)]^i}{i!} e^{-\rho_2'_a \cdot (k+1)} \cdot (1-p_1)^k \cdot p_1 \cdots \cdots i + j \geq N_{Q0} \cdot p_1 \end{cases} \quad (4-31)$$

In which  $\rho_2$  is the original traffic load of the NOC packets in  $H$  when no packets queue in  $Q_2$ . When the packets start to queue in  $Q_2$ , that is  $i + j \geq N_{Q0} \cdot p_1$ , some packets will be dropped and the traffic load of  $H$   $\rho_2$  varies because the real arrival rate of the NOC packets  $\lambda_2'$  is changing when the number of packets in  $Q_2$  changes.  $\rho_2'_g$  and  $\rho_2'_a$  are the Geometric mean and Arithmetic mean of  $\rho_2$  which are calculated from equations 4-18 and 4-19.

The following steps are all the same with the steps in M/D/1 model:

$$a_n = (a_{n-1} - \sum_{i=1}^{n-1} \alpha(i, n-i) * a_{n-i} - \alpha(n-1, 0) a_0) / \alpha(0, i) \quad (4-32)$$

$$q_n = a_n \cdot q_0 \quad (4-33)$$

$$q_0 + q_1 + \dots + q_{N-1} = 1 \quad (4-34)$$

$$p_i = c \cdot q_i \quad i < N \quad (4-35)$$

$$p_N = 1 - c \quad (4-36)$$

$$c = \frac{1}{q_0 + \rho 2^1 - \rho 2^1 \cdot \sum_{i=N-k}^{N-1} q_i d_i} \quad (4-37)$$

where  $d_i$  is the drop rate of system state  $p_i$

$$L_r = 1 - (1 - c \cdot q_0) / \rho \quad (4-38)$$

$$T_w = \left( \sum_{k=1}^N p_k \cdot k \right) \cdot T / p1 \quad (4-39)$$

From the above equations, we can calculate the loss rate  $L_r$  and the mean delay of the NOC packets  $T_w$  using Matlab.

#### 4.4 Verify the analytic models by *NodeSim*

The output link queuing system of DMP node can be simulated by running *NodeSim*. First, we set the size of Q0 is 1000, and p1 is 0.7. We define those parameters: the packet generators are Poisson process. The expected traffic rate of the Control packets is  $0.3 \times 10^6$  per second, and we set different value of the expected traffic rate of the NOC packets. The service time for one Control packet or one NOC packet is  $1.0 \times 10^{-6}$  second. So the offered load of the Control packets is  $\rho_c = 0.3$ , and the offered load of the NOC packets  $\rho_{NOC}$  varies.

The calculation is carried out on Matlab 7.0, and the scripts of calculation are list in Appendix C: The Matlab scripts.

Figure 4-2 shows the comparison of the average waiting time of the NOC packets from M/D/1 with infinite queue model and the average waiting time measured from the output file of *NodeSim*. We can see that when the traffic load of NOC packets is lower than 0.695, that is, the traffic load of total packets is 0.995, M/D/1 with infinite queue model is perfectly consistent with the simulation result. In this case, the loss rate is always zero.

Figure 4-3 shows the comparison of the average waiting time of the NOC packets



from the two models M/D/1 with dynamic queue model and M/B/1 with dynamic queue model, and those measured from the output file of *NodeSim*, when the traffic load of the NOC packets is from 0.7 to 1.1, that is, the traffic load of total packets is from 1.0 to 1.4. We can see both of the two models are consistent well with the simulation result, and M/B/1 with dynamic queue model is better than M/D/1 with dynamic queue model. Figure 4-4 shows the comparison of the loss rate of the NOC packets from the two mathematic models and from the simulation result and both of them are consistent with the simulation result perfectly.

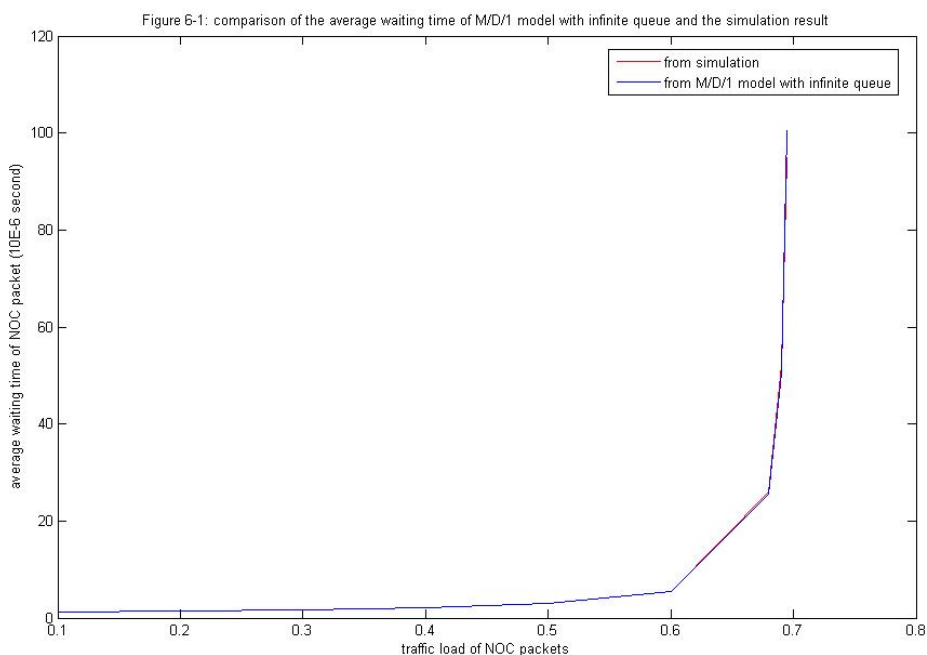


Figure 4-2: comparison of the average waiting time of M/D/1 with infinite queue model and the simulation result

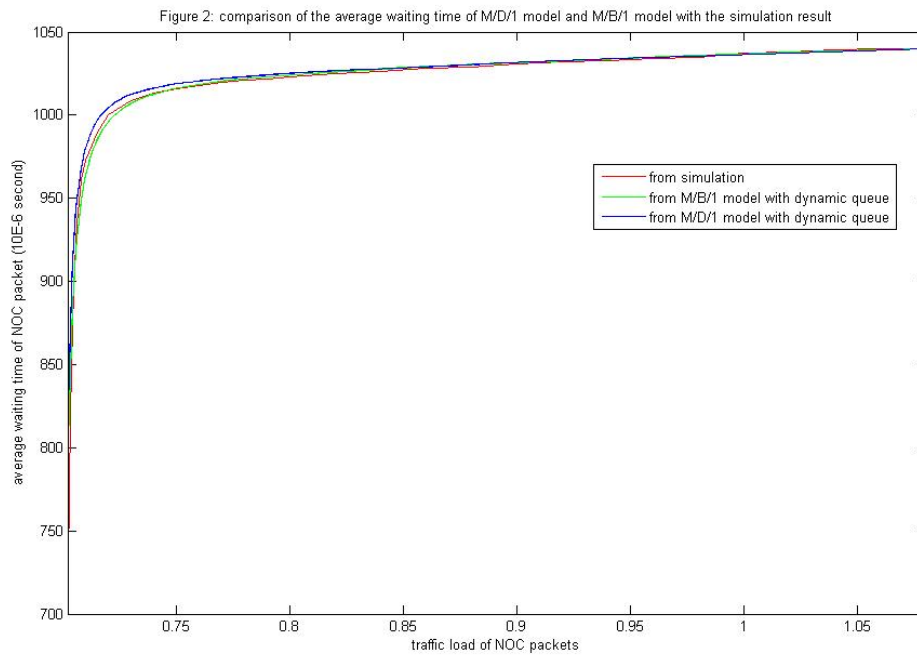


Figure 4-3: comparison of the average waiting time from M/D/1 with dynamic queue model, M/B/1 with dynamic queue model, and from the simulation application, while  $N_Q=1000$ ,  $p=0.7$ .

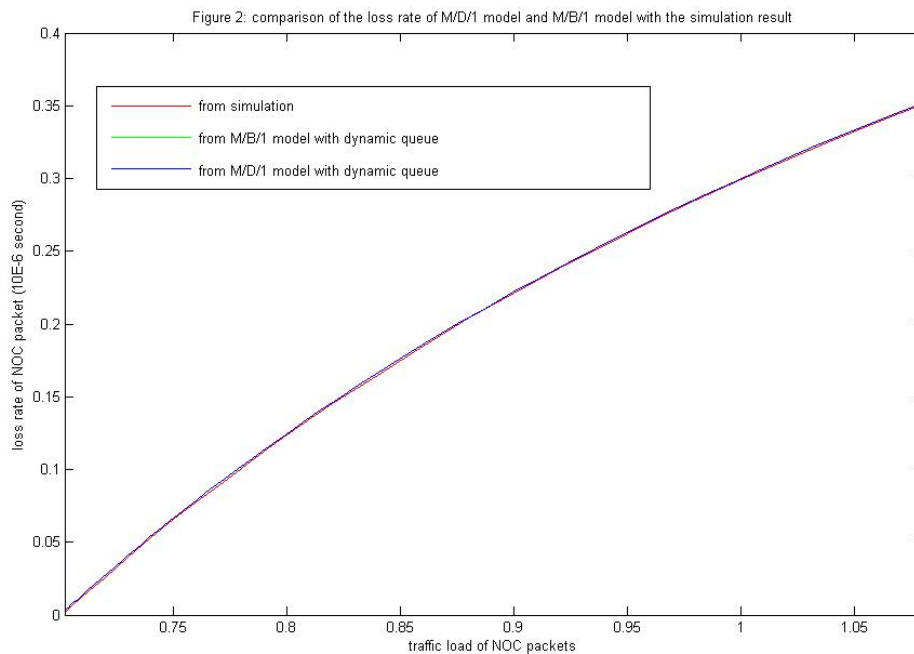


Figure 4-4: Comparison of the loss rate from M/D/1 with dynamic queue model, M/B/1 with dynamic queue model, and from the simulation application.  $N_Q=1000$ ,  $p=0.7$

We change the size of  $Q_0$  and the value of  $p_1$  and run the simulation and calculation again. Then we compare the result from output file with the calculation result. We change the size of  $Q_0$  into 1500,  $p_1$  is kept the same 0.7, the service time and carried load of the Control packets are not changed,  $\rho_c = 0.3$ , and the carried load of the NOC packets  $\rho_{NOC}$  varies. Figure 4-5 shows the comparison of the average waiting time of the NOC packets from the two mathematical models and those measured from *NodeSim*. Figure 4-6 shows the comparison of the loss rate of the NOC packets in this condition.

Then we change  $p_1$ . We set  $p_1=0.5$ ,  $Q_0$  size 1000, service time is the same, but change the traffic load of the Control packets. We set  $\rho_c = 0.5$ . Figure 4-7 shows the average waiting time of the NOC packets from the two mathematical models and those measured from *NodeSim* in different traffic load of the NOC packet. Figure 4-8 shows the loss rate of the NOC packets according to different traffic load of the NOC packet.

From those comparisons of simulation and analytic models, we can draw the conclusion that the mathematic models work well in different situations.

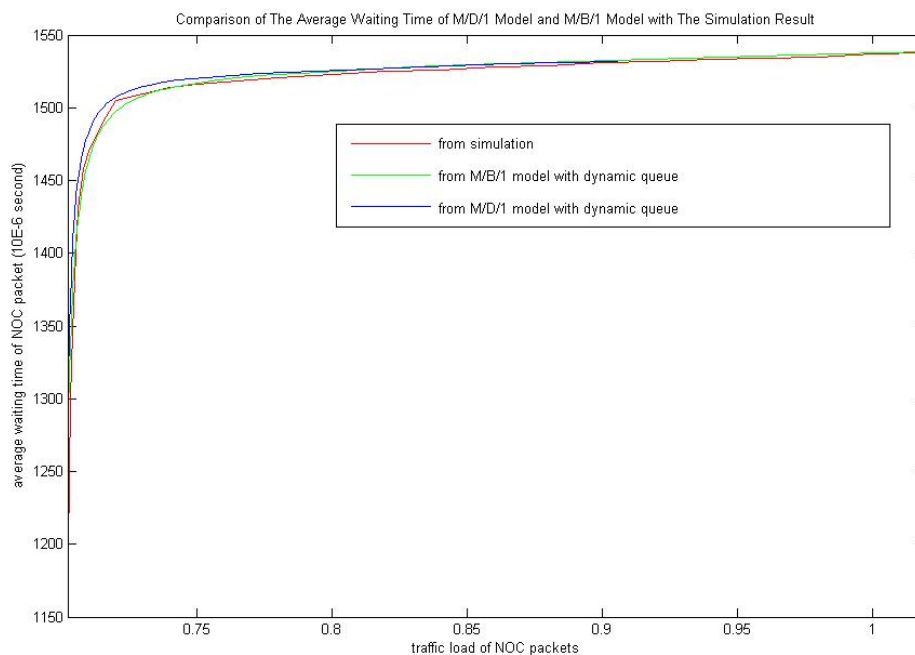


Figure 4-5: comparison of the average waiting time from M/D/1 with dynamic queue model, M/B/1 with dynamic queue model, and from the simulation, while  $N_{Q_0}=1500$ ,  $p_1=0.7$

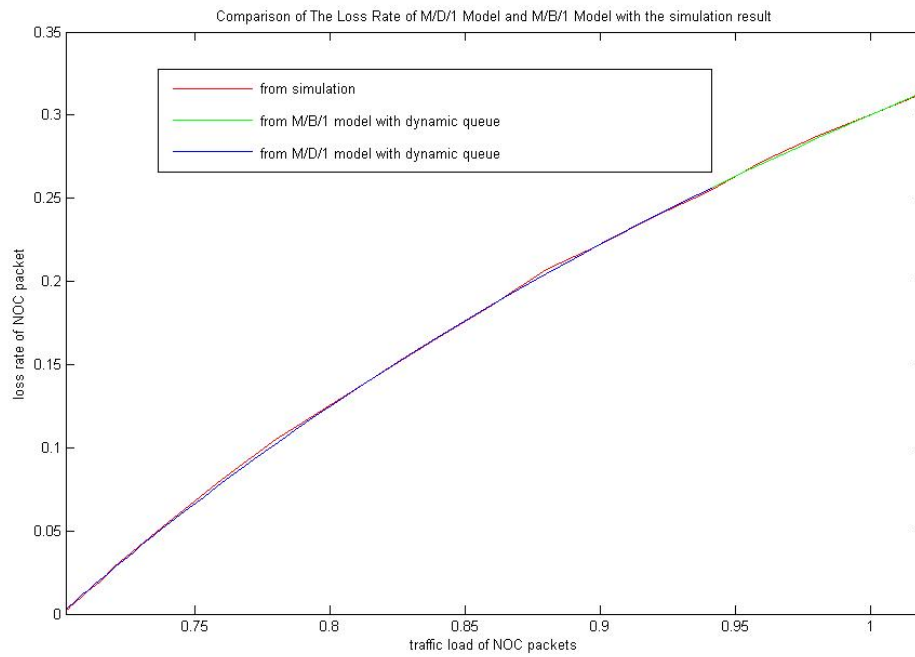


Figure 4-6: Comparison of the loss rate from M/D/1 with dynamic queue model, M/B/1 with dynamic queue model, and from *NodeSim*, while  $N\_Q0=1500$ ,  $p1=0.7$

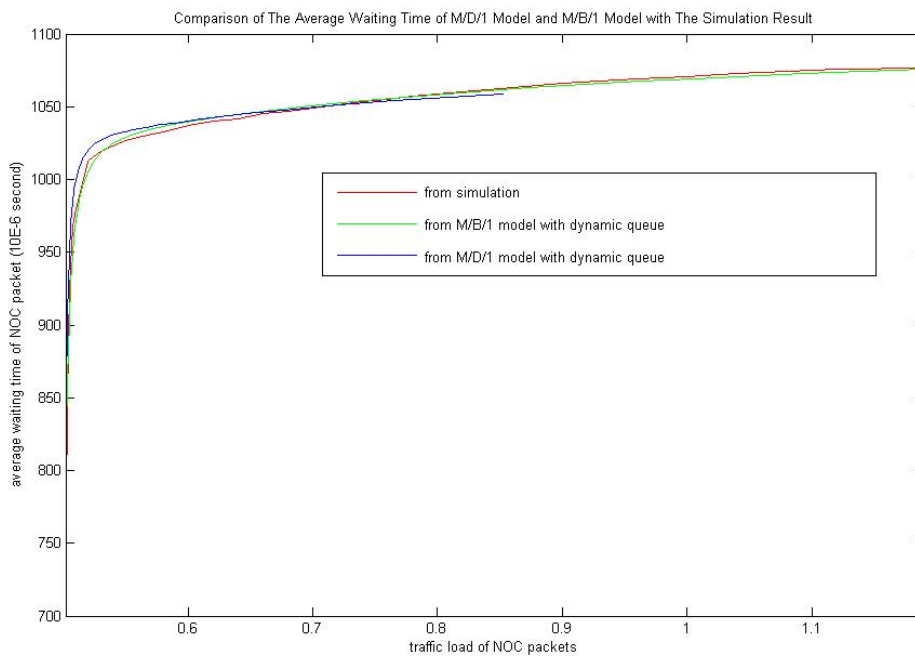


Figure 4-7: Comparison of the average waiting time from M/D/1 with dynamic queue model, M/B/1 with dynamic queue model, and from the *NodeSim*, while  $N\_Q0=1000$ ,  $p1=0.5$

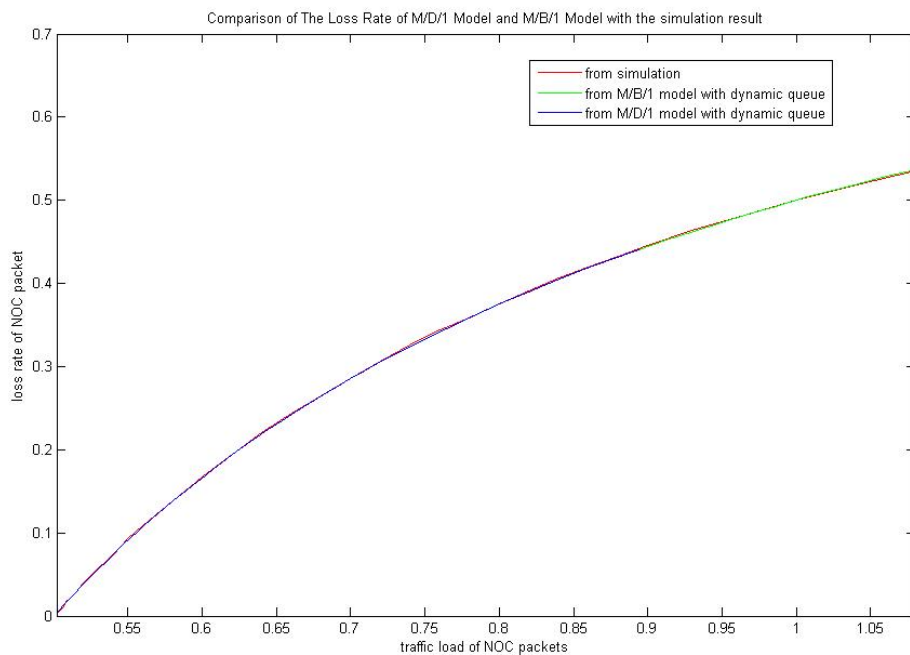


Figure 4-8: Comparison of the loss rate from M/D/1 with dynamic queue model, M/B/1 with dynamic queue model, and from *NodeSim*, while  $N_Q=1000$ ,  $p_1=0.5$ ;



## Chapter 5

# Analytic Model for a Multi-Collaboration

In Chapter 4, analytic models are given to analyze the delay and loss rate of the packets in one DMP node. In this chapter, we describe how to set up a multiparty DMP collaboration. Security issues are presented and Analytic models are given to compute the delay and loss rate of packets in a DMP collaboration. Several experiments on simulator *RouteSim* will also be present to explore the performance of the network nodes with different traffic interest in a DMP collaboration.

