A Markov Chain Model for the RIO Algorithm in Differentiated Services Networks

by

Junzhang Liang

A thesis submitted to the graduate faculty in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Engineering

Program of Study Committee
Ahmed E. Kamal, Major Professor
Manimaran Govindarasu
Thomas Barta

Iowa State University
Ames, Iowa
2001

Copyright © Junzhang Liang, 2001. All rights reserved.
This is to certify that the master's thesis of

Junzhang Liang

has met the thesis requirements of Iowa State University

Signatures have been redacted for privacy
# TABLE OF CONTENTS

**LIST OF FIGURES** vi

**ACKNOWLEDGEMENTS** v

**CHAPTER 1. INTRODUCTION**
1.1 Introduction to High Performance Networks 1
1.2 The Integrated Services Architecture 2
1.3 The Differentiated Services Architecture 7
1.4 Congestion Control and Queue Management 11
1.5 Thesis Contribution and Outline 13

**CHAPTER 2. ACTIVE QUEUE MANAGEMENT**
2.1 Introduction 14
2.2 Introduction to the Early Random Detection (RED) Algorithm 15
2.3 Introduction to the RED with in and out (RIO) Algorithm 18
2.4 Models of Active Queue Management Technique 20
2.5 Summary 22

**CHAPTER 3. ANALYTICAL MODELING OF RIO**
3.1 Maximum Threshold Calculation 23
3.2 The Markov Chain Model Assumptions and Approximations 27
3.3 Markov Model and Transition Probability Matrix Computation 31
3.4 Performance Measures 41
3.5 Summary 43

**CHAPTER 4. SIMULATION RESULTS**
4.1 Evaluation of the Maximum Threshold of RED 44
4.2 Network Model 48
4.3 Effect of Threshold Values on RIO Performance 50
4.4 Effect of Traffic Mix 51
4.5 Effect of the Maximum Dropping Probability 52
4.6 Summary 53

**CHAPTER 5  SUMMARY AND CONCLUSIONS**
5.1 Thesis Contributions 61
5.2 Future Directions 62

**REFERENCES** 64
# LIST OF FIGURES

| FIGURE 1.1 | (a) Architecture of the Integrated Service; (b) Workflow of RSVP; (c) Functional blocks of IS routers | 6 |
| FIGURE 1.2 | (a) Architecture of DiffServ; (b) Functional blocks | 10 |
| FIGURE 1.3 | Relation between the amount of input traffic and the *goodput* | 11 |
| FIGURE 2.1 | The RED algorithm | 17 |
| FIGURE 2.2 | The RIO algorithm | 19 |
| FIGURE 3.1 | Maximum threshold as a function of the initial instantaneous queue size and the initial average queue size (n=2, B=50) | 27 |
| FIGURE 3.2 | The network topology | 28 |
| FIGURE 4.1 | The network topology in RED simulations | 45 |
| FIGURE 4.2 | The instantaneous queue size (max=30) | 47 |
| FIGURE 4.3 | The instantaneous queue size (max=5) | 47 |
| FIGURE 4.4 | The instantaneous queue size (max=4) | 48 |
| FIGURE 4.5 | The network topology in RIO simulations | 49 |
| FIGURE 4.6 | Average queue sizes for IN and OUT queues (max_in=8, min_in=4, max_out=4, min_out=2, Pmax_in=0.1, Pmax_out=0.5, ratio=0.1) | 54 |
| FIGURE 4.7 | Average queue sizes for IN and OUT queues (max_in=8, min_in=4, max_out=4, min_out=2, Pmax_in=0.1, Pmax_out=0.5, ratio=0.3) | 54 |
| FIGURE 4.8 | Average queue sizes for IN and OUT queues (max_in=8, min_in=4, max_out=4, min_out=2, Pmax_in=0.1, Pmax_out=0.5, ratio=0.5) | 55 |
| FIGURE 4.9 | Average queue sizes for IN and OUT queues (max_in=8, min_in=4, max_out=4, min_out=2, Pmax_in=0.1, Pmax_out=0.5, ratio=0.9) | 55 |
| FIGURE 4.10 | Average queue sizes for IN and OUT queues (max_in=20, | |
|
\(\text{min}_{\text{in}}=10, \text{max}_{\text{out}}=10, \text{min}_{\text{out}}=5, \text{Pmax}_{\text{in}}=0.1, \text{Pmax}_{\text{out}}=0.5, \text{ratio}=0.1\)  

FIGURE 4.11 Average queue sizes for IN and OUT queues (\(\text{max}_{\text{in}}=20, \text{min}_{\text{in}}=10, \text{max}_{\text{out}}=10, \text{min}_{\text{out}}=5, \text{Pmax}_{\text{in}}=0.1, \text{Pmax}_{\text{out}}=0.5, \text{ratio}=0.3\))  

FIGURE 4.12 Average queue sizes for IN and OUT queues (\(\text{max}_{\text{in}}=20, \text{min}_{\text{in}}=10, \text{max}_{\text{out}}=10, \text{min}_{\text{out}}=5, \text{Pmax}_{\text{in}}=0.1, \text{Pmax}_{\text{out}}=0.5, \text{ratio}=0.5\))  

FIGURE 4.13 Average queue sizes for IN and OUT queues (\(\text{max}_{\text{in}}=20, \text{min}_{\text{in}}=10, \text{max}_{\text{out}}=10, \text{min}_{\text{out}}=5, \text{Pmax}_{\text{in}}=0.1, \text{Pmax}_{\text{out}}=0.5, \text{ratio}=0.9\))  