### 5.1 Establishment of a Multiparty DMP Collaboration

A DMP collaboration can be established between several users. When there are more than two users in one DMP collaboration, all users have to set up a one-way collaboration to all the other users.

DMP collaborations can be established between users in deferent places, maybe in different countries. In this section, we first review the topologies of the DMP network, and then introduce an example of a DMP collaboration setup.

#### 5.1.1 DMP Network Topologies

DMP networks are hierarchical, built with combinations of star and mesh topologies. The basic rule is that the DMP network uses fixed routes, because this is necessary to guarantee delays. However, alternative routes should be provided for a processing part or link goes down, and for load sharing of links between nodes. [1]

The alternate routes give the same number of hops with the main routes. In some

cases, when the traffic interests between two cities of two different European regions are high, a shorter path can be used, and shall be used until it is saturated. Then the main route takes over.

To provide a DMP service to all inhabitants in Europe, a hierarchical topological and addressing structure is proposed by Prof Leif Arne Rønningen, with the following node levels.

Node	IPv6 address, (Format prefix)	bits
• GlobalNode, <b>G</b>	(TLA)	13
• EuropeNode, <b>E</b>	(reserved)	8
• DistrictNode, <b>D</b>	(NLA1)	12
• CityNode, <b>C</b>	(NLA2)	12
• VillageNode, <b>V</b>	(SLA)	16
• AccessNode, <b>F</b>		16
• AccessLink, <b>FL</b>		48

The GlobalNode routes traffic to or from areas outside Europe, while a EuropeNode handles one European region, and traffic to or from other European regions. Within Europe, the maximum number of hops between two users is 11, but depending on traffic interests, shorter routes can be defined as suggested in the figures below. The example structure up to European level is shown in Figure 5-1.

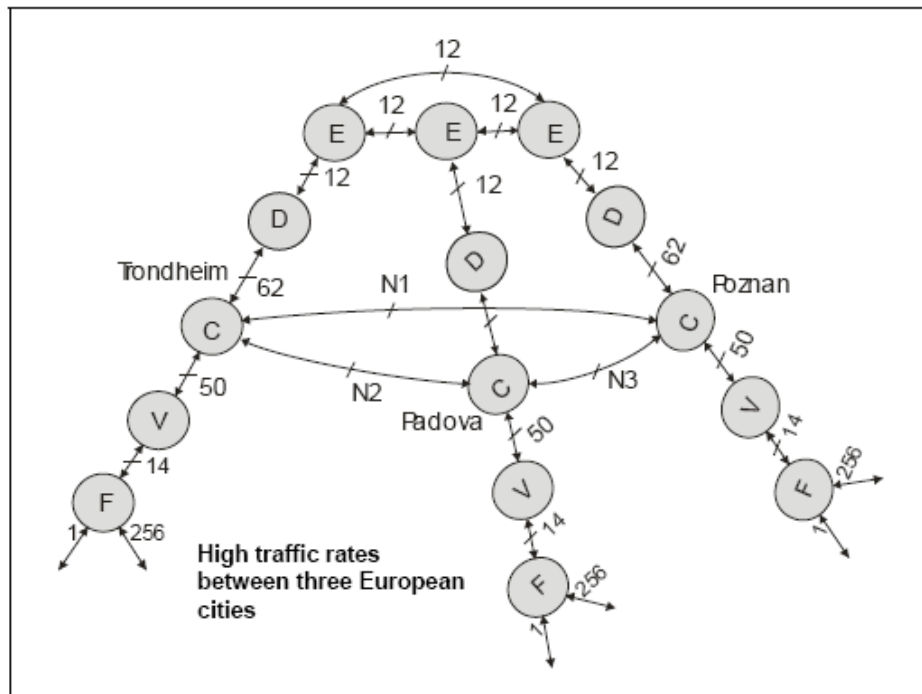


Figure 5-1: Network between three European cities. [1]



### 5.1.2 ServiceControl and Security of DMP collaborations

The ServiceControl is one of the main functional blocks on the DMP application layer for the user scene. It has the top responsibility for setting up, managing, and releasing collaborations between users, on request from a user. The Service Control manages and routes the packet exchange between users and various servers allocated to Access Nodes, and between access and destination network nodes. The ServiceControl has a number of supporting servers to its disposal: [1]

**QualityControl:** handles scene quality shaping by dropping sub-objects.

**Security Server:** implements IPsec AH and ESP, receives authentication requests from ServiceControl, returns acceptance or not. Encrypts and decrypts payload.

**Proxy and Address Server:** receives translation requests from the Service Control, returns acceptance or not, and stores information about on-going services, user IDs and address relations in the domain. This Server also handles adaptation of protocols and formats for other ITC systems, such as WWW and email.

**SceneProfile and QualityShapingProfile Server:** receives SceneProfile check request from ServiceControl, returns acceptance or not. Stores standardized SceneProfiles. Stores standardized QualityShaping Profiles. Supports the Service Control to adapt scene quality according to traffic load, using the optimal Quality-ShapingProfile.

The security of the DMP collaborations is quite important. The access management, connectionless integrity, data origin authentication, and the confidentiality of the content should be safely achieved.

All of the packets are all encrypted and integrity protected by shared keys. We use IPSec AH and ESP [17] [20] [21] to encrypted the payload and protect the integrity. The user authentication and key exchange process are handled by Security Server in every Access Node, using IKE protocol [22].

### 5.1.3 Establishment of a Multiparty DMP Collaboration

In a bidirectional collaboration of two users, two one-way collaborations should be set up between them to support the bidirectional communication. When there are  $N$  users in a collaboration, all users have to set up a one-way collaboration to all the others, giving  $N \cdot (N - 1)$  set-ups. The ServiceControls in the Access Nodes handle the setup and the management of the sessions until they are released. All servers and users are uniquely identified by their IP addresses. All possible services are described by standardized SceneProfiles. Users must choose SceneProfiles during set-up of a service, but this can be changed during the session. The example of how to set up a DMP collaboration can be found in [1]. We do not go into detail in this report.

## 5.2 Analytic Models

Utilizing the analytic result of the delay and loss rate of packets in one DMP node in Chapter 4, we can analyze the traffic delays and loss rate of a DMP collaboration.

### 5.2.1 Delay of the NOC packets

The delays of the packets include two delay types. Propagation delays are the duration that the electronic signals spending on the wire, while the delays in the nodes are the waiting time of the packets in the queues of network nodes.

The speed of light is  $3 \times 10^6$  kilometers per second, denoted by  $c$ . If the distance between the two users of the collaboration is  $D$ , the propagation delay of the packets in this collaboration, denoted by  $T_p$ , is

$$T_p = D \cdot c \quad (5-1)$$

There are three waiting queues used in the output link of one network node. The NOC packets queue in Q2 and Q0. Because Q0 is much longer than Q2, the maximum length of waiting queue of the NOC packets can be estimated as the length of Q0. Assume the size of Q0 is  $N\_Q0$  and the link capacity of  $H$  is  $ht$  seconds per packet. That is, a packet leaves the system from  $H$  every  $ht$  seconds. The maximum delay of a packet in this node, denoted by  $T_{NOC\_1node\_max}$ , can be calculated as

$$T_{NOC\_1node\_max} = N\_Q0 \cdot ht \quad (5-2)$$

In a one-way collaboration of two users, if the route contains  $N_{nk}$  network nodes, and the processing time in the Source Host is  $T_{NOC\_sc}$  and the processing time in the Destination Host is  $T_{NOC\_rv}$ , The maximum total delay of the NOC packets in this collaboration, denoted by  $T_{NOC\_max}$ , is

$$T_{NOC\_max} = T_p + N_{nk} \cdot T_{NOC\_1node\_max} + T_{NOC\_sc} + T_{NOC\_rv} \quad (5-3)$$

Equations 5-2 and 5-3 yields

$$T_{NOC\_max} = D \cdot c + N\_Q0 \cdot ht + T_{NOC\_sc} + T_{NOC\_rv} \quad (5-4)$$

The minimum delay is when the queues in each network node are all empty:

$$T_{NOC\_min} = D \cdot c + T_{NOC\_sc} + T_{NOC\_rv} \quad (5-5)$$

### 5.2.2 Delay of the Control packets

The propagation delay of the Control packets is calculated in the same way with the propagation delay of the NOC packets. If the distance of the two users of the collaboration is  $D$ , the propagation delay of the packets in this collaboration, denoted by  $T_p$ , can be calculated from the equation 5-1.

The delay varies significantly in different nodes of the collaboration. When the traffic load is high, the incoming Control packets queue in Q1, the size of which is very large. As a result, we can not even estimate the maximum delay of the Control packets in the nodes. Nevertheless, we can calculate the delays in each network node in the collaboration according to their traffic interest. The summation of those delays in each node is the total delay of the Control packets in the collaboration.

If we know the traffic interest of each node, the delay of the Control packets in each node can be calculated by Equations 4-1, 4-2, 4-3, 4-4, 4-5 in Section 4.2 of this thesis.

When the traffic load of a node is low, the delay of the Control packet in this node is very small, and can even be neglected compared with the propagation delay and the delay in others. However, when the traffic load of a node is high, and if we do not set up the probability parameter  $p1$  properly, the packets waiting in Q1 may approach infinity, thus induce a significant delay of the Control packets pass through this node. So we have to adjust  $p1$  carefully and properly to avoid enormous delays of Control packets in this node.

### 5.2.3 The Loss Rate of the NOC packets

The NOC packets will be dropped by the network node when the traffic load of this node is high. The dropping scheme was described in Section 2.2.2.

Given the traffic load of NOC packets as well as the traffic load of Control packets in the network node, and the probability parameter  $p1$ , we can calculate the loss rate of the NOC packets in this node by Equation 5-29 in Section 5.3.2

The traffic interest varies from one node to another in the route of the collaboration. If the traffic load of a node is low, it will never drop any packet. The total loss rate should refer to every node in the route that drops packets.

Figure 5-2 is an example of a one-way collaboration. If the loss rate of network node 1 is  $L_{r,1}$ , the loss rate of network node 2 is  $L_{r,2}$ , the loss rate of network node  $i$  is  $L_{r,i}$ , and there are  $k$  network nodes in the route. The loss rate of the NOC packets from The Source Host in this collaboration, denoted by  $L_{r\_route}$ , can be calculated:

$$L_{r\_route} = 1 - \prod_{i=1}^k (1 - L_{ri}) \quad (5-6)$$

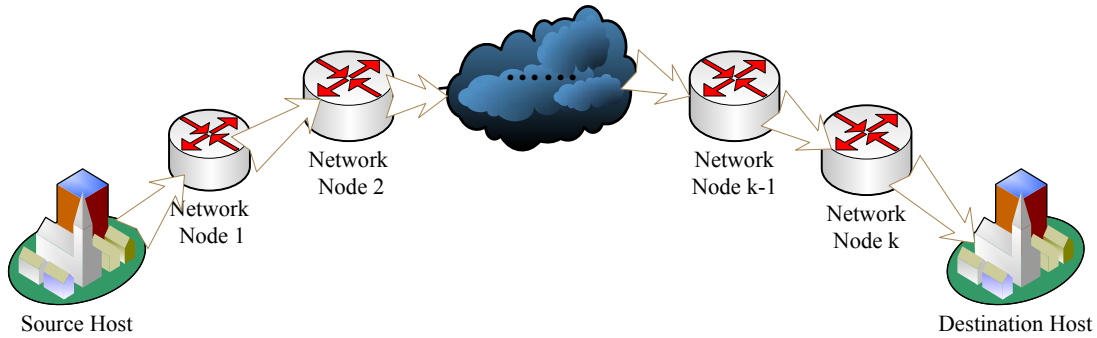


Figure 5-2: a one-way collaboration with 5 network nodes

If there is only one node in the route where the traffic load is higher than the threshold of dropping, while the traffic interest of other nodes are low or normal, the loss rate of the NOC packets from the Source Host would be the same as the loss rate of the NOC packets in this node.

If there are several nodes in the route where the traffic load is higher than the threshold of dropping, the loss rate of the NOC packets in this collaboration rises significantly. For example, if there are five nodes in the network dropping packets and the loss rate of the NOC packets in these nodes are all 5%, from Equation 5-6, we can get the loss rate of the NOC packets from the Source Host to the Destination Host is  $1 - (1 - 0.05)^5 = 22.6\%$ .

## 5.3 Experiments and Discussion

By executing *RouteSim*, we can do several experiments to explore the delay and loss rate of the packets in a DMP collaboration from the Source Host to the Destination Host, and their relationship with the link capacity of each network node.

### 5.3.1 Experiment 1: Delay and Loss Rate

In this experiment on *RouteSim*, we control the traffic interest of each network node by changing the traffic rate of the packets from other nodes. We observe the delay and loss rate of the NOC packets from the Source Host to the Destination Host when the offered load of the five network nodes is all more than 100%.

We set  $ht=10^{-6}$ ;  $N\_Q0=1000$ ; the traffic rate of the NOC packets from the Source Host is  $0.3 \times 10^5$ ; the traffic rate of the Control packets from the Source Host is  $2.7 \times 10^5$ ; The traffic rate of Control packets from the other nodes to each network node in the

path of collaboration is  $0.3 \times 10^5$ ; the offered load of the network node in the path of the collaboration exceed 100% means that the number of arrival packets exceed the processing capacity of network node, so some NOC packets must be dropped to lower the load of the node. We set the traffic rate of the NOC packet from  $6.28 \times 10^5$  to  $6.42 \times 10^5$ , observe the mean delay of the NOC packets and the loss rate from the output files of *RouteSim*. Table 5-1 shows the results.

Table 5-1: The mean delay and drop rate in different traffic interest of network nodes

Traffic rate of the NOC packets from the source	$6.28 \times 10^5$	$6.3 \times 10^5$	$6.32 \times 10^5$	$6.34 \times 10^5$	$6.36 \times 10^5$	$6.38 \times 10^5$	$6.4 \times 10^5$	$6.42 \times 10^5$
Offered load of each network node	99.8%	100%	100.2%	100.4%	100.6%	100.8%	101%	101.2%
Mean delay Of the NOC packets (ms)	1.0	1.4	2.6	4.1	4.5	4.7	4.8	4.8
Drop rate	0	0	0.14%	1.1%	2.3%	3.4%	4.3%	5.3%

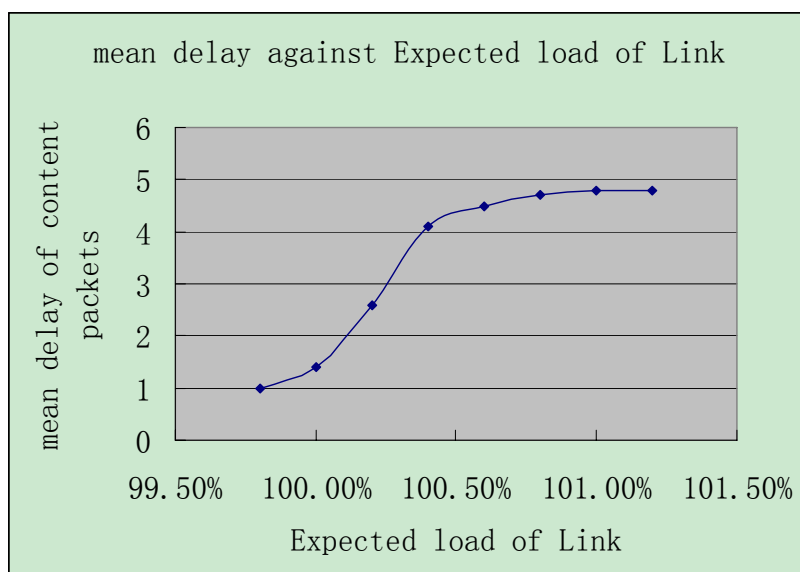


Figure 5-3: the mean delay of the NOC packets in different traffic interests.

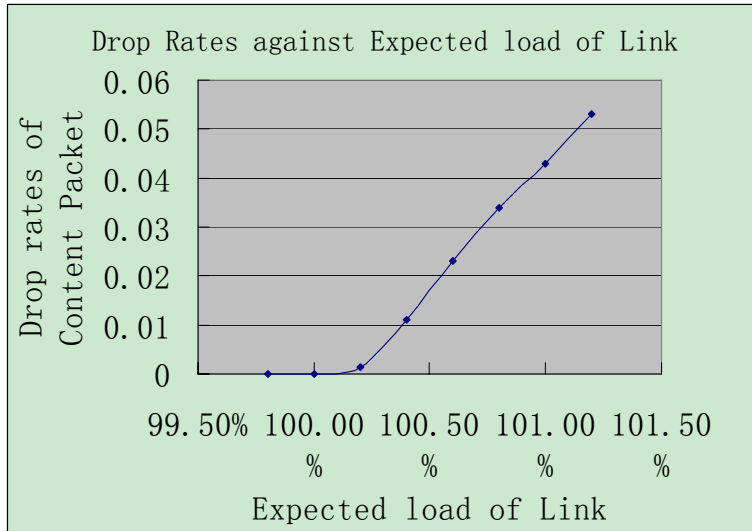


Figure 5-4: the loss rate of the NOC packets in different traffic interests.

From Figure 5-3 and 5-6, we can see that when the traffic interests of the network nodes become more and more heavy, the drop rate of the NOC packets from the Source Host increase dramatically. When the offered load of the five network nodes is 101%, the loss rate is 4.3%, which is nearly five times of the traffic load exceeding 100%. But the maximum delay is 4.8 ms in these experiments, which means the delay of the NOC packets can be controlled within an acceptable scale.

### 5.3.2 Experiment 2: The Link Capacity

In this Experiment, we want to find the relationship between processing capacity of the output port of network node and the drop rate when the offered load of the network nodes is 100%.

As we mentioned in Chapter 2, Distributed Multimedia Play requires delays as low as 10 ms for audio and 20 ms for video. The delay can be divided into two parts, one part is caused by the duration of light transmission in the wire, and the other part of delay comes from the waiting time and processing time in the network nodes. In the simulator RouteSim, the path of the collaboration consists of five network nodes, we suppose the distance between the Source Host and the Destination Host is about 1500 kilometers, which means the delay on the wire is about 5 ms (the speed of light divides the distance). To ensure the entire delay is less than 10 ms, the delay on the nodes should be less than 5 ms. There are five network nodes, so the maximum delay on each node is 1 ms. Because  $Q_0$  is much longer than  $Q_2$ , so the maximum length of waiting queue is approximately the length of  $Q_0$ . Supposing the length of  $Q_0$  is  $N_{Q_0}$  and packet processing time is  $T$ , the maximum delay in a node can be calculated as  $N_{Q_0} \cdot T$ . So we should ensure  $N_{Q_0} \cdot T \leq 1$  ms.

When we change the processing capacity of output port, we should also change the length of  $Q_0$  to ensure that  $N_{Q_0} \cdot T \leq 1$  ms. The Mean delay and drop rate of the

NOC packets when the traffic load is 100% from these experiments on *RouteSim* is shown in Table 5-2.

Table 5-2: The mean delay and drop rate in different link capacity when the traffic load is 100%

Link capacity (packets per second)	$10^5$	$2*10^5$	$3*10^5$	$4*10^5$	$5*10^5$	$6*10^5$	$7*10^5$	$8*10^5$	$9*10^5$	$10*10^5$
N_Q0	100	200	300	400	500	600	700	800	900	1000
Mean delay Of the NOC packets (ms)	3.2	2.7	2.4	2.1	2.0	1.8	1.7	1.6	1.5	1.4
Drop rate	2.4%	1.0%	0.58%	0.31%	0.17%	0.12%	0.05%	0.002%	0.002%	0

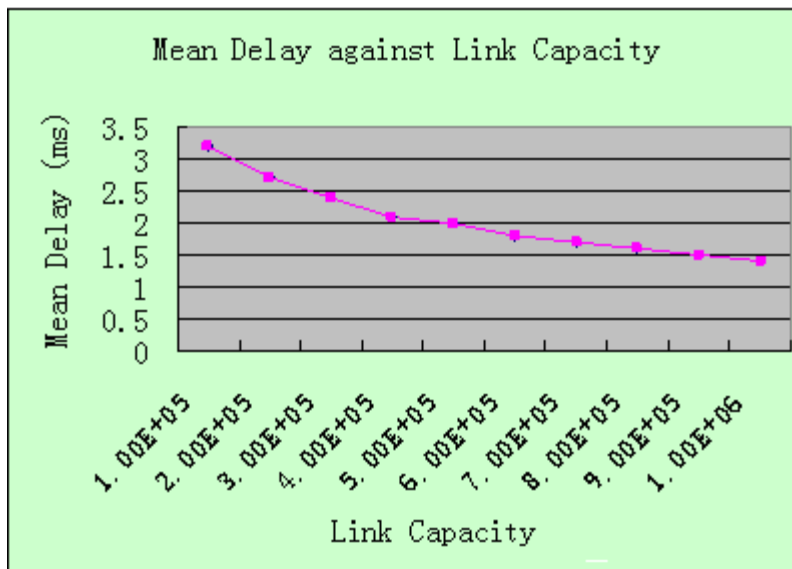


Figure 5-5: The mean delay against the link capacity when the traffic load is 100%

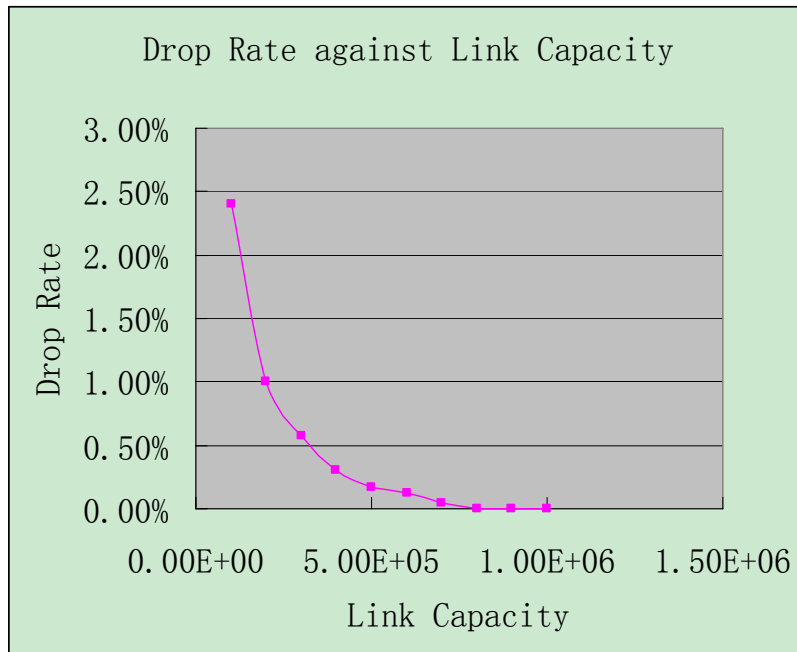


Figure 5-6: The loss rate against the link capacity when the traffic load is 100%

With a traditional non-adaptive video/audio encoder it was assumed that about 0.2% of the multimedia content could be dropped at the receiver without giving noticeable artifacts. [10] In this experiment, we can see if the link capacity is larger than  $5 \times 10^5$  packets per second, when the link utilization comes into 100%, the drop rate of the NOC packets is still lower than 0.2%, so to reach the link utilization 100%,  $5 \times 10^5$  packets per second is the minimum link capacity to support a DMP collaboration simulated by *RouteSim*.



## Chapter 6

# Guaranteed delay and reliable delivery using sub-obj seq. no 9

The packets delivered in DMP networks are classified into two types. The Control packets should be delivered reliably but its delay is tolerant, while the NOC packets require delay guaranteed but they can be dropped selectively.

There is a special kind of NOC packets that shall have delay guarantee, and they are so important that they should not be dropped in delivery. Those packets are assigned sequence number 9. In this chapter, we first give a brief introduction of the NOC coding scheme, then a updated version of output link queuing system is proposed to handle the NOC packets with sequence number 9. Finally we analyze the performance of the DMP network regarding this type of NOC packets.

### 6.1 Introduction of NOC

NOC (Near-natural Coding), uses the filtering and deflating compression principles from PNG [13], and adds new specifications. NOC can use up to 64 special sub-bands of the visible spectrum, each with a bit depth of up to 24 bits, and includes the PNG RGBA truecolour type, using a bit depth up to 16 bits per component.

In NOC, an image represents a sub-object of arbitrary shape, see Figure 2-2. To obtain a stereoscopic image, the object should be shot at least by two cameras. A large number of cameras have to be applied to obtain stereoscopic multi-view, depending on wanted quality. Four kinds of images are distinguished for NOC.

- **Source image:** The source image is the image available from the camera system. It must contain information necessary for the encoder to extract a reference image.

- **Reference image:** The reference image, which only exists inside the encoder, represents an arbitrary shaped area of pixels. It represents a sub-set of the source image, and can always be recovered from a NOC data stream. A reference image represents a sub-object. The sub-objects have arbitrary shape, and a shape mask with one bit per pixel, indicates with a '1' that a pixel belongs to the sub-object.
- **NOC image:** The NOC image is obtained from the reference image by a series of transformations. An encoder generates a NOC datastream from the NOC image. A decoder generates a NOC image from the datastream of a NOC packet.
- **Delivered image:** The delivered image is constructed by a decoder from a series of transformations applied to the NOC image.

The sequence number of the NOC packets is defined by the sub-object number. The sub-object number is given according to the drop priority of this object, the more important the sub-object is, the smaller number it will get. If all of the sub-objects of one object have the same drop priority, the sub-object number will be assigned to them circularly from 1 to 8. Some sub-objects require extremely low probability for being dropped because they play an essential role in reconstructing the delivered image, (or because of other reasons). Those sub-objects are given sub-object number 9. Thus, the sequence number of those NOC packets is 9.

Because the NOC packets with sequence number 9 require both guaranteed delay and extremely low loss probability, the number of these packets should be extremely small compared to the number of the NOC packets with other sequence numbers. Only the NOC packets that are very important will be assigned sequence number 9.

## 6.2 An Updated Version of Output Link Queuing System

Taking the NOC packets with sequence number 9 into account, the output link queuing system should be updated.

If the number of packets reaches the maximum length of Q2, in the output link queuing system introduced in Chapter 4 (see Figure 4-1), all of the incoming NOC packets with sequence number from 1 to 8 will be dropped. Because the number of the NOC packets with sequence number 9 is very small compared to the NOC packets with other sequence numbers, the NOC packets with sequence number 9 are not considered in that model to simplify the analysis.

In order to guarantee that the NOC packets with sequence number 9 are not being dropped, Q2 should be extended and it can accept the NOC packets with sequence number 9 when the traffic load is extremely high. Seeing Figure 6-1, Q2' is the

extended part of Q2. Only the NOC packets with sequence number 9 can wait in Q2'. That is, when a NOC packet arrives and the number of packets in Q2 has exceeded the original size of Q2, only the NOC packets with sequence number 9 will be accepted to queue in Q2' part, while all of the NOC packets with other sequence number will be dropped in this condition.

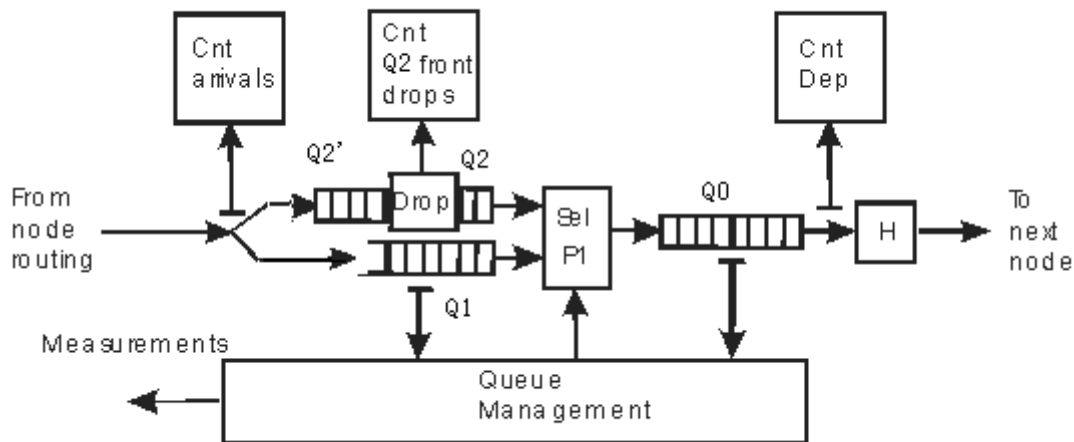


Figure 6-1: An updated version of output link queuing system of DMP node

Because the arrival rate of the NOC packets with sequence number 9 is rather small and spread events out in time, it is unlikely that a lot of NOC packets with sequence number 9 arrive during a very short time. The size of Q2' need not be large. If the original size of Q2 is  $N_{Q2}$  and the number of the NOC packets with sequence number 9 is one ninth of the total number of the NOC packets, the size of extended part Q2', denoted by  $N_{Q2'}$ , can be set

$$N_{Q2'} = N_{Q2} / 9 \quad (6-1)$$

In the analytic models in Chapter 4,  $N_{Q2}=50$ , so the length of extended part of Q2 can be set  $N_{Q2'}=6$ .

## 6.3 Analytic Model

The updated version of the output queuing system imports some new features to the analytic model. In this section, we analyze the delay and loss rate of the NOC packets in the new version of the output queuing system.

### 6.3.1 Delay Analysis

The delay of a NOC packet in a network node of DMP is the sum of its waiting times in Q0 and Q2 of this node. The maximum delay of the NOC packet is the product of processing time in  $H$  and the sum of Q0 size and Q2 size. Since the NOC packet with sequence number 1 to 8 can not wait in Q2', and the original size of Q2 is rather small compared to Q0, the waiting time in Q2 can be neglected compared to the waiting time in Q0.

The NOC packets with sequence number 9 may queue in Q2'. However, as we stated in Section 6.2, the size of extended part Q2' is also very small compared to the size of Q0. So if we calculate the delay roughly, the waiting time in Q2' of a NOC packet with sequence number 9 can also be neglected.

The delay of the NOC packets in a node is roughly the waiting times in Q0, and the maximum delay is roughly the product of processing time and the size of Q0. If we want to get the delay precisely, the analytic models we presented in 4.3 remain applicable for calculating the delay of the NOC packets in the updated version of output link queuing system in Figure 6-1.

The size of Q2 and the drop rate corresponding to different numbers of packets in Q2 should be modified. Nevertheless, all of the formulas and equations in Section 4.3 are still applicable in this condition. We can utilize Equations 4-32, 4-33, 4-34, 4-35, 4-36, 4-37, 4-38, and 4-39 to calculate the average waiting time of the NOC packets according to different traffic loads and the value of probability parameter  $p1$ .

### 6.3.2 Loss Rate

The NOC packets with sequence number 9 are never dropped, but other NOC packets might be dropped when the traffic load is high. The bigger the sequence number is, the more likely it will be dropped.

If the traffic load is low, there is no NOC packet being dropped. While the traffic load is high, the loss rate of the NOC packets can be calculated by utilizing Equations 4-22, 4-23, 4-24, 4-25, 4-26, 4-27, 4-28, 4-29, and 4-30 in Section 4.3.

Because the buffer Q0 is large, if the traffic load of the total packets is smaller than 100%, all of the packets can be buffered in Q0 and none of them will be dropped. When the traffic load of total packets is higher than 100%, we can assume that the proportion of packets beyond 100% will be dropped. If the arrival rate of the Control packets is  $\lambda1$ , the arrival rate of the NOC packets is  $\lambda2$ , and the service time for a packet is  $T$ , the loss rate of the NOC packets, denoted by  $L_r$ , is

$$L_r = \begin{cases} 0 & \dots\dots\dots (\lambda1 + \lambda2)T < 1 \\ \frac{(\lambda1 + \lambda2) - 1/T}{\lambda2} & \dots\dots\dots (\lambda1 + \lambda2)T > 1 \end{cases} \quad (6-2)$$

# Chapter 7

## Improve the Performance

In Chapter 4 and 5, analytic models are given to calculate the delay and loss rate of the packets delivered in the DMP networks. In this Chapter, a discussion is given to explore the causes of those delays, and several resolutions are proposed to restrict the delay and loss rate of DMP system.

### 7.1 Overview of the Delays

For a packet delivered in DMP networks, the end-to-end time delay of this packet is the sum of propagation time in the path, the processing time in user equipment, and the waiting time in the waiting queues of network nodes.

The propagation time is the time for a packet spending on the fiber between two nodes. The total propagation time in the path is given by the product of the speed of light  $c$  and the distance between two users.

The processing time in user equipment is determined by the processing capacity of the processors and the processing strategy of user equipment.

The waiting time in the waiting queues of a network node is determined by the traffic interest of this node, the size of waiting queue, and the output link capacity of the network node.

In the following sections, we will go deep into those delays and propose some resolutions to restrict each kind of delays respectively.

## 7.2 Faster than Light

The propagation delay seems inevitable if we use electromagnetic wave to carry the signals, because the speed of electromagnetic wave is fixed about  $3 \times 10^8$  m/s.

It takes 10 ms for the signals of a packet to spread over 3000 kilometers, which means the delay of this packet on the fiber is about 10 ms. Distributed Multimedia Play in some cases requires delays as low as 10 ms for audio and 20 ms for video. Those contents are coded into sub-objects and encapsulated in the NOC packets. To satisfy the quality requirements of DMP, the delays of the NOC packets should be guaranteed less than 10 ms. Therefore, 3000 kilometers is the upper limit of the diameter of the DMP network. If the distance of two users in a DMP collaboration is larger than 3000 kilometers, the propagation delay in the path will be more than 10 ms, which fails to satisfy the quality requirement of DMP for audio service.

If we want to set up a DMP collaboration for users farther than 3000 kilometers away, the quality of service has to be degraded, or we have to find a way to deliver information faster than light, which seems impossible to achieve. Nevertheless, DMP system is supposed to be realized in 10 or 15 years. If a signal carrier spreading faster than the electromagnetic waves is discovered at that time and a corresponding information transmission technology is developed to deliver message faster than light, this technology will be implemented in the DMP system to reduce the delays.

It was once assumed that the speed of gravity might be faster than light, and the gravitational wave can be used as a substitute of electromagnetic wave in information technology. However, the theory of general relativity and some recent experiments<sup>1</sup> shows that speed of gravity might be equal to the speed of light. [23] [24]

We do not go deep into those hypothesizes of superluminal communication. With very little probability we can realize superluminal communication when the DMP system is deployed. However, nothing is impossible.

---

<sup>1</sup> In September 2002, Sergei Kopeikin and Edward Fomalont announced that they had made an indirect measurement of the speed of gravity, using their data from VLBI measurement of the retarded position of Jupiter on its orbit during Jupiter's transit across the line-of-sight of the bright radio source quasar QSO J0842+1835. Kopeikin and Fomalont concluded that the speed of gravity is between 0.8 and 1.2 times the speed of light, which would be fully consistent with the theoretical prediction of general relativity that the speed of gravity is exactly the same as the speed of light. [25]

### 7.3 Parallel Processors

The user equipment of the DMP system uses ASIC (Application Specific Integrate Circuit) as the central processor. The processor is customized for particular use of DMP, handling the security headers and decoding the data stream.

The processing capability of the processors increases observably every year because of the progress of hardware technology. According to Moore’s Law, the number of transistors that can be inexpensively placed on an integrated circuit is increasing exponentially, doubling approximately every two years, which means the processing capacity of ASIC inexpensively double approximately every two years. [26]

To support near-natural virtual quality, the DMP collaborations require high spatial and temporal resolution. Therefore, the traffic of a DMP collaboration is extremely high, up to  $10^3 - 10^4$  times higher than today’s videoconference system. [18] Decrypting the IPSec payloads and decoding the data streams require huge computing capacity of the processor. Even ten years later, the computing capability of one ASIC processor may not satisfy the requirement. A parallel processing scheme is proposed to accelerate the processing speed.

Because sub-objects of DMP object are independently encapsulated in the NOC packets, the NOC packets can be processed in parallel. As shown in Figure 7-1, when the NOC packets arrive at the system, passing through Q2, they will be forward to one of the several processors, getting decrypted and decoded, and then be sent to the Scene object buffer to display.

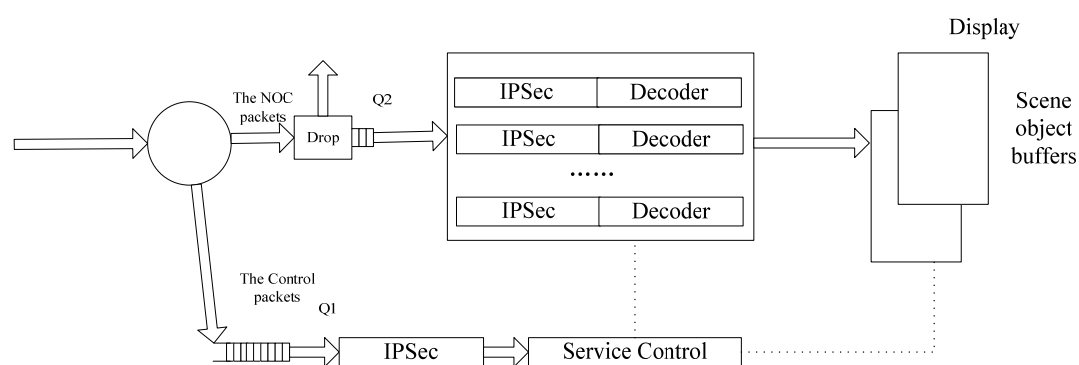


Figure 7-1: parallel processors in the user equipment of end node

Assuming that  $n$  parallel processors work together in the user equipment of a DMP end node, the processing capacity of this equipment can be  $n$  times larger than a user equipment with single processor.

## 7.4 Powerful Routers

The network nodes of the DMP system are responsible for routing and forwarding the DMP packets, the function of which are similar to the routers in Internet.