FIGURE 4.14 Average queue sizes for IN and OUT queues (\(\text{max}_{\text{in}}=30, \text{min}_{\text{in}}=15, \text{max}_{\text{out}}=15, \text{min}_{\text{out}}=7, \text{Pmax}_{\text{in}}=0.1, \text{Pmax}_{\text{out}}=0.5, \text{ratio}=0.1\))  

FIGURE 4.15 Average queue sizes for IN and OUT queues (\(\text{max}_{\text{in}}=30, \text{min}_{\text{in}}=15, \text{max}_{\text{out}}=15, \text{min}_{\text{out}}=7, \text{Pmax}_{\text{in}}=0.1, \text{Pmax}_{\text{out}}=0.5, \text{ratio}=0.3\))  

FIGURE 4.16 Average queue sizes for IN and OUT queues (\(\text{max}_{\text{in}}=30, \text{min}_{\text{in}}=15, \text{max}_{\text{out}}=15, \text{min}_{\text{out}}=7, \text{Pmax}_{\text{in}}=0.1, \text{Pmax}_{\text{out}}=0.5, \text{ratio}=0.5\))  

FIGURE 4.17 Average queue sizes for IN and OUT queues (\(\text{max}_{\text{in}}=30, \text{min}_{\text{in}}=15, \text{max}_{\text{out}}=15, \text{min}_{\text{out}}=7, \text{Pmax}_{\text{in}}=0.1, \text{Pmax}_{\text{out}}=0.5, \text{ratio}=0.9\))  

FIGURE 4.18 Average queue sizes for IN and OUT queues under different dropping probabilities (\(\text{max}_{\text{in}}=8, \text{min}_{\text{in}}=4, \text{max}_{\text{out}}=4, \text{min}_{\text{out}}=2, \text{ratio}=0.1\), (i) \(\text{Pmax}_{\text{in}}=0.1, \text{Pmax}_{\text{out}}=0.5\) (ii) \(\text{Pmax}_{\text{in}}=0.8, \text{Pmax}_{\text{out}}=0.9\)
Acknowledgements

The study experience at Iowa State University is both productive and enjoyable to me. I had the chance to meet a number of talented and warm-hearted people. First and foremost, I would like to express my sincere gratitude to my major advisor, Dr. Ahmed E. Kamal, for his invaluable guidance and support. His clarity of thought, taste of perfection and kindness impressed me.

I wish to express my sincere gratitude to graduate committee members, Dr. Govindarasu Manimaran and Dr. Tomas Barta, for their help and guidance during my courses and their generous contributions of time and effort in the process of this research project.

Various colleagues have influenced my work, and helped during the course of the research. I appreciate Jianwei Zhou and Bin Lin's valuable help and suggestions on my research. I would also like to thank Murali K. Viswanathan and Anirban Chakrabarti for helping me solving OPNET and ns related problems.

Finally I would like to offer my sincere appreciation to my parents. They have been continuously giving me strong support.
CHAPTER 1 INTRODUCTION

1.1 Introduction to High Performance Networks

Networks are used to transfer data between different applications. Based on different connection mode, network transfer service can be classified as *connectionless* and *connection-oriented*. Connection-oriented service delivers messages from the source to the destination in the correct order, while connectionless service transfers each packet of data to the destination one at a time, independently of the other packets, and therefore the transferred packets may arrive out of order. Applications that expect reliable and ordered transmissions of messages and guaranteed quality of service require connection-oriented service.

Today’s Internet only provides best-effort services, which is connectionless. Traffic is processed as quickly as possible, but there is no guarantee as to the timeliness of delivery, or even actual delivery. With the rapid transformation of the Internet into a commercial infrastructure, demands for service quality have rapidly developed. Future high performance networks should provide different types of service, which suit different types of application supported by the networks. In one extreme, they should provide hard guarantees in terms of bandwidth availability, delay variability, and data losses. And, in another extreme, they should continue to support the best-effort type of service. That is, high performance networks should provide different levels of Quality of Services (QoS). That is, they have the ability to reserve resources within the network and terminal devices so as to ensure that certain perceptual or objective performance measures are met [17]. Future internets will also be
scalable to support millions of users, and flexible and extensible to accommodate future applications.

The Internet Engineering Task Force (IETF) has proposed several service models and mechanisms to meet the demand for QoS. The Integrated Services model [12] and the Differentiated Services model [13] are the two most important ones. In the following two sections, we will introduce these two service models

1.2 The Integrated Services Architecture

The integrated services architecture (ISA) [12] was introduced by IETF to provide different levels of services to the different flows. This requires the identification of flows and their requirements. In addition to best-effort service, the Integrated Services model proposes two services classes. They are:

- Guaranteed service [14], which provides the users (applications) with an assured amount of bandwidth, firm end-to-end delay bounds, and no queuing loss for flows that conform to the parameters negotiated at the connection setup;
- Controlled-load service [15], which does not provide the network users with any firm quantitative guarantees, but assures that the users will get a service that is as close as possible to the one received by a best-effort flow in a lightly loaded network. This assurance is granted, provided the flow conforms to the traffic characteristics negotiated at session setup.

The ISA relies on resource reservation in order to provide the guaranteed type of service for each flow. For real-time applications, before data is transmitted, the applications must
first set up paths and reserve resources, and the resources are available during the lifetime of
the services.

The philosophy of this model is that “there is an inescapable requirement for routers to be
able to reserve resources in order to provide special QoS for specific user packet streams, or
flows”. This flow-based reservation requires flow-specific state handling in the router.