Nowadays, the most powerful routers support 10 Gbps Ethernet ports. [27] If we implement DMP systems now, we can expect that the maximum capacity of the ports of the DMP network node is also 10 Gbps. For a DMP packet with a fixed length of 1.5k bytes, the capacity of the port is  $8.33 \times 10^5$  packets per second. Assume that the size of buffer in a network node is 1000, which means that the packets might wait in a queue of 1000 packets when the traffic load is high. The waiting time in such a queue will be:

$$1000 \div 8.33 \times 10^5 / \text{sec} = 1.2 \text{ms}$$

If the collaboration contains 6 network nodes, as we discussed in Chapter 5, the maximum delay in the network nodes is  $1.2 \text{ms} \times 6 = 7.2 \text{ms}$ . Taking the delay in the fiber and the processing time in the user equipment of the end node into account, 7.2ms delay in the network nodes is too large.

However, Moore's Law also applies to the link capacity of the routers. The processing capacity inexpensively double approximately every two years. We can expect that ten years later the capacity of the routers will be  $2^5$  times stronger than nowadays routers. Then the waiting time of a DMP packet in the buffer of network node will be

$$1.2 \div 2^5 = 0.0375 \text{ms}$$

Within Europe, the maximum number of hops between two users is 11. The maximum waiting time in the queues of network nodes will be about 0.4ms ten years later. This delay is acceptable.



## Chapter 8

# Conclusions and the Future Work

The main goal of this Master's thesis is to analyze the traffic delay and loss rate of the packets passing through the DMP network nodes. The delay and loss rate determine the quality of service of the DMP system. In this chapter, a conclusion is given based on the analytic models and the experiments on the simulators. Additional guidelines for the future work are also presented.

### 8.1 Conclusion

In this Master's thesis, two simulators *NodeSim* and *RouteSim* are built to simulate the traffic passing through a DMP network node and the traffic passing through several DMP network nodes. Three analytic models based on Queuing Theory are built to calculate the average waiting time and the loss rate of the packets passing through a DMP network node.

The M/D/1 with infinite queue model is used to calculate the delay of the Control packets and the delay and loss rate of the NOC packets when the traffic load is lower than 100%. The M/D/1 with dynamic queue model and the M/B/1 with dynamic queue model is used to calculate the delay and loss rate of the NOC packets when the total offered load is higher than 100% and some NOC packets will be dropped selectively.

The calculations are conducted by using Matlab. The calculation results from the three mathematic models are consistent well with the delay and loss rate we get from the experiments on *NodeSim*. When the total traffic load is higher than 100%, the M/B/1 with dynamic queue model performs a little better than the M/D/1 with dynamic queue model.

We also have discussed the set-up of DMP multi-collaboration and the delay and loss rate of the traffic of a DMP collaboration. The experiments on *RouteSim* show that by implementing the dropping scheme on the network nodes, the traffic delay of a DMP

collaboration can be guaranteed, but the loss rate of the NOC packets may be out of the scope if the offered load of some network nodes is higher than 100%.

We proposed an updated version of the dropping scheme to support the requirement of guaranteed delay and reliable delivery of the packets with sub-obj seq. no 9.

Because of the restricted speed of light, the restricted processing capacity of processors, and the restricted link capacity of Routers, the DMP system can not be applied immediately. Parallel processors can solve the problem of processing capacity. Other problems are left to the technology development in the future.

The analytic models in this Master's thesis can also be applied to other queuing systems in addition to the dropping scheme of the DMP nodes. Any queuing systems like Figure 8-1 can use the analytic models in this report to calculate the average waiting time and the loss rate, if the interarrival distributions are Poisson Process.

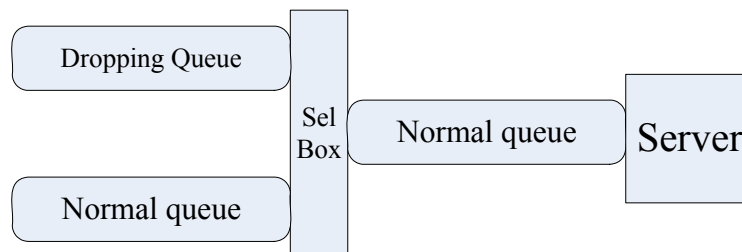


Figure 8-1: Queuing systems that the analytic models can be applied

## 8.2 Future work

The analytic models in this Master's thesis give the delay and loss rate of the packets when the interarrival distribution approaches Poisson Process. These models can be applied in the future to estimate the performance of DMP network when new quality requirements are add to DMP or new technologies are applied to DMP. New technologies may change the service time, the queue length or the dropping scheme of the nodes, but the analytic models are still applied by adjusting those parameters.

Future work on DMP will focus on the technical details like coding schemes of audio and video content and how to construct the stereoscopic videos in DMP. The analytic models of delay and loss rate of the DMP packets in the network nodes when the interarrival distribution is not Poisson process can also be researched in the future.

## Reference

1. L. A. Rønningen. *The DMP system and Physical Architecture*. September 2007. Available from <http://www.item.ntnu.no/fag/ttm4140/host06/index.html>, last visited 17 June 2008
2. L. A. Rønningen, *Analysis of a Traffic Shaping Scheme*, in *Proc. The 10th International Teletraffic Congress*, (Montreal, Canada), 1983.
3. J. Sklenar, *Introduction to OOP in SIMULA*. 1997. Available from <http://staff.um.edu.mt/jsk11/talk.html>, last visited 17 June 2008
4. Graham Birtwistle. *DEMOS - A system for Discrete Event Modelling on Simula*. Jananury 2003
5. L. A. Rønningen, *A protocol stack for futuristic multimedia*. Technical report, Dept of Telematics, NTNU, 2007
6. L. A. Rønningen, A. Lie. *Transient Behaviour of an Adaptive Traffic Control Scheme*, presented at EUNICE 2002, 02.09.2002
7. L. A. Rønningen, A. Lie. *Performance Control Of High-Capacity IP Networks For Collaborative Virtual Environments*, IBC 2002 Conference Proceedings, 12–15 September 2002
8. L. A. Rønningen, A. Lie. *Distributed Multimedia Plays with QoS guaranties over IP*, IEEE Wedelmusic'03, Leeds UK, 2003.
9. The DVB Project Office. *Digital Video Broadcasting (DVB); Multimedia Home Platform (MHP) Specification 1.2*. April, 2007. Available from <http://www.mhp.org/>.
10. E. Heiberg, *Stereoscopic video and requirements to spatial and temporal resolution*, Master's thesis, NTNU
11. Official JPEG 2000 page. Website <http://www.jpeg.org/jpeg2000/>, last visited 19

June 19, 2008

12. Shipeng Li, Weiping Li, *Shape Adaptive Discrete Wavelet Transforms for Arbitrarily Shaped Visual Object Coding*. IEEE Transaction on Circuits and System for Video Technology, August 2000
13. Greg Roelofs, A Basic Introduction to PNG Features. Available from <http://www.libpng.org/pub/png/pngintro.html>. Last visited 19 June 2008.
14. Kirk Martinez, *High Resolution Digital Imaging of Paintings: The VASARI Project*. History of Art Department Birkbeck College & University College of London, London WC1H 0PD, England. Available from <http://web.simmons.edu/~chen/nit/NIT'91/127-mar.htm>, last visited 19 June 2008
15. S. H. Johansen, T. M., and S. Krogdahl, Linux Software Map: Cim -- Simula Compiler based on the C programming. Available from <http://www.boutell.com/lsm/lsmbyid.cgi/000641>, last visited 19 June 2008.
16. RFC 2460, Internet Protocol, Version 6 (IPv6) Specification. Available from <http://www.ietf.org/rfc/rfc2460.txt>, last visited 19 June 2008.
17. RFC 2401, Security Architecture for the Internet Protocol. Available from: <http://rfc.net/rfc2401.html>, last visited 19 June 2008.
18. Olivier Brun and Jean-Marie, *Analytical solution of finite capacity M/D/1 queues* Garcia J. Appl. Probab. Volume 37, Number 4 (2000), 1092-1098.
19. Robert B. Cooper, *Introduction to Queuing Theory (third edition)*, ISBN 0-941893-03-0
20. RFC 2402, IP Authentication Header. Available from <http://www.ietf.org/rfc/rfc2402.txt>, last visited 20 June 2008.
21. RFC 2406, IP Encapsulating Security Payload (ESP). Available from <http://www.ietf.org/rfc/rfc2406.txt>, last visited 20 June 2008.
22. RFC 4306, Internet Key Exchange (IKEv2) Protocol. Available from <http://www.ietf.org/rfc/rfc4306.txt>, last visited 20 June 2008.
23. Poincaré, H. (1904/2000), *The present and the future of mathematical physics*, Bull. Amer. Math. Soc. 37: 25-38
24. Carlip, S. (2000). *Aberration and the Speed of Gravity*. Phys. Lett. A 267: 81–87.

25. S. M. Kopeikin, E. B. Fomalont, *Aberration and the Fundamental Speed of Gravity in the Jovian Deflection Experiment*, Found.Phys. 36 (2006) 1244-1285
26. Moore Gordon. *The Future of Integrated Electronics*. Fairchild Semiconductor internal publication (1964).
27. CISCO Company, Cisco 7600 Series Internet Routers. Available from <http://www.cisco.com/warp/public/cc/pd/rt/7600osr/index.shtml>, last visited 21 June 22, 2008.
28. L. Kleinrock. *Queueing Systems. Volume I: Theory*. John Wiley, New York. 1975.
29. Maria Kihl. *Queueing Systems* Lunds Universitet Printed in Sweden Lund 2004.
30. M. H. A. Davis; J. M. Howl. *A Markovian Analysis of the Finite-Buffer M/D/1 Queue*. Proceedings: Mathematical, Physical and Engineering Sciences, Vol. 453, No. 1964. (Sep. 8, 1997), pp. 1947-1962.
31. H. Tijms; K. Staats. *Negative Probabilities at Work in The M/D/1 Queue*. Probability in the Engineering and Informational Sciences, 21, 2007, 67–76.
32. S. Kasahara; H Takagi; Y. Takahashi; T. Hasegawa. *M/G/1/K System With Push-out Scheme Under Vacation Policy*. Journal of Applied Mathematics and Stochastic Analysis 9, Number 2, 1996, 143-157
33. The MathWorks, Inc. Matlab Overview. Available from <http://www.mathworks.com/products/matlab/index.html>, last visited 22 June 2008.



# Appendix A

## Using the simulator

In this chapter we will describe how to set up and use the simulators *NodeSim* and *RouteSim*.

The source code of the two simulators should be compiled by CIM compiler. To use the simulators, we must set up the simulation parameters by modifying *Input\_NodeSim.txt* and *Input\_RouteSim.txt*, then execute the simulator and get the simulation result.

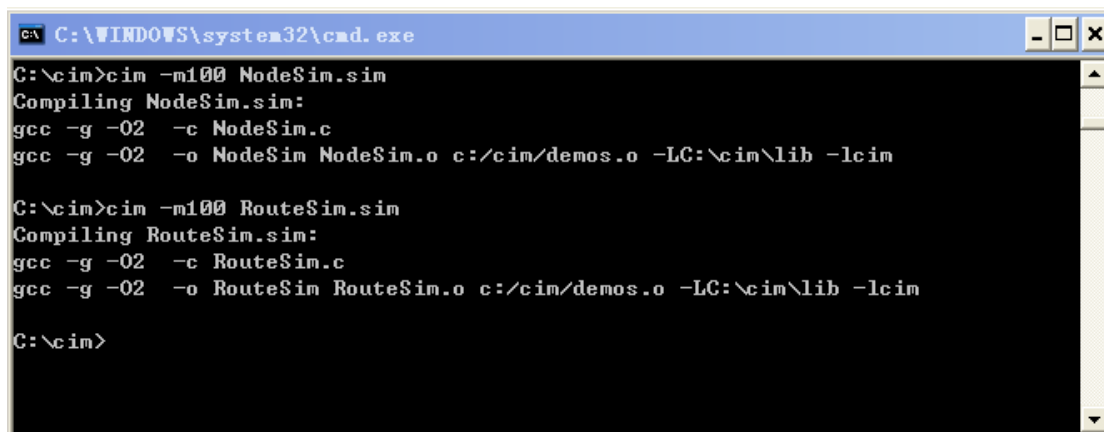
### A.1 Compiling

In the environment of windows XP, the two simulators *NodeSim* and *RouteSim* are compiled by the following commands in windows command line:

```
cim -m100 NodeSim.sim
```

```
cim -m100 RouteSim.sim
```

See Figure A-1.



```
cmd C:\WINDOWS\system32\cmd.exe
C:\cin>cim -m100 NodeSim.sim
Compiling NodeSim.sim:
gcc -g -O2 -c NodeSim.c
gcc -g -O2 -o NodeSim NodeSim.o c:/cin/demos.o -LC:\cin\lib -lcim

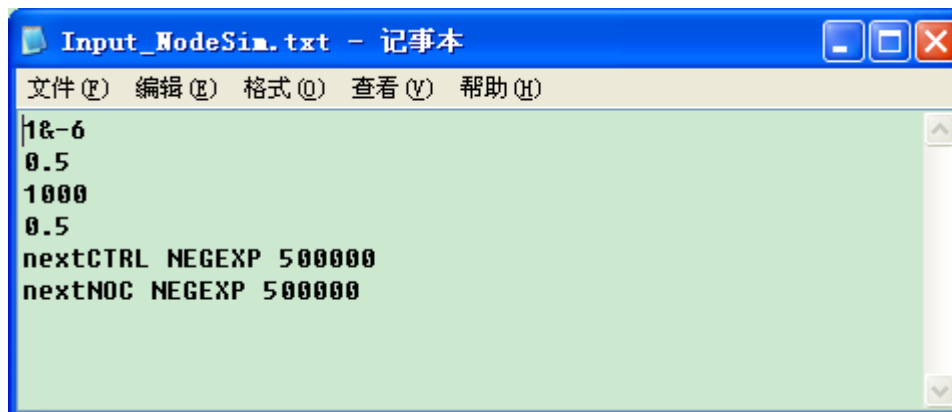
C:\cin>cim -m100 RouteSim.sim
Compiling RouteSim.sim:
gcc -g -O2 -c RouteSim.c
gcc -g -O2 -o RouteSim RouteSim.o c:/cin/demos.o -LC:\cin\lib -lcim

C:\cin>
```

Figure A-1: Compile the two simulators

## A.2 Define the parameters of *NodeSim*

To use *NodeSim*, we should define those parameters: the link capacity of the output port of the node (the fixed service time of  $H$ ), the value of  $p_1$ , the size of  $Q_0$ , the length of one simulation period, the interarrival distribution of the Control packets and its parameter(s), the interarrival distribution of the NOC packets and its parameter(s). They are defined in *Input\_NodeSim.txt* orderly line by line, see Figure A-2.

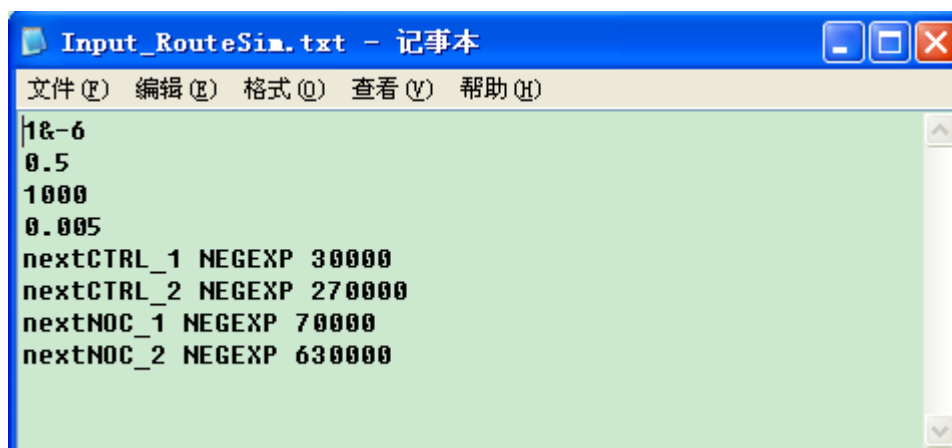


```
1e-6
0.5
1000
0.5
nextCTRL NEGEXP 500000
nextNOC NEGEXP 500000
```

Figure A-2: *Input\_NodeSim.txt*

## A.3 Define the parameters of *RouteSim*

To use *RouteSim*, we should define those parameters: the link capacity of the output port (the fixed service time of  $H$ ) of every node, the value of  $p_1$  of every node, the size of  $Q_0$  of every node, the length of one simulation period, the interarrival distribution of the Control packets from The Source Host and its parameter(s), the interarrival distribution of the Control packets from other nodes to any one of the five network nodes and its parameter(s), the interarrival distribution of the NOC packets from The Source Host and its parameter(s), the interarrival distribution of the NOC packets from other nodes to any one of the five network nodes and its parameter(s). They are defined in *Input\_NodeSim.txt* orderly line by line, See Figure A-4.



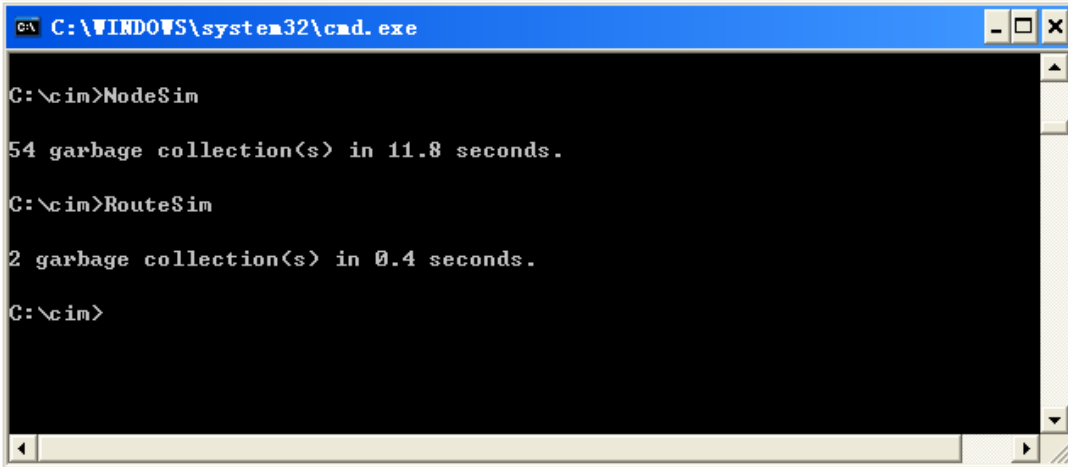
```
1e-6
0.5
1000
0.005
nextCTRL_1 NEGEXP 30000
nextCTRL_2 NEGEXP 270000
nextNOC_1 NEGEXP 70000
nextNOC_2 NEGEXP 630000
```

Figure A-3: *Input\_NodeSim.txt*



## A.4 Execute the simulators

The two simulators can be executed by double clicking the files *NodeSim.exe* and *RouteSim.exe* directly, or by executing *NodeSim* and *RouteSim* in Command line of Windows XP environment, see figure A-4.



```

C:\WINDOWS\system32\cmd.exe

C:\cim>NodeSim

54 garbage collection(s) in 11.8 seconds.

C:\cim>RouteSim

2 garbage collection(s) in 0.4 seconds.

C:\cim>
    
```

Figure A-4: Executing *NodeSim* and *RouteSim*

## A.5 Output of *NodeSim*

The output of *NodeSim* is saved in four text files.