The ISA therefore employs the following four components:

- Packet Scheduler. The packet scheduler manages the forwarding of different packet
  streams in manners congruent with the QoS requirements using a set of queues and
  perhaps other mechanisms such as timers. Therefore, packet scheduling must be
  implemented at the point where packets are queued.

- Packet Classifier. The packet classifier operates upstream of the packet scheduler and
  maps each incoming packet into some class, in such a way that all packets in the same
  class get the same treatment from the packet scheduler. Packets are placed into
  specific queues corresponding to different classes.

- Admission Control Routine. The admission control routine implements the decision
  algorithm that a router or host uses to determine whether a new flow can be granted
  the requested QoS without impacting earlier guarantees. The admission control
  routine takes responsibility for enforcing the reservation policies set by the network
  administrator.

- The signaling protocol for setting up paths and reserving resources, namely, the
  Resource Reservation Protocol (RSVP [12]). An adequate reservation protocol faces a
  fourfold trial mainly related to routing:
RSVP reserves a portion of the output link in each router along the path of a flow. The sender periodically sends out a PATH message, which describes the type of traffic being sent and the resource requirements necessary to support the traffic stream. A receiver that gets the PATH message responds by sending a reserve (RESV) message toward the sender, tracing back through the same set of routers traversed by the original PATH message. At each router along the path, the RESV message is processed and the reservation is incorporated into the router. When the RESV message reaches the sender, and end-to-end reservation is established.

The architecture of the Integrated Service is shown in Figure 1.1 (a). Figure 1.1 (b) shows the workflow of RSVP. Figure 1.1 (c) shows the architecture of a router in the Integrated Service domain. The upper part of Figure 1.1 (c) shows the functional blocks for processing RSVP messages, and the lower part of Figure 1.1 (c) shows the functional blocks for processing the actual data.

The Integrated Services model has a number of shortcomings. These include:

- The amount of state increases proportionally with the number of flows since routers need the information of the connection state and the QoS requirements for each flow. The state includes information to identify the flow, track the flow's resource consumption, police excess traffic beyond the reservation, and schedule the traffic
based on the reservation commitments. The core of the network could contain millions of reservations that need to be managed. Further, if a topology change occurs, all the reservations would need to be renegotiated simultaneously. This places a huge storage and processing overhead on the routers, which causes such architectures not to be scalable.

- The requirements imposed on routers are high. All routers must implement the RSVP, admission control, classification, and packet scheduling. This adds to the complexity of the routers and affects their throughputs.

- Ubiquitous deployment is required for guaranteed service. Incremental deployment of controlled-load service is possible by deploying controlled-load service and RSVP functionality at the bottleneck nodes of a domain and tunneling the RSVP messages over other parts of the domain.

- Resource must be reserved before data is transmitted. For short-lived flows, the processing overhead is greater than the processing of all the packets in the flow. The vast majority of Internet traffic consists of short-lived flows. In situations where some modest level of QoS is important to short-lived flows, the ISA is an overkill.

From the above, it is evident that the ISA does not meet all the requirements of high performance networks.
Figure 1.1 (a) Architecture of the Integrated Service
(b) Workflow of RSVP
(c) Functional blocks of IS routers
1.3 The Differentiated Services Architecture

The Differentiated Services Architecture (DiffServ) was introduced as an answer to ISA. The DiffServ philosophy is twofold: to provide services to aggregates of flows, rather than single flows, thus circumventing the scalability problem; and, to provide the guarantees on a per-hop basis, rather than on an end-to-end basis, hence reducing the processing requirements.

DiffServ uses different semantics for the type of service (TOS) byte in the IP header, and renames the TOS field as the DS (for DiffServ) field. One part of the DS field is a 6-bit field referred to as the differentiated services code point (DSCP), which is used to specify the service class. The remaining two bits of the DS field, referred to as currently unused (CU), are reserved for future use. As such, only a limited number of service classes are defined and service is allocated in the granularity of classes instead of single flows. The amount of state information is therefore proportional to the number of classes rather than to the number of flows. Therefore the Differentiated Service model is more scalable.

In order for a customer to receive a certain level of differentiated service from its network, the customer must establish a service level agreement (SLA) with its ISP. An SLA basically specifies the supported service classes and the amount of traffic allowed in each class. The DiffServ model can provide the following types of service [20]:

- Premium service. It provides low-delay and low-jitter service for customers (applications) that generate fixed peak bit rate traffic. Each customer will have an SLA with its ISP. The SLA specifies a desired peak bit-rate for a specific flow or an aggregation of flows. The contracted bandwidth is guaranteed to be available along
the path when the traffic is sent, and the excess traffic will be dropped. Premium packets are forwarded before packets of other classes.

- Assured service. It is intended for customers requiring better reliability than best-effort service, even in times of network congestion. The SLAs will specify the amount of bandwidth allocated for the customers. The assured service traffic that does not exceed the bit rate specified by the SLA is considered **in**, otherwise, it is considered **out**. **In** packets are dropped with lower probability than **out** packets when network congestion happens. Thus the customers will perceive a predictable service from the network. When there is no congestion, **out** packets will also be delivered.

The DiffServ model defines a basic set of packet forwarding treatments (per-hop behaviors, or PHBs), which support the above type of service. PHB is class-based, and specifies how the resource is allocated for each traffic class. In the DiffServ model, resources are allocated at the packet arrivals. No QoS requirements are exchanged between the source and the destination, and no resource reservation is needed, eliminating the inherent setup costs associated with RSVP.

The DiffServ model provides the following functions.

- Classifying. The DiffServ architecture requires the router to perform a number of functions to support any of the service categories described above. A router must be able to look at each packet and identify the aggregation to which it belongs.