*NodeSim\_delay\_NOC.txt* records the delay of every NOC packet when they pass through this node, see Figure A-5.



```

NodeSim_delay_NOC.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

OK
0.000001000 0.000001000 0.000001397 0.000001366 0.000001831 0.000001752 0.000002276
0.000003393
0.000003863 0.000004518 0.000005229 0.000004794 0.000005235 0.000006024 0.000007278
0.000007632
0.000002905 0.000003899 0.000004932 0.000005622 0.000007074 0.000004287 0.000005468
0.000004311
0.000003746 0.000002486 0.000002810 0.000001000 0.000001611 0.000001000 0.000001000
0.000001000
0.000001000 0.000004117 0.000005817 0.000003081 0.000003859 0.000003844 0.000004761
0.000005391
0.000003976 0.000004229 0.000004438 0.000005424 0.000005017 0.000006757 0.000005146
0.000007201
0.000007678 0.000005604 0.000004090 0.000003336 0.000003882 0.000002308 0.000002015
0.000002112
0.000001900 0.000001000 0.000001408 0.000003457 0.000002873 0.000003566 0.000002580
0.000002701
0.000002947 0.000003794 0.000007510 0.000006902 0.000005923 0.000007581 0.000008557
0.00000494
0.000009615 0.000008467 0.000009091 0.000009351 0.000009879 0.000010640 0.000010288
0.000009838
0.000008752 0.000006414 0.000007253 0.000008074 0.000007920 0.000008567 0.000009105
0.000007567
0.000007811 0.000008387 0.000009605 0.000008472 0.000010712 0.000011451 0.000008631
0.000009796
0.000010419 0.000011357 0.000011356 0.000011947 0.000012121 0.000012226 0.000011113
0.000011943
    
```

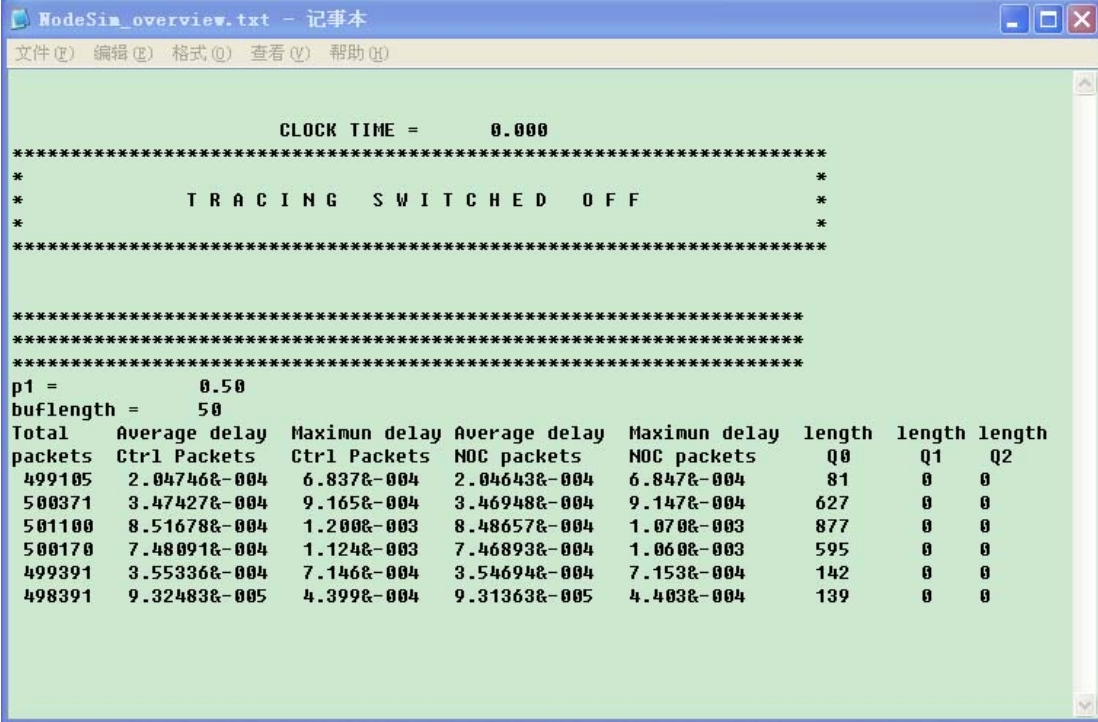
Figure A-5: *NodeSim\_delay\_NOC.txt*

*NodeSim\_delay\_Ctrl.txt* records the delay of every Control packet when they pass through this node, see Figure A-6.



Figure A-6: *NodeSim\_delay\_NOC.txt*

*NodeSim\_overview.txt* gives the average delay of the NOC packets, the maximum delay of the NOC packets, the average delay of the Control packets, the maximum delay of the Control packets, and the number of packets in the three waiting queues during the six running periods, see Figure A-7.



```

NodeSim_overview.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

                                CLOCK TIME =      0.000
*****
*                                TRACING SWITCHED OFF                                *
*                                                                                      *
*****

*****
*****
*****
p1 =                               0.50
buflength =                          50
Total Average delay  Maximun delay Average delay  Maximun delay  length  length length
packets Ctrl Packets  Ctrl Packets  NOC packets  NOC packets  Q0     Q1     Q2
499105  2.04746e-004    6.837e-004    2.04643e-004  6.847e-004    81     0     0
500371  3.47427e-004    9.165e-004    3.46948e-004  9.147e-004    627    0     0
501100  8.51678e-004    1.200e-003    8.48657e-004  1.070e-003    877    0     0
500170  7.48091e-004    1.124e-003    7.46893e-004  1.060e-003    595    0     0
499391  3.55336e-004    7.146e-004    3.54694e-004  7.153e-004    142    0     0
498391  9.32483e-005    4.399e-004    9.31363e-005  4.403e-004    139    0     0
    
```

Figure A-7: NodeSim\_overview.txt

*NodeSim\_drop.txt* shows the number of packets with different sequence numbers dropped by the node and the total loss rate of this node during a period of run, see Figure A-8.



```

NodeSim_drop.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

OK
Incoming  drop      drop      drop      drop      drop      drop      drop      drop      drop
Packets   SN_8     SN_7     SN_6     SN_5     SN_4     SN_3     SN_2     SN_1     Rate
249225    0         0         0         0         0         0         0         0         0.0000e+000
250449    0         0         0         0         0         0         0         0         0.0000e+000
250677    661      172      17         0         0         0         0         0         3.3908e-003
249917    374      73        5         0         0         0         0         0         1.8086e-003
249893    0         0         0         0         0         0         0         0         0.0000e+000
248894    0         0         0         0         0         0         0         0         0.0000e+000
    
```

Figure A-8: NodeSim\_drop.txt

## A.6 Output of *RouteSim*

The output of *RouteSim* is saved in eighteen text files.

*RouteSim\_delay\_Ctrl.txt* records the delay of every Control packet from the source to the destination, see Figure A-9.



Figure A-9: *RouteSim\_delay\_Ctrl.txt*

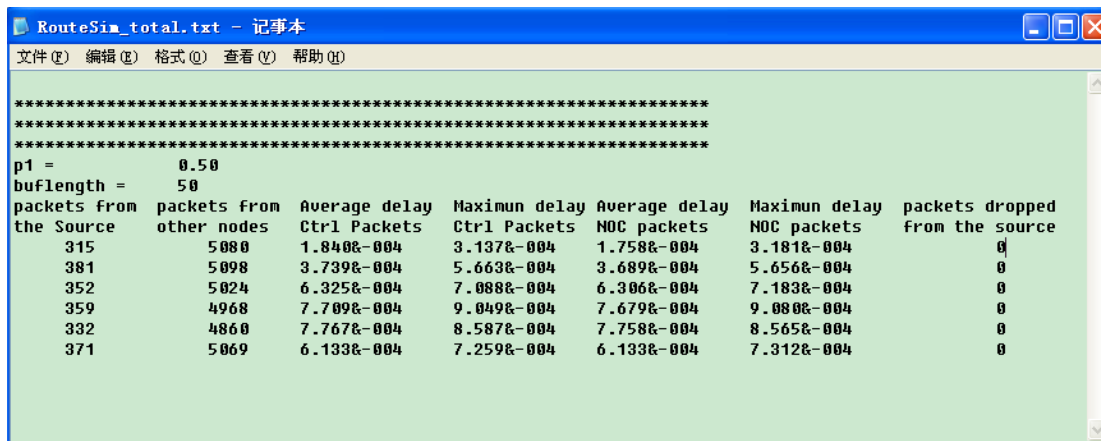
*RouteSim\_delay\_NOC.txt* records the delay of every NOC packet from the source to the destination, see Figure A-10.



Figure A-10: *RouteSim\_delay\_NOC.txt*



*RouteSim\_total.txt* gives the statistic characters of the packets coming from the Source Host to the Destination Host. It gives the average delay of the NOC packets, the maximum delay of the NOC packets, the average delay of the Control packets, the maximum delay of the Control packets, and the number of packets dropped by the network nodes during the six running periods, see Figure A-11.



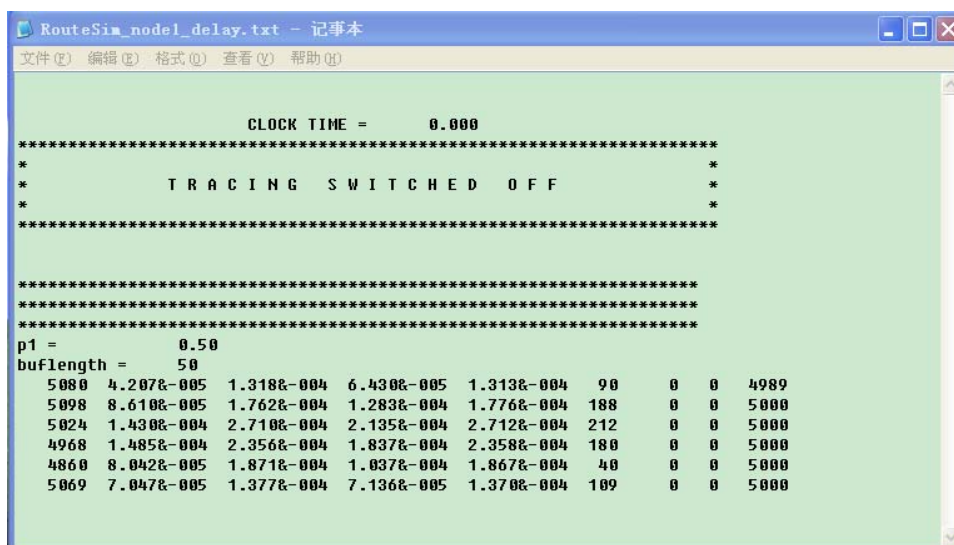
```

*****
*****
*****
p1 =          0.50
buflength =    50
packets from the Source      packets from other nodes      Average delay Ctrl Packets      Maximum delay Ctrl Packets      Average delay NOC packets      Maximum delay NOC packets      packets dropped from the source
315          5080      1.840e-004      3.137e-004      1.758e-004      3.181e-004      0
381          5098      3.739e-004      5.663e-004      3.689e-004      5.656e-004      0
352          5024      6.325e-004      7.888e-004      6.306e-004      7.183e-004      0
359          4968      7.709e-004      9.849e-004      7.679e-004      9.880e-004      0
332          4860      7.767e-004      8.587e-004      7.758e-004      8.565e-004      0
371          5069      6.133e-004      7.259e-004      6.133e-004      7.312e-004      0
    
```

Figure A-11: *RouteSim\_total.txt*

*RouteSim\_node1\_delay.txt*, *RouteSim\_node2\_delay.txt*, *RouteSim\_node3\_delay.txt*, *RouteSim\_node4\_delay.txt*, and *RouteSim\_node5\_delay.txt* give the statistic characters of all packets passing through one unique network node in the path. It give the average delay of the NOC packets, the maximum delay of the NOC packets, the average delay of the Control packets, the maximum delay of the Control packets, and the number of packets in the three waiting queues during the six running periods.

For instance, *RouteSim\_node1\_delay.txt* gives statistic properties of all packets passing through node1, see Figure A-12.



```

*****
*****
*****
CLOCK TIME =    0.000
*****
*
*          T R A C I N G   S W I T C H E D   O F F
*
*
*****
*****
*****
p1 =          0.50
buflength =    50
5080  4.207e-005  1.318e-004  6.430e-005  1.313e-004  90      0  0  4989
5098  8.610e-005  1.762e-004  1.283e-004  1.776e-004  188     0  0  5000
5024  1.430e-004  2.710e-004  2.135e-004  2.712e-004  212     0  0  5000
4968  1.485e-004  2.356e-004  1.837e-004  2.358e-004  180     0  0  5000
4860  8.042e-005  1.871e-004  1.837e-004  1.867e-004  40      0  0  5000
5069  7.047e-005  1.377e-004  7.136e-005  1.370e-004  109     0  0  5000
    
```

Figure A-12: *RouteSim\_node1\_delay.txt*

*RouteSim\_node1\_drop.txt*, *RouteSim\_node2\_drop.txt*, *RouteSim\_node3\_drop.txt*, *RouteSim\_node4\_drop.txt*, and *RouteSim\_node5\_drop.txt* give the total number of packets with different sequence numbers dropped by each node, see Figure A-13.



Figure A-13: *RouteSim\_node1\_drop.txt*

*RouteSim\_ori\_node1\_drop.txt*, *RouteSim\_ori\_node2\_drop.txt*, *RouteSim\_ori\_node3\_drop.txt*, *RouteSim\_ori\_node4\_drop.txt*, and *RouteSim\_ori\_node5\_drop.txt* give the number of packets from the Source Host with different sequence numbers dropped by each node, see Figure A-14.

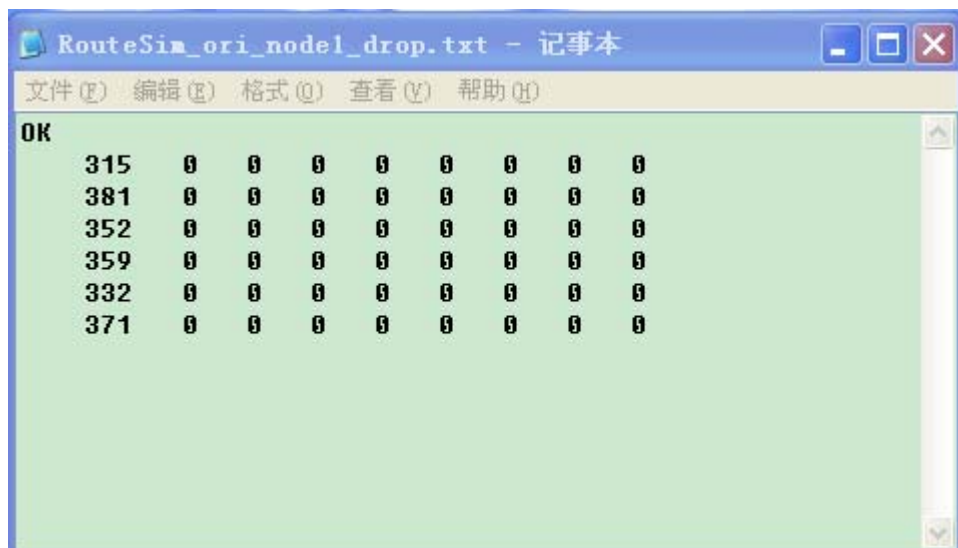


Figure A-14: *RouteSim\_ori\_node1\_drop.txt*

## Appendix B

# Discussion of a Special Case of Non-Homogeneous Poisson Process

In this section, we discuss the derivation of Equation 4-20, which plays key role in exploring the mathematical nature of a dynamic queue in the M/D/1 with dynamic queue model and M/B/1 with dynamic queue model.

### B.1 mathematical proof

Let us review the problem in Section 4.3.1. Some NOC packets arriving when Q0 is full will be queued in Q2 first, and some of them will be dropped according to their sub-object sequence number and the number of packets waiting in Q2. This can be considered as the arrival rate of NOC packets varies when the number of packets queuing in Q2 changes. See Figure 4-1 and its description.

Our purpose is to calculate  $\alpha(i, j)$  which denote the probability of  $i$  packets arriving during a customer service period while  $j$  customers have already been in the queue. The packets arriving process can be characterized by Poisson Process, but the arrival rate varies according to the number of packets in Q2. Even in these  $i$  incoming packets, the arrival rate of latter packets might be different from the previous ones, because the number of packets in Q2 has changed. This is a special case of non-homogeneous Poisson Process. In general, when we talk about non-homogeneous Poisson Process, we mean the arrival rate parameter changes over time. But in our case, the arrival rate parameter changes over the number of events occurred ahead of it, that is, the first packet arrives with the rate  $\rho_1$ , the second packet arrives with the rate  $\rho_2$ , and the  $i$ th packet arrives with the rate  $\rho_i$ . The arrival rate of an event is determined only by the number of events that occurs before it.

The Poisson Distribution is derived from Binomial Distribution with parameters  $n$  and  $\lambda/n$  as  $n$  approaches infinity, i.e., the probability distribution of the number of successes in  $n$  trials, with the probability of success on each trial  $\lambda/n$ . In our case, the non-homogenous Poisson Process can be modeled as the probability of success on

each trial  $\lambda/n$  changes when a successful trial happens. In this model, parameter  $\lambda$  in the Binomial Distribution is parameter  $\rho$  in the corresponding Poisson Distribution.

To calculate the probability of  $k$  successful trials among those  $n$  trials, we sort the  $n$  trials into  $k+1$  sorts by the number of successful trials happening before it. The numbers of trials in each sort is denoted by  $n_i$ , in which  $i$  is the number of successful trials before this sort of trials., the corresponding success rate of  $\{n_0, n_1, n_2, \dots, n_k\}$  is  $\{\lambda_0/n, \lambda_1/n, \lambda_2/n, \dots, \lambda_k/n\}$ , i.e., the first successful trial happens after  $n_0-1$  unsuccessful trials, and the success probability of each of the first  $n_0$  trial is  $\lambda_0/n$ , then  $n_1-1$  unsuccessful trials follow after the first successful trial and before the second successful trial, and the success probability of each of those  $n_1$  trial is  $\lambda_1/n$ , and so on. After the  $k$ th which is also the last successful trial, there are  $n_k$  unsuccessful trials left, whose success probability in theory is  $\lambda_k/n$ .

It is known that

$$n_0 + n_1 + n_2 + \dots + n_k = n \quad (\text{B-1}),$$

There are  $C_{n-1}^k$  groups of roots that satisfy Equation B-1. And given a specific root  $\{n_0, n_1, n_2, \dots, n_k\}$ , the probability of realizing this solution is

$$P(\{n_0, n_1, n_2, \dots, n_k\}) = \prod_{i=0}^{k-1} \frac{\lambda_i}{n} \cdot \prod_{i=0}^k (1 - \frac{\lambda_i}{n})^{n_i-1} \quad (\text{B-2})$$

Because  $n$  approaches infinity,

$$\lim_{n \rightarrow \infty} P(\{n_0, n_1, n_2, \dots, n_k\}) = \prod_{i=0}^{k-1} \frac{\lambda_i}{n} \cdot \prod_{i=0}^k e^{-\lambda_i \frac{n_i-1}{n}} = e^{-\sum_{i=0}^k \lambda_i \frac{n_i-1}{n}} \cdot \prod_{i=0}^{k-1} \frac{\lambda_i}{n} \quad (\text{B-3})$$

The probability of  $k$  successful trials among  $n$  trials is the summation of the probability of every possible root  $\{n_0, n_1, n_2, \dots, n_k\}$  of Equation B-1,

$$\begin{aligned} P(X = k) &= \lim_{n \rightarrow \infty} \sum_{n_0+n_1+\dots+n_k=n} (e^{-\sum_{i=0}^k \lambda_i \frac{n_i-1}{n}} \cdot \prod_{i=0}^{k-1} \frac{\lambda_i}{n}) \\ &= \lim_{n \rightarrow \infty} \prod_{i=0}^{k-1} \frac{\lambda_i}{n} \cdot \sum_{n_0+n_1+\dots+n_k=n} e^{-\sum_{i=0}^k \lambda_i \frac{n_i-1}{n}} \end{aligned} \quad (\text{B-4})$$

$n_i$  can be considered as a random variable with the sample space  $\{0 < n_i < n-k\}$ , **Lemma B-1** gives the expected value of  $n_i$ . And then we can get the expected value

$$\text{of } \sum_{i=1}^{k+1} \lambda_i \cdot \frac{n_i-1}{n}$$



**Lemma B-1:**  $\{n_1, n_2, \dots, n_k\}$  is a group of random variables which are all integers with sample space  $\{1 \leq n_i \leq n - k + 1\}$ ,  $n$  is a constant and  $\{n_1, n_2, \dots, n_k\}$  satisfy the equation  $n_1 + n_2 + n_3 + \dots + n_k = n$ . The probability of the appearance of any specific root  $\{n_1, n_2, \dots, n_k\}$  is equal to each other. If  $n$  approaches infinite, the expected value of any random valuable  $n_i \in \{n_1, n_2, \dots, n_k\}$ ,  $E(n_i) = n / k$ .