- Metering. After a flow is classified, its resource consumption must be measured. To determine that the flow is not exceeding the agreed resource consumption limits
specified in its SLA, a router must measure a flow’s volume over some period of time and the size of its traffic bursts.

- **Shaping.** When a flow contains a burst of packets, a router can choose to process the burst in a number of ways. One alternative is to process it normally if it falls within some predefined or negotiated limit. Another alternative is to absorb the burst and pace the packets out over a longer period of time.

- **Dropping.** When a flow exceeds the negotiated rate or a burst exceeds a maximum threshold, a router may choose to drop one or more packets in the flow.

One of the main tenets of DiffServ is its distinction between the edge and the core of a DiffServ domain. Unlike ISA that performs classifications and policing functions on all packets matching a reservation in every router along the path, DiffServ pushes most of the classification and policing functions to the edges of the DiffServ domain, thus simplifying the forwarding functions in the core of the DiffServ domain. There are two types of routers in the DiffServ architecture, namely, edge routers and core routers. Edge routers are located at the ingress of the ISP networks. They classify, police, and possibly shape the arriving packets in a manner derived from the SLAs. The amount of buffering space needed for these operations is also derived from the SLAs. When a packet enters one domain from another domain, its DS field may change depending on the SLA between the two domains. Core routers are the routers within the ISP networks. They schedule and forward the packets based on their class types and the PHBs of the different classes. Hence, sophisticated classification, marking, policing, and shaping operations are only needed at the boundary of the networks. ISP core
routers need only have behavior aggregate (BA) classification, rather than flow-based specification. Therefore, it is easier to implement and deploy differentiated services.

The architecture of DiffServ is shown in Figure 1.2 (a). Figure 1.2 (b) shows the functional block of edge routers, and (c) shows the functional block of core routers.

DiffServ Domain

Figure 1.2 (a) Architecture of DiffServ
(b) Functional blocks of edge routers
(c) Functional blocks of core routers
1.4 Congestion Control and Queue Management

Network congestion is the state of sustained network overload where the demand for network resources, e.g., bandwidth, is close to or exceeds capacity. Figure 1.3 shows the relation between the amount of input traffic and goodput\(^1\) at a router. Before the amount of input traffic reaches the bandwidth of the output link of the router, the queue size at the router and the goodput increase as the amount of the input traffic increases. When the amount of input traffic is close to the output link capacity, depending on the different queue management techniques, some packets may be dropped. The sources therefore slow down their sending rate and re-transmit the dropped packets. Thus the increase of the goodput slows down, or even doesn’t increase. When the amount of input traffic exceeds the output link capacity, most of the incoming packets are dropped, and network congestion happens. The sources then decrease their sending rate, and re-transmit most of the packets, so most of the incoming traffic to the router is re-transmit traffic. Therefore the goodput decreases as the amount of input traffic increases.

\[ \text{Goodput} \]
\[ \text{Output Link Capacity} \]
\[ \text{Output Link Capacity} \]
\[ \text{Input} \]

Figure 1.3 Relation between the amount of input traffic and the goodput
Network congestion can prevent high performance networks from providing QoS guarantees to users. It can cause high packet loss rates and increased delays. Network congestion can be alleviated by introducing some level of interaction of the end-to-end congestion control and queue management. By using an appropriate queue management technique, routers can issue a signal to end users when congestion happens, and end users can therefore change their sending rate based on the signal.

There are two basic approaches for managing a queue at a router. Traditional Internet routers discard arriving packets if the buffer of the output port overflows. Different mechanisms, which belong to this category, include drop-tail, drop-front [18], and random drop [19]. These mechanisms suffer from a common problem as they allow the queue to remain full at most of time, which increases the queuing delay and prevent the network from providing guaranteed delay bound. Drop-tail may also cause a global synchronization problem. Contrary to Drop-Tail, active queue management (AQM) mechanisms [2] start randomly dropping packets on the onset of congestion and end nodes use these random packet drops as notifications of incipient congestion. We will discuss active queue management mechanisms in Chapter 2 in more details.

---

1 *Goodput* is the effective output. It equals the difference between the amount of traffic transferred and the amount of re-transmitted traffic in a time unit.

2 In Drop-Tail, multiple packets are dropped after the router buffer becomes full. If these packets belong to different TCP connections, these connections then experience losses at about the same time, decrease their sending rates in synchrony, and then tend to stay synchronized. Thus the output link of the router may become under utilization and the bandwidth is wasted.
1.5 Thesis Contribution and Outline

In this thesis, we propose a discrete time stochastic approach to analytically model the Random Early Detection *with in and out* (RIO) algorithm. We will also introduce a strategy to decide the maximum threshold in the Random Early Detection (RED) algorithm, which can be extended to other mechanisms.

The rest of the thesis is organized as follows. In Chapter 2, the active queue management techniques are introduced. Specifically two important techniques, RED and RIO, are described in detail. A literature review of the modeling approaches for active queue management is given. In Chapter 3 our strategy to decide the maximum threshold in the RED algorithm, and the analytical model for RIO are described. Chapter 4 evaluates our model by comparing the results from an OPNET simulator and the results from our model. Finally, Chapter 5 concludes the thesis and provides directions for future work.
Chapter 2 Active Queue Management

2.1 Introduction

One of the most important components in the DiffServ architecture is the queue management mechanism used at core routers. In addition to providing service guarantees, queue management also assists the congestion control function. In today’s internets, congestion control is performed mainly by the TCP transport protocols at end hosts, and in response to acknowledgements. While numerous studies have provided improvements to the TCP behavior, TCP connections may still suffer from high loss ratios in the presence of network congestion. On the other hand, as continuous media multicast applications (which usually do not employ TCP) become widely deployed on the Internet, it becomes difficult, if not impossible, to exclusively rely on end hosts to perform end-to-end congestion control. It has been widely agreed upon that the network itself must now participate in congestion control and resource management. The Internet Engineering Task Force (IETF) is advocating deployment of explicit congestion notification (ECN) and active queue management mechanisms at routers as a means of congestion control. By “active queue management”, it is meant that core routers inside networks are equipped with the capability to detect incipient congestion and to explicitly signal traffic sources before congestion actually occurs [4].