Proof: Because the terms  $n_1, n_2, \dots, n_k$  in the equation  $n_1 + n_2 + n_3 + \dots + n_k = n$  are symmetrical, any random valuable  $n_i \in \{n_1, n_2, \dots, n_k\}$  follows the same distribution.

$$\text{So } E(n_1) = E(n_2) = E(n_3) = \dots = E(n_k) \quad (\text{B-5}),$$

$$\text{And } E(n_1) + E(n_2) + E(n_3) + \dots + E(n_k) = E(n_1 + n_2 + n_3 + \dots + n_k) = E(n) = n \quad (\text{B-6}),$$

$$\text{So } E(n_1) = E(n_2) = E(n_3) = \dots = E(n_k) = \frac{n}{k} \quad (\text{B-7})$$

Let us back to Equation B-4. Define the random function  $X(n_0, n_1, \dots, n_k)$ :

$$X(n_0, n_1, \dots, n_k) = \lim_{n \rightarrow \infty} \sum_{i=0}^k \lambda_i \cdot \frac{n_i - 1}{n} \quad (\text{B-8})$$

where  $(n_0, n_1, \dots, n_k)$  satisfy Equation B-1.

From **Lemma B-1** and Equation B-1, we can get that to any  $n_i \in \{n_0, n_1, \dots, n_k\}$ ,

$$E(n_i) = \frac{n}{k+1} \quad (\text{B-9}),$$

So

$$E(X) = \lim_{n \rightarrow \infty} \sum_{i=0}^k \lambda_i \cdot \frac{E(n_i) - 1}{n} = \lim_{n \rightarrow \infty} \sum_{i=0}^k \lambda_i \cdot \frac{\frac{n}{k+1} - 1}{n} = \frac{\sum_{i=0}^k \lambda_i}{k+1} \quad (\text{B-10})$$

$$\text{Define } Y(X) = e^X = \lim_{n \rightarrow \infty} e^{-\sum_{i=0}^k \lambda_i \cdot \frac{n_i - 1}{n}} \quad (\text{B-11})$$

Because  $f(x) = e^x$  is a Convex function,

$$E(Y) \geq Y[E(X)]$$

Only when  $X$  is a constant function,  $E(Y) = Y[E(X)]$ .

If the Variance of  $\lambda_i$  is 0, that is,  $\lambda_0 = \lambda_1 = \dots = \lambda_k$

$$X(n_0, n_1, \dots, n_k) = \lim_{n \rightarrow \infty} \sum_{i=0}^k \lambda_i \cdot \frac{n_i - 1}{n} = \lim_{n \rightarrow \infty} \lambda_i \cdot \sum_{i=0}^k \frac{n_i - 1}{n} = \lambda_i,$$

So  $X$  is a constant function, and  $E(Y) = Y[E(X)]$  when  $\lambda_0 = \lambda_1 = \dots = \lambda_k$ .

When the variance of  $\lambda_i$  increase, the gap between  $E(Y)$  and  $Y[E(X)]$  increase. Fortunately, in our cases, see Section 4.1.2, the variance of continuous several  $\lambda_i$ 's is never large. So we make an assumption that

$$E(Y) \approx Y[E(X)] \quad (\text{B-12})$$

Then we can estimate

$$\sum_{n_1+n_2+\dots+n_{k+1}=n} Y(X) = C_{n-1}^k \cdot E(Y) \approx C_{n-1}^k \cdot Y(E(X)) = C_{n-1}^k \cdot e^{\frac{\sum_{i=0}^k \lambda_i}{k+1}} \quad (\text{B-13})$$

Then Equation B-4 becomes:

$$\begin{aligned} P(X = k) &= \lim_{n \rightarrow \infty} \prod_{i=0}^{k-1} \frac{\lambda_i}{n} \cdot \sum_{n_0+n_1+\dots+n_k=n} e^{-\sum_{i=0}^k \lambda_i \cdot \frac{n_i-1}{n}} \\ &= \lim_{n \rightarrow \infty} \left( \prod_{i=0}^{k-1} \frac{\lambda_i}{n} \right) \cdot C_{n-1}^k \cdot e^{\frac{\sum_{i=0}^k \lambda_i}{k+1}} \\ &= \lim_{n \rightarrow \infty} \frac{(n-1)!}{(n-1-k)! \cdot k! \cdot n^k} \cdot e^{\frac{\sum_{i=0}^k \lambda_i}{k+1}} \cdot \prod_{i=0}^{k-1} \lambda_i \\ &= \frac{\prod_{i=0}^{k-1} \lambda_i}{k!} \cdot e^{\frac{\sum_{i=0}^k \lambda_i}{k+1}} \end{aligned} \quad (\text{B-14})$$

So the corresponding Poisson Distribution:

$$P(X = k) = \frac{\prod_{i=0}^{k-1} \rho_i}{k!} \cdot e^{\frac{\sum_{i=0}^k \rho_i}{k+1}} \quad (\text{B-15})$$

We define the Geometric mean of  $\{\rho_0, \rho_1, \dots, \rho_{k-1}, \rho_k\}$  is  $\rho_g$ , and the Arithmetic mean of  $\{\rho_0, \rho_1, \dots, \rho_{k-1}\}$  is  $\rho_a$ , then we get

$$P(X = k) = \frac{\rho_a^k}{k!} \cdot e^{\rho_g} \quad (\text{B-16})$$

Equations 4-20 and 4-31 derive from Equation B-16.

Because Equation B-12 is still an assumption, the proof of B-16 is still not persuasive. In the next section, a simulation program is developed to verify the equation which

simulates the non-homogeneous Poisson Process caused by the dropping queue Q2 described in Section 4.1.1

## B.2 Verify by simulation program

In order to verify Equation B-16, a simulation program is built in Matlab to simulate this kind of non-homogeneous Poisson Process. (See Appendix C.6 Poissontest.m.) In the simulation program, events occur one by one and the interval of the events follows negative exponential distribution. The mean value of the interval distribution of different events varies according to the order of their occurrence.

The interval between the beginning of the simulation and the occurrence of the first event follows negative exponential distribution with mean value  $1/\lambda_1$ ; The interval between the occurrences of the first and second events follows negative exponential distribution with mean value  $1/\lambda_2$ ; The interval between the occurrences of the (i-1)th and ith events follows negative exponential distribution with mean value  $1/\lambda_i$ . We record the number of events occurred in a period T. The processes are repeated 500,000 times in one simulation, and we can calculate the probabilities of different numbers of events occur in a period.

We compare the probability of different number of packets arrive in a period from the simulator with the result computed by Equation B-12. Different choices of group parameters  $\lambda_1, \lambda_2, \lambda_3, \dots$  lead to different results. However, we only need to evaluate Equation B-12 in the condition of the dropping scheme in Section 4.1.2. That is:

If Q2.length 0-12 then  $\lambda_2' = \lambda_2$  else  
 If Q2.length 13-18, then  $\lambda_2' = \frac{7}{8} \lambda_2$  else  
 If Q2.length 19-24, then  $\lambda_2' = \frac{6}{8} \lambda_2$  else  
 If Q2.length 25-29, then  $\lambda_2' = \frac{5}{8} \lambda_2$  else  
 If Q2.length 30-34, then  $\lambda_2' = \frac{4}{8} \lambda_2$  else  
 If Q2.length 35-39, then  $\lambda_2' = \frac{3}{8} \lambda_2$  else  
 If Q2.length 40-45, then  $\lambda_2' = \frac{2}{8} \lambda_2$  else  
 If Q2.length 46-49, then  $\lambda_2' = \frac{1}{8} \lambda_2$  else  
 If Q2.length  $\geq 50$ , then  $\lambda_2' = 0$ .

So when the number of packets in Q2 increases from zero to fifty, the corresponding arrival rates of the NOC packets constitute an array:  $\lambda_2, \lambda_2, \lambda_2, \lambda_2, \lambda_2, \lambda_2, \lambda_2, \lambda_2$ ,

$$\lambda_2, \lambda_2, \lambda_2, \lambda_2, \frac{7}{8}\lambda_2, \frac{7}{8}\lambda_2, \frac{7}{8}\lambda_2, \frac{7}{8}\lambda_2, \frac{7}{8}\lambda_2, \frac{6}{8}\lambda_2, \frac{6}{8}\lambda_2, \frac{6}{8}\lambda_2, \frac{6}{8}\lambda_2, \frac{6}{8}\lambda_2, \frac{6}{8}\lambda_2, \frac{5}{8}\lambda_2, \frac{5}{8}\lambda_2, \frac{5}{8}\lambda_2, \frac{5}{8}\lambda_2, \frac{5}{8}\lambda_2, \frac{4}{8}\lambda_2, \frac{4}{8}\lambda_2, \frac{4}{8}\lambda_2, \frac{4}{8}\lambda_2, \frac{4}{8}\lambda_2, \frac{3}{8}\lambda_2, \frac{3}{8}\lambda_2, \frac{3}{8}\lambda_2, \frac{3}{8}\lambda_2, \frac{3}{8}\lambda_2, \frac{2}{8}\lambda_2, \frac{2}{8}\lambda_2, \frac{2}{8}\lambda_2, \frac{2}{8}\lambda_2, \frac{2}{8}\lambda_2, \frac{2}{8}\lambda_2, \frac{1}{8}\lambda_2, \frac{1}{8}\lambda_2, \frac{1}{8}\lambda_2, \frac{1}{8}\lambda_2, 0.$$

In the condition of the dropping scheme in Section 4.1.2, the group parameters  $\lambda_1, \lambda_2, \lambda_3, \dots$  of the interval distributions of the simulation program can be any subset of the above array.

To prove that Equation B-12 can approximately compute the probabilities of different numbers of events occur in a period, whose arrival rate varies according the dropping scheme in Section 4.1.2, we give different groups of values to the group parameters  $\lambda_1, \lambda_2, \lambda_3, \dots$ , and compare the result of the simulation with the probabilities calculated from Equation B-12.

We set the simulation period 1,  $\lambda_2=1$ , the number of packets in Q2 is 9, then  $\lambda_1, \lambda_2, \lambda_3, \dots = 1, 1, 1, 0.875, 0.875, 0.875, 0.875, 0.875, 0.75, 0.75, 0.75, 0.75, 0.75, 0.625, 0.625, 0.625, 0.625, \dots$ . The comparison result is in Table B-1.

Table B-1: Comparison of results from Equation B-12 and the simulation 1

The number of events occur during the period	0	1	2	3	4	5	6	7
Probabilities calculated from B-12	0.3679	0.3679	0.1839	0.0633	0.0141	0.0022	0.0003	0
Probabilities from the simulation	0.3679	0.3673	0.1839	0.0639	0.0141	0.0023	0.0003	0

We set the simulation period 1,  $\lambda_2=2$ , the number of packets in Q2 is 34, then  $\lambda_1, \lambda_2, \lambda_3, \dots = 1, 0.75, 0.75, 0.75, 0.75, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.25, 0.25, 0.25, 0.25$ . The comparison result is in Table B-2.

Table B-2: Comparison of results from Equation B-12 and the simulation 2

The number of events occur during the period	0	1	2	3	4	5	6	7
Probabilities calculated from B-12	0.3679	0.4169	0.1630	0.0416	0.0079	0.0008	0	0
Probabilities from the simulation	0.3664	0.4183	0.1639	0.0419	0.0080	0.0008	0.0001	0

We set the simulation period 1,  $\lambda_2=2$ , the number of packets in Q2 is 20, then  $\lambda_1, \lambda_2, \lambda_3, \dots = 1.5, 1.5, 1.5, 1.25, 1.25, 1.25, 1.25, 1.25, 1, 1, 1, 1, 1, 0.75, 0.75, 0.75, 0.75, \dots$ . The comparison result is in Table B-3.

Table B-3: Comparison of results from Equation B-12 and the simulation 3

The number of events occur during the period	0	1	2	3	4	5	6	7	8
Probabilities calculated from B-12	0.2231	0.3347	0.2510	0.1336	0.0433	0.0111	0.0023	0.0003	0
Probabilities from the simulation	0.2230	0.3351	0.2510	0.1337	0.0433	0.0111	0.0023	0.0003	0.0001

In the three experiments, we can see that the results from Equation B-12 are consistent well with the result from the simulation program of the non-homogeneous Poisson Process in the condition of the dropping scheme of Q2 in Figure 4-1.

The results of the experiments support that we can use Equation 4-20 to estimate  $\alpha(i, j)$  when the NOC packets queue in Q2, although this is not a perfect mathematic proof.



# Appendix C

## The Matlab scripts

This Chapter gives the scripts running on Matlab 7.0 to do the calculation based on the analytic models given in Chapter 4.

### C.1 *newm7\_plot.m*

*newm7\_plot.m* is used to calculate the delay and loss rate of the NOC packets from the M/B/1 with dynamic queue model, given the traffic load of the NOC packets, value of  $p_1$  and the size of  $Q_0$ .

```
function Y=newm7_plot(x)
roori=x;
ro2=1;
p0=0.3;
a=zeros(200);
a2=0;
f=0;

p=0.3;
for k=0:12
    ro=roori;

    a2=0;
    for i=1:1000
        a2=a2+exp(-ro*i)*(ro*i)^k/prod(1:k)*(1-p)*p^(i-1);
    end
    a(k+1)=a2;

    a2=0;
    ro=7/8*roori;
    for i=1:1000
        a2=a2+exp(-ro*i)*(ro*i)^k/prod(1:k)*(1-p)*p^(i-1);
    end
    a(20+k+1)=a2;

    a2=0;
    ro=6/8*roori;
    for i=1:1000
```

```
a2=a2+exp(-ro*i)*(ro*i)^k/prod(1:k)*(1-p)*p^(i-1);
end
a(40+k+1)=a2;

a2=0;
ro=5/8*roori;
for i=1:1000
    a2=a2+exp(-ro*i)*(ro*i)^k/prod(1:k)*(1-p)*p^(i-1);
end
a(60+k+1)=a2;

a2=0;
ro=4/8*roori;
for i=1:1000
    a2=a2+exp(-ro*i)*(ro*i)^k/prod(1:k)*(1-p)*p^(i-1);
end
a(80+k+1)=a2;

a2=0;
ro=3/8*roori;
for i=1:1000
    a2=a2+exp(-ro*i)*(ro*i)^k/prod(1:k)*(1-p)*p^(i-1);
end
a(100+k+1)=a2;

a2=0;
ro=2/8*roori;
for i=1:1000
    a2=a2+exp(-ro*i)*(ro*i)^k/prod(1:k)*(1-p)*p^(i-1);
end
a(120+k+1)=a2;

a2=0;
ro=1/8*roori;
for i=1:1000
    a2=a2+exp(-ro*i)*(ro*i)^k/prod(1:k)*(1-p)*p^(i-1);
end
a(140+k+1)=a2;

a2=0;
ro=1/9*roori;
for i=1:1000
    a2=a2+exp(-ro*i)*(ro*i)^k/prod(1:k)*(1-p)*p^(i-1);
end
a(160+k+1)=a2;

end

alfa=zeros(13,750);
ro=roori;
p=0.3;
for n=0:12
    for m=690:750
        if and((m+n)<=712,m~712)
            x=0;
            for i=1:1000
                x=x+((ro*i).^n)/prod(1:n).*exp(-ro*i)*p.^(i-1)*(1-p);
            end
        end
    end
end
```



```

    alfa (n+1,m+1)=x;
elseif (712<=m+n & m+n<=718)
    if (n==0)
        ro2=ro*7/8;
        ro3=(ro*7/8)^n;
        if (m==718)
            ro2=ro*6/8;
            ro3=(ro*6/8)^n;
        end
    else
        ro3=ro^(select(712-m)*(ro*7/8)^(n-select(712-m)));
        ro2=(select(712-m)*ro+(n-select(712-m))*ro*7/8)/n;
    end

    x=0;
    for i=1:1000
        x=x+(ro3*i^n)/prod(1:n).*exp(-ro2*i)*p.^(i-1)*(1-p);
    end
    alfa (n+1,m+1)=x;
elseif (718<(m+n) & m+n<=724)
    if (n==0)
        ro2=ro*6/8;
        ro3=(ro*6/8)^n;
        if (m==724)
            ro2=ro*5/8;
            ro3=(ro*5/8)^n;
        end
    else
        ro3=ro^select(712-m)*(ro*7/8)^select(6-select(m-712))*(ro*6/8)^(n-select(718-m));
        ro2=(select(712-m)*ro+select(6-select(m-712))*ro*7/8+(n-select(718-m))*ro*6/8)/n;
    end

    x=0;
    for i=1:1000
        x=x+(ro3*i.^n)/prod(1:n).*exp(-ro2*i)*p.^(i-1)*(1-p);
    end
    alfa (n+1,m+1)=x;

elseif (724<(m+n)& m+n<=730)
    if (n==0)
        ro2=ro*5/8;
        ro3=(ro*5/8)^n;
        if (m==730)
            ro2=ro*4/8;
            ro3=(ro*4/8)^n;
        end
    else
        ro3=ro^select(712-m)*(ro*7/8)^select(6-select(m-712))*(ro*6/8)^select(6-select(m-718))*(ro*5/8)^(n-select(724-m));
        ro2=(select(712-m)*ro+select(6-select(m-712))*ro*7/8+select(6-select(m-718))*ro*6/8+(n-select(724-m))*ro*5/8)/n;
    end

    x=0;
    for i=1:1000