Active queue management mechanisms differ from the traditional drop tail mechanism in that in a drop-tail queue packets are dropped when the buffer overflows, i.e., when congestion actually happens, while with active queue management, packets may be dropped before congestion occurs. Several active queue management algorithms have been
introduced, e.g., Random Early Detection (RED) [2] and its variants, such as Fair RED (FRED) [16], stabilized RED (SRED) [21], and balanced RED (BRED) [22], and RED with in and out (RIO) [3]. These algorithms differ in several aspects, which include:

1. The parameter used as an index of traffic load (and congestion),
2. The policy used to detect congestion (or the likelihood of congestion), and
3. The policy used to adjust the packet dropping probability in response to (an increased likelihood of) congestion.

Because of the different policies used, queue management mechanisms also differ in terms of time complexity and scalability. In the following two sections, we summarize the two mechanisms, RED and RED with in and out, which have received significant consideration in the literature.

2.2 Introduction to the Random Early Detection (RED) Algorithm

An RED router operates as follows. It computes the average queue size, \( \text{avg} \_ \text{queue} \), upon packet arrival using a low-pass filter from the instantaneous queue size as follows:

\[
\text{avg} \_ \text{queue} = \begin{cases} 
(1 - w) \times \text{avg} \_ \text{queue} + w \times \text{queue} & \text{if } \text{queue} > 0 \\
(1 - w)^m \times \text{avg} \_ \text{queue} & \text{if } \text{queue} = 0 
\end{cases}
\]  

(2.1)

where \( \text{queue} \) is the instantaneous queue size at the arrival instant. \( w \) is a weigh factor. When the value of \( w \) is low, \( \text{avg} \_ \text{queue} \) is decided mainly by its history, and when the value of \( w \) is high, \( \text{avg} \_ \text{queue} \) is decided mainly by the instantaneous queue size. \( m \) represents the number of packets that might have been transmitted by the router during the period of time when the queue is empty. The value of \( m \) can be calculated using the following equation:
\[ m = \frac{(\text{time} - q_{\text{time}})}{T_x} \]  

(2.2)

where \text{time} is the time when the current packet arrives, \( q_{\text{time}} \) is the start of the queue idle time, and \( T_x \) is the packet transmission rate. The choice of \( m \) is based on the following consideration: The average queue size may have a high value at the last packet arrival. Since then, it may have been a long time before the current arriving packet finds that the queue is empty. Because the average queue size is only calculated upon the arrival of packets, it may not be updated for a long time and keeps a high value. If this is the case, the arriving packet may suffer a high dropping probability. However, since the queue is empty, the arriving packet should not be dropped. By considering the number of packets that might have been transmitted during the period of time when the queue is empty, the average queue size can be adjusted appropriately.

The parameter \text{avg\ queue} is used to measure traffic load. The policy used to detect the likelihood of congestion is characterized by two thresholds, \text{min} and \text{max}. There are three phases in RED, defined by the average queue size in the range of \([0, \text{min}), [\text{min}, \text{max}), \text{and} [\text{max}, \infty)\), and these correspond to normal operation, congestion avoidance, and congestion control, respectively. During the normal operation phase, when the average queue size is below \text{min}, the router does not drop any packets. When the average queue size is between the two thresholds, the router is operating in the congestion avoidance phase, and each packet drop serves the purpose of notifying the end-host transport layer to reduce its sending rate. Therefore, the dropping probability is a function of the average queue size and the maximum packet dropping probability \( P_{\text{max}} \), which is usually very small, and can be computed using equation \[ p = P_{\text{max}} \times \frac{\text{avg\ queue} - \text{min}}{\text{max} - \text{min}}. \] When the average queue size is above
max, the router drops every arriving packet, hoping to maintain a short queue size. The complete RED algorithm, shown in Figure 2.1 below, takes into account the variable count, the number of packets that have not been dropped when the average queue size is between min and max. However, in commercial implementations of RED algorithm, such as Cisco routers, count is not implemented. Therefore in the analytical model presented later in this thesis, it is also not taken into account.

```
Initialization:
    avg_queue = 0; count = -1;

For each packet arrival
    Calculate the average queue size avg_queue with equation (2.1)
    Determine packet discard
        if avg_queue < min
            Accept the packet
            count = -1
        else if min <= avg_queue <= max
            count = count + 1
            Calculate the dropping probability p
            With probability p
                Discard the packet
                count = 0
            else with probability 1-p
                accept the packet
        else if avg_queue > max
            Discard the packet
            count = 0

When queue becomes empty
    q_time = time
```

Figure 2.1 The RED algorithm

Being one of the earliest active queue management algorithms to be proposed, RED was shown to prevent global synchronization, accommodates bursty traffic, incurs little overheads, and coordinates well with TCP under serious congestion conditions.
The performance of RED, however, heavily depends on whether or not the two thresholds are properly selected. If the thresholds are set too small, buffer overflow is avoided but at the expense of link under utilization. On the other hand, if the thresholds are set too large, congestion occurs before end-hosts are notified to reduce their sending rates, and a large amount of bandwidth is wasted on transporting packets that will be eventually dropped. On heavily congested links, it may be difficult to keep both high link utilization and low packet loss ratio simultaneously. In Section 3.1, we propose a maximum threshold selection strategy.