```

```

        x=x+(ro3*i.^n)/prod(1:n).*exp(-ro2*i)*p.^(i-1)*(1-p);
    end
    alfa (n+1,m+1)=x;
elseif (730<(m+n)& m+n<=735)
    if (n==0)
        ro2=ro*4/8;
        ro3=(ro*4/8)^n;
        if (m==735)
            ro2=ro*3/8;
            ro3=(ro*3/8)^n;
        end
    end
else

ro3=ro^select(712-m)*(ro*7/8)^select(6-select(m-712))*(ro*6/8)^select(6-select(m-718))*(ro*5/8)^select(6-select(m-724))*(ro*4/8)^(n-select(730-m));

ro2=(select(712-m)*ro+select(6-select(m-712))*ro*7/8+select(6-select(m-718))*ro*6/8+select(6-select(m-724))*ro*5/8+(n-select(730-m))*ro*4/8)/n;

    end
    x=0;
    for i=1:1000
        x=x+(ro3*i.^n)/prod(1:n).*exp(-ro2*i)*p.^(i-1)*(1-p);
    end
    alfa (n+1,m+1)=x;
elseif (735<(m+n)& m+n<=740)
    if (n==0)
        ro2=ro*3/8;
        ro3=(ro*3/8)^n;
        if (m==740)
            ro2=ro*2/8;
            ro3=(ro*2/8)^n;
        end
    end
else

ro3=ro^select(712-m)*(ro*7/8)^select(6-select(m-712))*(ro*6/8)^select(6-select(m-718))*(ro*5/8)^select(6-select(m-724))*(ro*4/8)^select(5-select(m-730))*(ro*3/8)^(n-select(735-m));

ro2=(select(712-m)*ro+select(6-select(m-712))*ro*7/8+select(6-select(m-718))*ro*6/8+select(6-select(m-724))*ro*5/8+select(5-select(m-730))*ro*4/8+(n-select(735-m))*ro*3/8)/n;

    end
    x=0;
    for i=1:1000
        x=x+(ro3*i.^n)/prod(1:n).*exp(-ro2*i)*p.^(i-1)*(1-p);
    end
    alfa (n+1,m+1)=x;
elseif (740<(m+n)& m+n<=746)
    if (n==0)
        ro2=ro*2/8;
        ro3=(ro*2/8)^n;
        if (m==746)
            ro2=ro*1/8;
            ro3=(ro*1/8)^n;
        end
    end
else

ro3=ro^select(712-m)*(ro*7/8)^select(6-select(m-712))*(ro*6/8)^select(6-select(m-718))*(ro*5/8)^select(6-select(m-724))*(ro*4/8)^select(5-select(m-730))*(ro*3/8)^select(5-select(m-735))*(ro*

```

```

2/8)^(n-select(740-m));

ro2=(select(712-m)*ro+select(6-select(m-712))*ro*7/8+select(6-select(m-718))*ro*6/8+select(6-s
elect(m-724))*ro*5/8+select(5-select(m-730))*ro*4/8+select(5-select(m-735))*ro*3/8+(n-select(7
40-m))*ro*2/8)/n;
    end
    x=0;
    for i=1:1000
        x=x+(ro3*i^n)/prod(1:n).*exp(-ro2*i)*p.^(i-1)*(1-p);
    end
    alfa (n+1,m+1)=x;
elseif (746<(m+n)& m+n<=750)
    if (n==0)
        ro2=ro*1/8;
        ro3=(ro*1/8)^n;
    else

ro3=ro^select(712-m)*(ro*7/8)^select(6-select(m-712))*(ro*6/8)^select(6-select(m-718))*(ro*5/8
)^select(6-select(m-724))*(ro*4/8)^select(5-select(m-730))*(ro*3/8)^select(5-select(m-735))*(ro*
2/8)^select(5-select(m-740))*(ro*1/8)^(n-select(746-m));

ro2=(select(712-m)*ro+select(6-select(m-712))*ro*7/8+select(6-select(m-718))*ro*6/8+select(6-s
elect(m-724))*ro*5/8+select(5-select(m-730))*ro*4/8+select(5-select(m-735))*ro*3/8+select(5-se
lect(m-740))*ro*2/8+(n-select(746-m))*ro*1/8)/n;
    end
    x=0;
    for i=1:1000
        x=x+(ro3*i^n)/prod(1:n).*exp(-ro2*i)*p.^(i-1)*(1-p);
    end
    alfa (n+1,m+1)=x;
end
end
end
end
end

```

```

b=zeros(750);
b(1)=(1-a(1))/a(1);
b(2)=(b(1)-a(2)-b(1)*a(2))/a(1);
b(3)=(b(2)-a(3)-b(1)*a(3)-b(2)*a(2))/a(1);
b(4)=(b(3)-a(4)-b(1)*a(4)-b(2)*a(3)-b(3)*a(2))/a(1);
b(5)=(b(4)-a(5)-b(1)*a(5)-b(2)*a(4)-b(3)*a(3)-b(4)*a(2))/a(1);
b(6)=(b(5)-a(6)-b(1)*a(6)-b(2)*a(5)-b(3)*a(4)-b(4)*a(3)-b(5)*a(2))/a(1);
b(7)=(b(6)-a(7)-b(1)*a(7)-b(2)*a(6)-b(3)*a(5)-b(4)*a(4)-b(5)*a(3)-b(6)*a(2))/a(1);
b(8)=(b(7)-a(8)-b(1)*a(8)-b(2)*a(7)-b(3)*a(6)-b(4)*a(5)-b(5)*a(4)-b(6)*a(3)-b(7)*a(2))/a(1);
b(9)=(b(8)-a(9)-b(1)*a(9)-b(2)*a(8)-b(3)*a(7)-b(4)*a(6)-b(5)*a(5)-b(6)*a(4)-b(7)*a(3)-b(8)*a(2)
)/a(1);
b(10)=(b(9)-a(10)-b(1)*a(10)-b(2)*a(9)-b(3)*a(8)-b(4)*a(7)-b(5)*a(6)-b(6)*a(5)-b(7)*a(4)-b(8)*a
(3)-b(9)*a(2))/a(1);
b(11)=(b(10)-a(11)-b(1)*a(11)-b(2)*a(10)-b(3)*a(9)-b(4)*a(8)-b(5)*a(7)-b(6)*a(6)-b(7)*a(5)-b(8)
*a(4)-b(9)*a(3)-b(10)*a(2))/a(1);
b(12)=(b(11)-a(12)-b(1)*a(12)-b(2)*a(11)-b(3)*a(10)-b(4)*a(9)-b(5)*a(8)-b(6)*a(7)-b(7)*a(6)-b(
8)*a(5)-b(9)*a(4)-b(10)*a(3)-b(11)*a(2))/a(1);
lrate=0;

```

```
for i=13:712
```

```
    b(i)=(b(i-1)-b(i-12)*a(13)-b(i-11)*a(12)-b(i-10)*a(11)-b(i-9)*a(10)-b(i-8)*a(9)-b(i-7)*a(8)-b(i-6)*a(7)-b(i-5)*a(6)-b(i-4)*a(5)-b(i-3)*a(4)-b(i-2)*a(3)-b(i-1)*a(2))/a(1);
```

```
end
```

```
for i=713:750
```

```
    alfa(1,i+1)
```

```
    b(i)=(b(i-1)-b(i-12)*alfa(13,i-11-1)-b(i-11)*alfa(12,i-10-1)-b(i-10)*alfa(11,i-9-1)-b(i-9)*alfa(10,i-8-1)-b(i-8)*alfa(9,i-7-1)-b(i-7)*alfa(8,i-6-1)-b(i-6)*alfa(7,i-5-1)-b(i-5)*alfa(6,i-4-1)-b(i-4)*alfa(5,i-3-1)-b(i-3)*alfa(4,i-2-1)-b(i-2)*alfa(3,i-1-1)-b(i-1)*alfa(2,i-1))/alfa(1,i+1-1);
```

```
end
```

```
c=1;
```

```
for n=1:750
```

```
    c=c+b(n);
```

```
end
```

```
p0=1/c
```

```
L=0;
```

```
for n=1:750
```

```
    L=L+n*b(n)*p0;
```

```
end
```

```
f=0;
```

```
for n=714:719
```

```
    f=f+b(n)*p0*1/8;
```

```
end
```

```
for n=720:725
```

```
    f=f+b(n)*p0*2/8;
```

```
end
```

```
for n=726:731
```

```
    f=f+b(n)*p0*3/8;
```

```
end
```

```
for n=732:736
```

```
    f=f+b(n)*p0*4/8;
```

```
end
```

```
for n=737:741
```

```
    f=f+b(n)*p0*5/8;
```

```
end
```

```
for n=742:747
```

```
    f=f+b(n)*p0*6/8;
```

```
end
```

```
for n=748:750
```

```
    f=f+b(n)*p0*7/8;
```

```
end
```

```
f
```

```

c=1/(p0+roori/0.7-roori/0.7*f);
c
pi=zeros(752);

pi(1)=p0*c;

for i=2:751
    pi(i)=b(i-1)*c*p0;
end

pi(752)=1-c;

Lrate=f*c+1-c
L=0;

for i=1:752
    L=L+(i-1)*pi(i);
end
wait=(L)/0.7
Y=wait

```

## C.2 *newm7\_plot\_md1.m*

*newm7\_plot.m* is used to calculate the delay and loss rate of the NOC packets from the M/D/1 with dynamic queue model, given the traffic load of the NOC packets, value of  $p_1$  and the size of  $Q_0$ .

```

function Y=newm7_plot_md1 (x)
k=5;
roori=x;
ro2=1;
p=0.3;
a=zeros(200);
a2=0;
f=0;
for k=0:12
    ro=roori/0.7;
    a(k+1)=exp(-ro)*(ro)^k/prod(1:k);
end

alfa=zeros(13,750);
ro=roori/0.7;
for n=0:12
    for m=690:750
        if and((m+n)<=712,m~=712)

            x=0;
            for i=1:1000
                x=x+((ro*i).^n)/prod(1:n).*exp(-ro*i)*p.^(i-1)*(1-p);
            end
            alfa (n+1,m+1)=exp(-ro)*ro^n/prod(1:n);
        elseif (712<=m+n & m+n<=718)
            if (n==0)
                ro2=ro*7/8;
                ro3=(ro*7/8)^n;
                if (m==718)

```

```

        ro2=ro*6/8;
        ro3=(ro*6/8)^n;
    end
else
    ro3=ro^(select(712-m))*(ro*7/8)^((n-select(712-m)));
    ro2=(select(712-m)*ro+((n-select(712-m))*ro*7/8))/n;
end

x=0;
for i=1:1000
    x=x+(ro3*i^n)/prod(1:n).*exp(-ro2*i)*p.^(i-1)*(1-p);
end
alfa (n+1,m+1)=exp(-ro2)*ro3/prod(1:n);
elseif (718<(m+n) & m+n<=724)
    if (n==0)
        ro2=ro*6/8;
        ro3=(ro*6/8)^n;
        if (m==724)
            ro2=ro*5/8;
            ro3=(ro*5/8)^n;
        end
    end
else

ro3=ro^select(712-m)*(ro*7/8)^select(6-select(m-712))*(ro*6/8)^(n-select(718-m));

ro2=(select(712-m)*ro+select(6-select(m-712))*ro*7/8+(n-select(718-m))*ro*6/8)/n;

x=0;
for i=1:1000
    x=x+(ro3*i.^n)/prod(1:n).*exp(-ro2*i)*p.^(i-1)*(1-p);
end
alfa (n+1,m+1)=exp(-ro2)*ro3/prod(1:n);

elseif (724<(m+n)& m+n<=730)
    if (n==0)
        ro2=ro*5/8;
        ro3=(ro*5/8)^n;
        if (m==730)
            ro2=ro*4/8;
            ro3=(ro*4/8)^n;
        end
    end
else

ro3=ro^select(712-m)*(ro*7/8)^select(6-select(m-712))*(ro*6/8)^select(6-select(m-718))*(ro*5/8)^(n-select(724-m));

ro2=(select(712-m)*ro+select(6-select(m-712))*ro*7/8+select(6-select(m-718))*ro*6/8+(n-select(724-m))*ro*5/8)/n;

x=0;
for i=1:1000
    x=x+(ro3*i.^n)/prod(1:n).*exp(-ro2*i)*p.^(i-1)*(1-p);
end
alfa (n+1,m+1)=exp(-ro2)*ro3/prod(1:n);
elseif (730<(m+n)& m+n<=735)
    if (n==0)
        ro2=ro*4/8;

```

```

        ro3=(ro*4/8)^n;
        if (m==735)
            ro2=ro*3/8;
            ro3=(ro*3/8)^n;
        end
    else

ro3=ro^select(712-m)*(ro*7/8)^select(6-select(m-712))*(ro*6/8)^select(6-select(m-718))*(ro*5/8)^select(6-select(m-724))*(ro*4/8)^(n-select(730-m));

ro2=(select(712-m)*ro+select(6-select(m-712))*ro*7/8+select(6-select(m-718))*ro*6/8+select(6-select(m-724))*ro*5/8+(n-select(730-m))*ro*4/8)/n;

        end
        x=0;
        for i=1:1000
            x=x+(ro3*i.^n)/prod(1:n).*exp(-ro2*i)*p.^(i-1)*(1-p);
        end
        alfa (n+1,m+1)=exp(-ro2)*ro3/prod(1:n);
    elseif (735<(m+n)& m+n<=740)
        if (n==0)
            ro2=ro*3/8;
            ro3=(ro*3/8)^n;
            if (m==740)
                ro2=ro*2/8;
                ro3=(ro*2/8)^n;
            end
        end
    else

ro3=ro^select(712-m)*(ro*7/8)^select(6-select(m-712))*(ro*6/8)^select(6-select(m-718))*(ro*5/8)^select(6-select(m-724))*(ro*4/8)^select(5-select(m-730))*(ro*3/8)^(n-select(735-m));

ro2=(select(712-m)*ro+select(6-select(m-712))*ro*7/8+select(6-select(m-718))*ro*6/8+select(6-select(m-724))*ro*5/8+select(5-select(m-730))*ro*4/8+(n-select(735-m))*ro*3/8)/n;

        end

        alfa (n+1,m+1)=exp(-ro2)*ro3/prod(1:n);
    elseif (740<(m+n)& m+n<=746)
        if (n==0)
            ro2=ro*2/8;
            ro3=(ro*2/8)^n;
            if (m==746)
                ro2=ro*1/8;
                ro3=(ro*1/8)^n;
            end
        end
    else

ro3=ro^select(712-m)*(ro*7/8)^select(6-select(m-712))*(ro*6/8)^select(6-select(m-718))*(ro*5/8)^select(6-select(m-724))*(ro*4/8)^select(5-select(m-730))*(ro*3/8)^select(5-select(m-735))*(ro*2/8)^(n-select(740-m));

ro2=(select(712-m)*ro+select(6-select(m-712))*ro*7/8+select(6-select(m-718))*ro*6/8+select(6-select(m-724))*ro*5/8+select(5-select(m-730))*ro*4/8+select(5-select(m-735))*ro*3/8+(n-select(740-m))*ro*2/8)/n;

        end

        alfa (n+1,m+1)=exp(-ro2)*ro3/prod(1:n);
    elseif (746<(m+n)& m+n<=750)

```

```

if (n==0)
    ro2=ro*1/8;
    ro3=(ro*1/8)^n;
else

ro3=ro^select(712-m)*(ro*7/8)^select(6-select(m-712))*(ro*6/8)^select(6-select(m-718))*(ro*5/8)^select(6-select(m-724))*(ro*4/8)^select(5-select(m-730))*(ro*3/8)^select(5-select(m-735))*(ro*2/8)^select(5-select(m-740))*(ro*1/8)^(n-select(746-m));

ro2=(select(712-m)*ro+select(6-select(m-712))*ro*7/8+select(6-select(m-718))*ro*6/8+select(6-select(m-724))*ro*5/8+select(5-select(m-730))*ro*4/8+select(5-select(m-735))*ro*3/8+select(5-select(m-740))*ro*2/8+(n-select(746-m))*ro*1/8)/n;
end
x=0;
for i=1:1000
    x=x+(ro3*i^n)/prod(1:n).*exp(-ro2*i)*p.^(i-1)*(1-p);
end
alfa (n+1,m+1)=exp(-ro2)*ro3/prod(1:n);
end
end
end

b=zeros(750);
b(1)=(1-a(1))/a(1);
b(2)=(b(1)-a(2)-b(1)*a(2))/a(1);
b(3)=(b(2)-a(3)-b(1)*a(3)-b(2)*a(2))/a(1);
b(4)=(b(3)-a(4)-b(1)*a(4)-b(2)*a(3)-b(3)*a(2))/a(1);
b(5)=(b(4)-a(5)-b(1)*a(5)-b(2)*a(4)-b(3)*a(3)-b(4)*a(2))/a(1);
b(6)=(b(5)-a(6)-b(1)*a(6)-b(2)*a(5)-b(3)*a(4)-b(4)*a(3)-b(5)*a(2))/a(1);
b(7)=(b(6)-a(7)-b(1)*a(7)-b(2)*a(6)-b(3)*a(5)-b(4)*a(4)-b(5)*a(3)-b(6)*a(2))/a(1);
b(8)=(b(7)-a(8)-b(1)*a(8)-b(2)*a(7)-b(3)*a(6)-b(4)*a(5)-b(5)*a(4)-b(6)*a(3)-b(7)*a(2))/a(1);
b(9)=(b(8)-a(9)-b(1)*a(9)-b(2)*a(8)-b(3)*a(7)-b(4)*a(6)-b(5)*a(5)-b(6)*a(4)-b(7)*a(3)-b(8)*a(2))/a(1);
b(10)=(b(9)-a(10)-b(1)*a(10)-b(2)*a(9)-b(3)*a(8)-b(4)*a(7)-b(5)*a(6)-b(6)*a(5)-b(7)*a(4)-b(8)*a(3)-b(9)*a(2))/a(1);
b(11)=(b(10)-a(11)-b(1)*a(11)-b(2)*a(10)-b(3)*a(9)-b(4)*a(8)-b(5)*a(7)-b(6)*a(6)-b(7)*a(5)-b(8)*a(4)-b(9)*a(3)-b(10)*a(2))/a(1);
b(12)=(b(11)-a(12)-b(1)*a(12)-b(2)*a(11)-b(3)*a(10)-b(4)*a(9)-b(5)*a(8)-b(6)*a(7)-b(7)*a(6)-b(8)*a(5)-b(9)*a(4)-b(10)*a(3)-b(11)*a(2))/a(1);
lrate=0;

for i=13:712

b(i)=(b(i-1)-b(i-12)*a(13)-b(i-11)*a(12)-b(i-10)*a(11)-b(i-9)*a(10)-b(i-8)*a(9)-b(i-7)*a(8)-b(i-6)*a(7)-b(i-5)*a(6)-b(i-4)*a(5)-b(i-3)*a(4)-b(i-2)*a(3)-b(i-1)*a(2))/a(1);
end

for i=713:750
    alfa(1,i+1)

b(i)=(b(i-1)-b(i-12)*alfa(13,i-11)-b(i-11)*alfa(12,i-10)-b(i-10)*alfa(11,i-9)-b(i-9)*alfa(10,i-8)-b(i-8)*alfa(9,i-7)-b(i-7)*alfa(8,i-6)-b(i-6)*alfa(7,i-5)-b(i-5)*alfa(6,i-4)-b(i-4)*alfa(5,i-3)-b(i-3)*alfa(4,i-2)-b(i-2)*alfa(3,i-1)-b(i-1)*alfa(2,i))/alfa(1,i+1);

```



end

```
c=1;
for n=1:750
    c=c+b(n);
end
```

```
p0=1/c
L=0;
for n=1:750
    L=L+n*b(n)*p0;
end
```

```
f=0;
for n=714:719
    f=f+b(n)*p0*1/8;
end
```

```
for n=720:725
    f=f+b(n)*p0*2/8;
end
```

```
for n=726:731
    f=f+b(n)*p0*3/8;
end
```

```
for n=732:736
    f=f+b(n)*p0*4/8;
end
```

```
for n=737:741
    f=f+b(n)*p0*5/8;
end
```

```
for n=742:747
    f=f+b(n)*p0*6/8;
end
```

```
for n=748:750
    f=f+b(n)*p0*7/8;
end
```

```
f
c=1/(p0+roori/0.7-roori/0.7*f);
c
pi=zeros(752);
```

```
pi(1)=p0*c;
```

```
for i=2:751
    pi(i)=b(i-1)*c*p0;
end
```

```
pi(752)=1-c;
```

```
Lrate=f*c+1-c
L=0;

for i=1:752
    L=L+(i-1)*pi(i);
end
wait=(L)/0.7;
Y=wait
```