2.3 Introduction to the RED with in and out (RIO) Algorithm

In the Differentiated Services architecture, each user subscribes to a certain level of services and is therefore provided with a service allocation profiles. However, the traffic users send may exceed their service allocation profiles. The core router should be able to process both the packets within the service allocation profile, and the packets outside the service allocation profile, albeit differently. To achieve this goal, an enhanced RED algorithm, RED with in and out (RIO), has been proposed. The core of the idea is simple — monitor the traffic of each user as it enters the network and tag packets as either in or out of their service allocation profiles, then at each congested router, preferentially drop packets that are tagged as being out.

The RIO algorithm operates as follows. It uses the same mechanism as in RED but is configured with two sets of parameters, one for in packets and one for out packets. Upon each packet arrival at the router, the router checks whether the packet is tagged as in or out. If it is an in packet, the router calculates $avg_{\text{in}}$, the average queue size for the in packets
only; if it is an *out* packet, the router calculates \( \text{avg } _{\text{total}} \), which is the average queue size for all (both *in* and *out*) arriving packets. The probability of dropping an *in* packet depends on \( \text{avg } _{\text{in}} \), and the probability of dropping an *out* packet depends on \( \text{avg } _{\text{total}} \). The algorithm is shown in Figure 2.1 below. The calculation of packet dropping probabilities for RIO is similar to RED, except that the thresholds of IN and OUT queues should be used correspondingly.

**Initialization:**
\[
\text{avg } _{\text{in}} = 0; \quad \text{avg } _{\text{out}} = 0;
\]

For each packet arrival
- If it is an *In* packet
  - Calculate the average *In* queue size \( \text{avg } _{\text{in}} \);
  - Calculate the average queue size \( \text{avg } _{\text{total}} \);
- If it is an *In* packet
  - if \( \text{avg } _{\text{in}} < \text{min } _{\text{in}} \)
    - Accept the packet
  - else if \( \text{min } _{\text{in}} \leq \text{avg } _{\text{in}} \leq \text{max } _{\text{in}} \)
    - Calculate dropping probability \( P_{\text{in}} \);
    - With \( P_{\text{in}} \), discard the packet;
  - else if \( \text{max } _{\text{in}} < \text{avg } _{\text{in}} \)
    - Discard the packet.
- If it is an *Out* packet
  - if \( \text{avg } _{\text{out}} < \text{min } _{\text{out}} \)
    - Accept the packet
  - else if \( \text{min } _{\text{out}} \leq \text{avg } _{\text{total}} \leq \text{max } _{\text{out}} \)
    - Calculate dropping probability \( P_{\text{out}} \);
    - With \( P_{\text{out}} \), discard the packet;
  - else if \( \text{max } _{\text{out}} < \text{avg } _{\text{total}} \)
    - Discard the packet.

**Figure 2.2 The RIO algorithm**

There are three parameters for each of the algorithms. The three parameters \( \text{min } _{\text{in}} \), \( \text{max } _{\text{in}} \), and \( P_{\text{max } _{\text{in}}} \) define the normal operation \([0, \text{min } _{\text{in}}]\), congestion avoidance
[min\_in, max\_in), and congestion control [max\_in, \infty) phases for in packets. Similarly, min\_out, max\_out, and \( P_{max\_out} \) define the corresponding phases for out packets.

The discrimination against out packets in RIO is created by carefully choosing the parameters (min\_in, max\_in, \( P_{max\_in} \)) and (min\_out, max\_out, \( P_{max\_out} \)).

2.4 Models of Active Queue Management Technique

To evaluate the performance of the active queue management techniques for Differentiated Services architecture, different approaches have been proposed. Among these approaches, simulation and measurements have been the major tools of choice for a long time. Only a few studies, such as [6], [8] and [9], have proposed analytical models to quantify the performance of different active queue management techniques. In this section, we summarize the recent studies.

In [6], the authors provided an analytical model for RED using Markov chain representation. However, they used the instantaneous queue size as an index of the traffic load instead of the average queue size, and took the instantaneous queue size as the only state variable of the Markov process. As is well known, in most cases, the average queue size is far lower than the instantaneous queue size. This model therefore lacks an accurate representation of RED. In addition, this model assumes that all the packets arriving at the router in the same traffic burst have the same dropping probability, which is not necessarily true.

In [8], a stochastic model for evaluating the performance of the RED algorithm was proposed using a two-dimensional second-order discrete-time Markov chain that captures the
feedback effect of packets dropping/marking on the incoming traffic. The authors concluded that the current instantaneous queue size depends on the instantaneous queue size in the previous two time slots. Hence the model uses the instantaneous queue size in two adjacent time slots as the state variables, and therefore has the same deficiency as the previous model. Also, the model assumed that all the packets arriving at the router are either all dropped or all accepted, which is not accurate.

In [9], the authors model RED as a feedback control system with TCP flows. In their model, the controlled systems are the TCP senders, and the controlling element is the drop module at the router. The feedback signal is the dropping probability, and the controlled variables are the TCP sending rates. They obtain a model of the average queue size as a function of the average packet drop probability. Combined with the RED dropping function in which the packet drop probability is a function of the average queue size, they obtain the steady state of this feedback system. This model is more appropriate for the analysis of the dynamic behavior of the RED system. However, in this model, the authors use the time-average queue size instead of the average queue size in RED, which also is not an accurate representation. Also they make the assumption that each flow should have the same configuration profile so that they can simplify the system of multiple flows to a single flow.