### C.3 *select.m*

```
function y = select(x)
if (x>0)
    y=x;
else
    y=0;
end
```

### C.4 *plotm.m*

*plotm.m* is used to plot the result of mean delays from M/B/1 with dynamic queue model, M/D/1 with dynamic queue model and from the simulator *NodeSim*.

```
X=[0.702,0.703,0.704,0.705,0.706,0.708,0.710,0.715,0.72,0.73,0.74,0.75,0.76,0.77,0.78,0.79,0.80
,0.82,0.84,0.86,0.88,0.90,0.92,0.94,0.96,0.98,1.00,1.02,1.04,1.06,1.08];
```

```
Y=[700,820,875,910,935,961,973,990,1000,1009,1013,1016,1018,1020,1021,1022,1023,1025,10
26,1028,1029,1031,1032,1033,1034,1035,1037,1038,1039,1040,1040];
```

```
plot(X,Y,'r')
hold on
fplot('newm7_plot',[0.702,1.08],'g')
hold on
fplot('newm7_plot_md1',[0.702,1.08],'b')
hh = xlabel('traffic load of NOC packets')
hh = ylabel('average waiting time of NOC packet (10E-6 second)')
hh= title ('Figure 2: comparison of the average waiting time of M/D/1 model and M/B/1 model
with the simulation result')
legend ('from simulation','from M/B/1 model with dynamic queue','from M/D/1 model with
dynamic queue')
```

### C.5 *plotm\_lr.m*

*plotm.m* is used to plot the result of loss rate from M/B/1 with dynamic queue model, M/D/1 with dynamic queue model and from the simulator *NodeSim*.

```
X=[0.702,0.703,0.704,0.705,0.706,0.708,0.710,0.715,0.720,0.730,0.740,0.750,0.760,0.770,0.780,0.790
,0.800,0.820,0.840,0.86,0.88,0.90,0.92,0.94,0.96,0.98,1.00,1.02,1.04,1.06,1.08];
```

```
Y=[0.0018,0.0033,0.0047,0.0063,0.0076,0.0101,0.0132,0.0200,0.026,0.040,0.053,0.066,0.078,0.089,0.
101,0.113,0.124,0.145,0.165,0.185,0.204,0.221,0.238,0.254,0.270,0.285,0.299,0.313,0.326,0.339,0.351
];
```

```
plot(X,Y,'r')
hold on
fplot('newm7_plot_lr',[0.702,1.08],'g')
hold on
fplot('newm7_plot_md1_lr',[0.702,1.08],'b')
hh = xlabel('traffic load of NOC packets')
hh = ylabel('loss rate of NOC packet (10E-6 second)')
hh= title ('Figure 2: comparison of the loss rate of M/D/1 model and M/B/1 model with the simulation
result')
legend ('from simulation','from M/B/1 model with dynamic queue','from M/D/1 model with dynamic
queue')
```

## C.6 Poissonest.m

*Poissonest.m* is used to simulate the special non-homogeneous Poisson Process described in Appendix B to verify Equation B-16.

```
rp=7500;
a=zeros(rp);
ro=[0.8,0.6,0.5,0.3,0.7,0.6,0.5,0.4,0.3,0.1,0.1,0.1,0.1,1,1,1,1,1];
deadline=1;
nmax=1;

for i=1:rp
    n=1;
    time=exprnd(1/ro(n));
    ro2=0;
    ro3=1;
    while (time<deadline)
        n=n+1;
        time=time+exprnd(1/ro(n));
    end
    a(i)=n-1;
    if (n-1>nmax)
        nmax=n;
    end
end

nmax
```

```
c=zeros(nmax);
for i=0:nmax-1
    for j=1:rp
        if (a(j)==i)
            c(i+1)=c(i+1)+1;
        end
    end
    c(i+1)=c(i+1)/rp;
end

b(1)=exp(-ro(1));
for i=1:nmax-1
    ro2=0;
    ro3=1;
    for x=1:i
        ro2=ro2+ro(x);
        ro3=ro3*ro(x);
    end
    b(i+1)=exp(-ro2/i)*ro3/prod(1:i);
end
c
b
```

## Appendix D

# The Scripts of the Simulators

This chapter gives the scripts of *NodeSim* in Simula. The scripts of *Routesim* is too large (3676 lines) to be attached here, but the basic idea and components of *Routesim* is similar with the scripts of *NodeSim*

```
!*****.  
!*                                     *;  
!* DMP Node output link queueing, (version q1q2e) *;  
!* adaptive B2 dropping mechanism is implemented *;  
!* Designer: LAR, Nov-Dec 2006 *;  
!*                                     *;  
!*****.
```

```
!* Make sure this path points to the location of your 'demos.atr' file. *;  
!* This is the path if cim is installed on m:/ *;
```

```
EXTERNAL CLASS demos="c:/cim/demos.atr";
```

```
demos begin
```

```
ref (outfile) outf1, outf2, outf3, outf4;
```

```
ref (waitq) Q1, Q2, B2;
```

```
ref (condq) cq;
```

```
ref (CTRL) PCTRL;
```

```
ref (NOC) PNOC;
```

```
ref (histogram) TotTimeCTRL, TotTimeNOC;
```

```
ref (count) CntB1DrNOC8, CntB1DrNOC7, CntB1DrNOC6,  
CntB1DrNOC5, CntB1DrNOC4, CntB1DrNOC3,  
CntB1DrNOC2, CntB1DrNOC1;
```

```
ref (rdist) nextCTRL, nextNOC, nextNOC2, xdraw;
```

```
long real simTime, ht, Q1m,Q1n, changeTimeNOC1,  
      changeTimeNOC2, changeTimeNOC3, changeTimeCTRL;
```

```
real holds;
```

```
real p1,tx;
```

```
integer ix, buflen,CntQ1DrNOC;
```

```
boolean procedure and2(a,b); name a,b; boolean a,b;  
and2 := if a then b else false;
```

```
boolean procedure or2(a,b); name a,b; boolean a,b;  
or2 := if a then true else b;
```

```
!*****  
!* Control packet generation *;  
!*****
```

```
entity class CTRL;  
begin  
  ref (packetCTRL) pk1;  
  integer SN;  
  
  SN := 1;  
  while time < changeTimeCTRL do  
  begin  
    hold(nextCTRL.sample);  
    new packetCTRL("packetCTRL",1,SN, 50&3).schedule(0.0);  
    SN := SN +1;  
  end;  
  
end***CTRL***;
```

```
!*****  
!* NOC packet generation *;  
!*****
```

```
entity class NOC;  
begin  
  ref (packetNOC) pk2;  
  integer SN;  
  
  SN := 1;  
  while time < changeTimeNOC1 do  
  begin  
    hold(nextNOC.sample);  
    new packetNOC("packetNOC",2,SN,750&3).schedule(0.0);  
    if SN < 9 then  
    begin  
      SN := SN + 1;  
    end else  
    begin  
      SN := 1;  
    end  
  end  
end
```

```

        end;
    end;

end***NOC***;

```

```

!*****.
!*   Control Packet                               *.
!*****.

```

```

entity class packetCTRL(PT, SeqNo,Rate);
integer PT, SeqNo;
real Rate;

```

```

begin
    long real t1;
    t1 := time;

    cq.signal;
    Q2.wait;

    Q1.wait;

    if PT = 1 then
        begin
            TotTimeCTRL.update(time - t1);
            outf4.outfix(time-t1, 9, 14);
        end;
    end;
end***packet***;

```

```

!*****.
!*   NOC Packet                                   *.
!*                                               *.
!* (ex. 7/4 = 1,75 || 7//4 = 1 )                 *.
!*****.

```

```

entity class packetNOC(PT, SeqNo,Rate);
integer PT, SeqNo;
real Rate;

```

```

begin
    integer l1;
    long real t1;
    real help1;
    t1 := time;

    if B2.length <= 12 then
        begin
            cq.signal;
            B2.wait;
        end else

    if and2(B2.length > 12, B2.length <= 18) then

```

```
begin
  if SeqNo/8 eq SeqNo//8 then
    begin
      CntB1DrNOC8.update(1);
      goto Lterm;
    end else
    begin
      cq.signal;
      B2.wait;
    end;
  end else

  if and2(B2.length > 18, B2.length <= 24) then
    begin
      if SeqNo/8 eq SeqNo//8 then
        begin
          CntB1DrNOC8.update(1);
          goto Lterm;
        end else
        if SeqNo/7 eq SeqNo//7 then
          begin
            CntB1DrNOC7.update(1);
            goto Lterm;
          end else
          begin
            cq.signal;
            B2.wait;
          end;
        end else

        if and2(B2.length > 24, B2.length <= 30) then
          begin
            if SeqNo/8 eq SeqNo//8 then
              begin
                CntB1DrNOC8.update(1);
                goto Lterm;
              end else
              if SeqNo/7 eq SeqNo//7 then
                begin
                  CntB1DrNOC7.update(1);
                  goto Lterm;
                end else
                if SeqNo/6 eq SeqNo//6 then
                  begin
                    CntB1DrNOC6.update(1);
                    goto Lterm;
                  end else
                  begin
                    cq.signal;
                    B2.wait;
                  end;
                end else

                if and2(B2.length > 30, B2.length <= 36) then
                  begin
                    if SeqNo/8 eq SeqNo//8 then
                      begin
                        CntB1DrNOC8.update(1);
```



```
        goto Lterm;
    end else
    if SeqNo/7 eq SeqNo//7 then
    begin
        CntB1DrNOC7.update(1);
        goto Lterm;
    end else
    if SeqNo/6 eq SeqNo//6 then
    begin
        CntB1DrNOC6.update(1);
        goto Lterm;
    end else
    if SeqNo/5 eq SeqNo//5 then
    begin
        CntB1DrNOC5.update(1);
        goto Lterm;
    end else
    begin
        cq.signal;
        B2.wait;
    end;
end else

if and2(B2.length > 36, B2.length <= 40) then
begin
    if SeqNo/8 eq SeqNo//8 then
    begin
        CntB1DrNOC8.update(1);
        goto Lterm;
    end else
    if SeqNo/7 eq SeqNo//7 then
    begin
        CntB1DrNOC7.update(1);
        goto Lterm;
    end else
    if SeqNo/6 eq SeqNo//6 then
    begin
        CntB1DrNOC6.update(1);
        goto Lterm;
    end else
    if SeqNo/5 eq SeqNo//5 then
    begin
        CntB1DrNOC5.update(1);
        goto Lterm;
    end else
    if SeqNo/4 eq SeqNo//4 then
    begin
        CntB1DrNOC4.update(1);
        goto Lterm;
    end else
    begin
        cq.signal;
        B2.wait;
    end;
end else

if and2(B2.length > 40, B2.length <= 46) then
begin
```

```
if SeqNo/8 eq SeqNo//8 then
begin
  CntB1DrNOC8.update(1);
  goto Lterm;
end else
if SeqNo/7 eq SeqNo//7 then
begin
  CntB1DrNOC7.update(1);
  goto Lterm;
end else
if SeqNo/6 eq SeqNo//6 then
begin
  CntB1DrNOC6.update(1);
  goto Lterm;
end else
if SeqNo/5 eq SeqNo//5 then
begin
  CntB1DrNOC5.update(1);
  goto Lterm;
end else
if SeqNo/4 eq SeqNo//4 then
begin
  CntB1DrNOC4.update(1);
  goto Lterm;
end else
if SeqNo/3 eq SeqNo//3 then
begin
  CntB1DrNOC3.update(1);
  goto Lterm;
end else
begin
  cq.signal;
  B2.wait;
end;
end else

if and2(B2.length > 46, B2.length <= 50) then
begin
  if SeqNo/8 eq SeqNo//8 then
  begin
    CntB1DrNOC8.update(1);
    goto Lterm;
  end else
  if SeqNo/7 eq SeqNo//7 then
  begin
    CntB1DrNOC7.update(1);
    goto Lterm;
  end else
  if SeqNo/6 eq SeqNo//6 then
  begin
    CntB1DrNOC6.update(1);
    goto Lterm;
  end else
  if SeqNo/5 eq SeqNo//5 then
  begin
    CntB1DrNOC5.update(1);
    goto Lterm;
  end else
```

```

    if SeqNo/4 eq SeqNo//4 then
    begin
        CntB1DrNOC4.update(1);
        goto Lterm;
    end else
    if SeqNo/3 eq SeqNo//3 then
    begin
        CntB1DrNOC3.update(1);
        goto Lterm;
    end else
    if SeqNo/2 eq SeqNo//2 then
    begin
        CntB1DrNOC2.update(1);
        goto Lterm;
    end else
    begin
        cq.signal;
        B2.wait;
    end;
end else goto Lterm;

Q1.wait;

if PT = 2 then
begin
    TotTimeNOC.update(time - t1);
    outf3.outfix(time-t1, 9, 14);
end;

Lterm:

end***packet***;

!*****.
!*   Box server                               *.
!*****.

entity class Box;
begin
    ref (entity) ep;
    real p;

loop:
    cq.waituntil(or2(
        and2(Q2.length > 0,Q1.length < Q1n),
        and2(B2.length > 0,Q1.length < Q1n)));

    begin
        p := xdraw.sample;
        ep :- none;

        if p > p1 then
            begin

```

```

        if B2.length > 0 then ep :- B2.coopt;
    end else
    begin
        if Q2.length > 0 then ep :- Q2.coopt;
    end;

    if ep /= none then
    begin
        ep.schedule(now);
    end;
    end;

    repeat;
end***Box***;

```

```

!*****.
!*   Output link server           *.
!*****.

```

```

entity class H;
begin
    ref (entity) ep;
loop:
    ep :- Q1.coopt;
    cq.signal;
    hold(ht);
    ep.schedule(0.0);
    repeat;
end***H***;

```

```

!*****.
!*   Simulation:                 *.
!*                               *.
!*   one second is used as time unit      *.
!*   rates in packets per second         *.
!*****.

```

```

INF :- NEW INFILE("Input_NodeSim.txt");
inf.open(blanks(80));
ht :=inf.inreal;
p1 :=inf.inreal;
Q1n :=inf.inint;
holds :=inf.inreal;
readdist(nextCTRL, "nextCTRL");
readdist(nextNOC, "nextNOC");

```

```

CntB1DrNOC8 :- new count("CntB1DrNOC8");
CntB1DrNOC7 :- new count("CntB1DrNOC7");
CntB1DrNOC6 :- new count("CntB1DrNOC6");
CntB1DrNOC5 :- new count("CntB1DrNOC5");
CntB1DrNOC4 :- new count("CntB1DrNOC4");
CntB1DrNOC3 :- new count("CntB1DrNOC3");

```

```
CntB1DrNOC2 :- new count("CntB1DrNOC2");
CntB1DrNOC1 :- new count("CntB1DrNOC1");

xdraw :- new uniform("xdraw",0.0, 1.0);

Q1 :- new waitq("Q1");
Q2 :- new waitq("Q2");
B2 :- new waitq("B2");

cq :- new condq("cq");

TotTimeCTRL :- new histogram("TotTimeCTRL",1&-6,11&-3,11);
TotTimeNOC :- new histogram("TotTimeNOC",1&-6, 1.1&-3,11);

buflength := 50;

Q1m := 1.1&3;

outf1 :- new outfile("NodeSim_overview.txt");
outf1.open(blanks(120));

PCTRL :- new CTRL("CTRL");
PCTRL.schedule(0.0);
PNOC :- new NOC("NOC");
PNOC.schedule(0.0);

new Box("Box").schedule(0.0);
new H("H").schedule(0.0);

changeTimeNOC1 := 1000;
changeTimeNOC2 := 50;
changeTimeNOC3 := 10;

changeTimeCTRL := 1000;

outf2 :- new outfile("NodeSim_drop.txt");
outf2.open(blanks(120));
outf2.outtext("OK");
outf2.outimage;
outf2.outtext ("Incoming      drop      drop      drop      drop      drop      drop      drop
drop      drop");
outf2.outimage;
outf2.outtext ("Packets      SN_8      SN_7      SN_6      SN_5      SN_4      SN_3
SN_2      SN_1      Rate");
outf2.outimage;

outf :- outf1;
```

```

outf3 :- new outfile ("NodeSim_delay_NOC.txt");
outf3.open(blanks(120));
outf3.outtext("OK");
outf3.outimage;

outf4 :- new outfile ("NodeSim_delay_Ctrl.txt");
outf4.open(blanks(120));
outf4.outtext("OK");
outf4.outimage;
!*****.
!* Output files:                *;
!*                               *;
!* one second is used as time unit *;
!* rates in packets per second   *;
!*****.

notrace;
noreport;

begin

outf1.outimage;
outf1.outtext("*****");
outf1.outtext("*****");
outf1.outimage;
outf1.outtext("*****");
outf1.outtext("*****");
outf1.outimage;
outf1.outtext("*****");
outf1.outtext("*****");
outf1.outimage;
outf1.outtext("p1 = ");
outf1.outfix(p1,2,13);
outf1.outimage;
outf1.outtext("buflength = ");
outf1.outint(buflength,4);
outf1.outimage;
length outf1.outtext("Total Average delay Maximun delay Average delay Maximun delay
length length length");
outf1.outimage;
Q0 outf1.outtext("packets Ctrl Packets Ctrl Packets NOC packets NOC packets
Q1 Q2");
outf1.outimage;

for ix := 1 step 1 until 6 do
begin
hold(holds);
outf1.outint(nextCTRL.obs + nextNOC.obs,7);
if TotTimeCTRL.Myt.Obs > 0 then
outf1.outreal(TotTimeCTRL.Myt.Sum/TotTimeCTRL.Myt.Obs,6,15);
outf1.outreal(TotTimeCTRL.Myt.Max,4,13);
if TotTimeNOC.Myt.Obs > 0 then
begin
outf1.outreal(TotTimeNOC.Myt.Sum/TotTimeNOC.Myt.Obs,6,15);
end else outf1.outint(0, 15);
outf1.outreal(TotTimeNOC.Myt.Max,4,13);
outf1.outint(Q1.LENGTH,9);

```

```
    outf1.outint(Q2.LENGTH,7);
    outf1.outint(B2.LENGTH,5);
    outf1.outimage;

    outf2.outint(nextNOC.obs,7);
    outf2.outint(CntB1DrNOC8.obs,8);
    outf2.outint(CntB1DrNOC7.obs,8);
    outf2.outint(CntB1DrNOC6.obs,8);
    outf2.outint(CntB1DrNOC5.obs,8);
    outf2.outint(CntB1DrNOC4.obs,8);
    outf2.outint(CntB1DrNOC3.obs,8);
    outf2.outint(CntB1DrNOC2.obs,8);
    outf2.outint(CntB1DrNOC1.obs,8);

outf2.outreal((CntB1DrNOC1.obs+CntB1DrNOC2.obs+CntB1DrNOC3.obs+CntB1DrNOC4.obs
+CntB1DrNOC5.obs+CntB1DrNOC6.obs+CntB1DrNOC7.obs+CntB1DrNOC8.obs)/nextNOC.o
bs,5,16);
    outf2.outimage;

        reset;
    end;
end;

!   outf1.close;
    outf2.close;

end***demos***;
```