Based on the above introduction, it is clear that only a few analytical models for active queue management exist. It is also clear that all of them make one or more simplifying assumptions, which degrade the models accuracy. The most serious problem is that they use different approaches to work around the average queue size when they try to model the RED system, rather than using the average queue size exactly. For example, they may use the
instantaneous queue size, or the time-average queue size. This is due to the fact that modeling the average queue size exactly requires keeping track of several state variables, which assume large values, and thus makes the model intractable.

2.5 Summary

The chapter introduces active queue management. Random Early Detection (RED) algorithm is one of the active queue management techniques that have received significant consideration, and is introduced in detail. RED with in and out (RIO), an extension of RED, is also introduced.

To evaluate the performance of different active queue management techniques, several analytical models have been proposed. This chapter introduced some of them, and summarizes their shortcomings.

In the next chapter, we present an analytical model of RIO, which takes into account the average queue size as calculated by RED. Although the model makes a few simplifying assumptions, it will be shown by comparison to simulation that it is highly accurate.
In this chapter, we first develop a strategy for maximum threshold selection in the RED algorithm. It can be extended to the RIO algorithm. Then we present our analytical model for the RIO algorithm, which is based on a Markov chain model.

3.1 Maximum Threshold Calculation

As we discussed in Chapter 2, the performance of RED and RIO is highly dependent on the RED configuration, specifically the thresholds. Unfortunately, most of the studies propose RED configurations based on heuristics and simulations, and do not employ a systematic approach [2], [16]. The problem with these approaches is that they are only good for the particular traffic conditions studied, but may not be valid under different conditions. In this section, we study this problem and propose a strategy that may assist the network operator to decide the maximum threshold in RED. Using an extension of this, one can also evaluate the maximum threshold in RIO.

The maximum threshold should be determined based on the following criterion:

\[ \text{The average queue size should reach the maximum threshold when, or before the instantaneous queue size reaches the maximum buffer size.} \]

If the above is not satisfied, then if the buffer becomes full before the average queue size reaches the maximum threshold, RED will be downgraded to drop-tail, and the problem of TCP synchronization can happen.
Based on the above concept, we consider the worst case: the router experiences very heavy load continuously. We assume that the time is slotted and a time slot is equal to the packet transmission time (all the packets are assumed to be of the same fixed length). We also assume that during one time slot, the number of arriving packets is always $n$, which corresponds to the presence of $n$ sources and that the router can process exactly one packet after the $n^{th}$ arrival, if the queue is not empty. The average queue size is calculated at each packet arrival using equation (2.1) in Section 2.2.

In the following, $q^k_j$ and $a^l_j$ are used to denote the instantaneous queue size and the average queue size after the $j^{\text{th}}$ arriving packet in the $k^{\text{th}}$ time slot, respectively. Hence $q^k_0$ and $a^l_0$ are the instantaneous queue size and the average queue size at the beginning of the $k^{\text{th}}$ slot, respectively. The initial instantaneous queue size and the average queue size of the system are $q^1_0$ and $a^1_0$. We also assume that the maximum buffer size is $B$ packets and the maximum threshold we need to decide is $\text{max}$.

To simplify our analysis, we assume that no packets are dropped before the instantaneous queue size reaches the buffer size. This approximation is reasonable under the case we are considering, since the router experiences very heavy traffic load continuously, and the instantaneous queue size increases much faster than the average queue size. The average queue size is still small when the instantaneous queue size reaches the buffer size. So the packet drop event may be neglected. Also the maximum dropping probability is usually on the order of 0.01 or 0.02, which is very low, which means that most of the arriving packets are accepted before the buffer is full.
In the following we consider the evolution of the instantaneous queue size and the average queue size.

Time slot 1:

After the arrival of packet 1:

\[ q_1^1 = q_0^1 + 1, \]
\[ a_1^1 = (1 - w) \times a_0^1 + w \times q_0^1 \]

After the arrival of packet 2:

\[ q_2^1 = q_0^1 + 2, \]
\[ a_2^1 = (1 - w) \times a_1^1 + w \times q_1^1 \]

\[ \vdots \]

After the arrival of packet \( n \):

\[ q_n^1 = q_0^1 + n, \]
\[ a_n^1 = (1 - w) \times a_{n-1}^1 + w \times q_{n-1}^1 \]

From the above relations, we have:

\[ a_n^1 = (1 - w)^n \times a_0^1 + w \times (1 - w)^{n-1} \times q_0^1 + \ldots + w \times (1 - w) \times q_{n-2}^1 + w \times q_{n-1}^1 \]
\[ q_j^1 = j \]

which simplifies to:

\[ a_n^1 = (1 - w)^n \times a_0^1 + w \times (1 - w)^{n-1} \times q_0^1 + \ldots + w \times (1 - w) \times q_{n-2}^1 + w \times q_{n-1}^1 + \]
\[ (1 - w)^{n-2} \times w + \ldots + (1 - w) \times w \times (n - 2) + w \times (n - 1) \]
\[ = (1 - w)^n \times a_0^1 + \left[ 1 - (1 - w)^n \right] \times q_0^1 + w \times \sum_{i=1}^{n-1} i \times (1 - w)^{n-1-i} \]

Notice that, when \( w \times n \ll 1 \),

\[ a_n^1 = (1 - w)^n \times a_0^1 + \left[ 1 - (1 - w)^n \right] \times q_0^1 + n \times (n - 1) \times w \times \frac{1}{2}. \]
Similarly we can get the equation for $a_n^k$ and $q_n^k$ (for $k > 1$):

$$a_n^k = (1-w)^{n-1} \left[ (1-w) \times a_0^1 + w \times q_0^1 \right] + (1-w)^{n-2} \times w \times q_1^1 + ... + (1-w) \times w \times q_{n-2}^k + w \times q_{n-1}^k$$

$$q_n^k = q_0^j + (k-1) \times (n-1) + n$$

$$a_0^k = a_n^1$$

$$q_0^k = q_n^{k-1} - 1 = q_0^1 + (k-1) \times (n-1)$$

From above, we obtain $(k > 1)$:

$$a_n^k = (1-n \times w)^k \times a_0^1 + n \times w \times q_0^1 \times \sum_{i=0}^{k-1} (1-n \times w)^i + 0.5 \times n \times (n-1) \times w \times \sum_{i=0}^{k-1} (1-n \times w)^i + n \times (n-1) \times w \times \sum_{i=0}^{k-2} (1-n \times w)^i \times (k-1-i)$$

By solving the following equations, we may obtain the maximum threshold $\text{max}$:

$$q_n^k = B$$

$$a_n^k \leq \text{max}$$

From above, given a certain $n$ and $B$, $\text{max}$ is the function of the initial instantaneous queue size $q_0^1$ and the initial average queue size $a_0^1$. The relation between $\text{max}$, $q_0^1$ and $a_0^1$ is shown in Figure 3.1. Once the queue has been idle for a period of time, the maximum threshold can be changed based on this relation upon the arrival of a packet.
3.2 The Markov Chain Model Assumptions and Approximations

In [5], a discrete time stochastic model has been developed to model TCP Reno protocol with RED-based gateways. This paper models the RED-based router with a discrete time Markov chain. To extend this model to a more general case, e.g., RIO, the remainder of this chapter presents a discrete time stochastic model for evaluating the performance of the RIO algorithm using a two-dimensional, second-order Markov chain. Between embedding points, several discrete time intervals take place.

In our model, the following assumptions are made:
1. We consider the network topology shown in Figure 3.2. Router A is a bottlenecked router, which implements the RIO algorithm, and all the connections which go through this router have the same round trip time (RTT).

![Figure 3.2 The network topology](image)

2. A discrete time stochastic model is used for evaluating the performance of the RIO algorithm. Time is slotted and packets are of fixed length, such that the packet transmission time is exactly equal to one time slot.

3. The system is time synchronous. All packet arrivals and departures are synchronized to the slot boundaries.

4. The packet arrival process is a batch Poisson distribution. Multiple packets may arrive at the router during one time slot.
5. The packet drop decision is made on packet-by-packet basis instead of slot-by-slot basis, which means that the average queue size is computed for each arriving packet in a time slot, and is then used to decide whether to drop the packet or not. This is more accurate than the slot-by-slot dropping decision used in [8].

6. The average queue size for all the packets is assumed equal to the sum of the average queue size of IN packets and the average queue size of OUT packets, i.e.,

\[ a = a_{in} + a_{out}. \]

Although this is an approximation, it is accurate enough when the number of IN packets is much greater than that of OUT packets. This issue will be explained further in this section.

7. After the beginning of each time slot, if the buffer is not empty, one packet is processed. The probability of an IN packet being served is assumed proportional to the ratio of the number of IN packets to that of all the packets. Similarly the probability of an OUT packet being served is assumed proportional to the ratio of the number of OUT packets to that of all the packets. This uniform selection process is because of the memoryless arrival process.

Before illustrating the details of calculating the transition probability matrix, we define some variables as follows.

- \( B \): The buffer size of the router
- \( max_{in} \): The maximum threshold of the IN queue
- \( max_{out} \): The maximum threshold of the OUT queue
- \( min_{in} \): The minimum threshold of the IN queue
- \( min_{out} \): The minimum threshold of the OUT queue
$P_{\text{max}_{-}\text{in}}$: The maximum dropping probability for the IN queue

$P_{\text{max}_{-}\text{out}}$: The maximum dropping probability for the OUT queue

$q_{\text{in}}$: The instantaneous queue size of the IN queue

$q_{\text{out}}$: The instantaneous queue size of the OUT queue

$a_{\text{in}}$: The average queue size of the IN queue

$a_{\text{out}}$: The average queue size of the OUT queue

$\text{RTT}$: The round trip time between the source and the destination of a connection

$a_{\text{in}}$: The mean of the average queue size for the IN queue

$a_{\text{out}}$: The mean of the average queue size for the OUT queue

$q_{\text{in}}$: The mean of the instantaneous queue size for the IN queue

$q_{\text{out}}$: The mean of the instantaneous queue size for the OUT queue

$p(q_{\text{in}}, a_{\text{in}} | q_{\text{in}}, a_{\text{in}})$: The transition probability for the IN queue between two successive time slots from the state in which the instantaneous queue size is $q_{\text{in}}$ and the average queue size is $a_{\text{in}}$ to the state in which the instantaneous queue size is $q'_{\text{in}}$ and the average queue size is $a'_{\text{in}}$

$\text{drop}_{-}\text{in}(q_{\text{in}}, a_{\text{in}})$: The number of IN packets that are dropped with the instantaneous IN queue size being equal to $i$ and the average IN queue size being equal to $j$ within one $\text{RTT}$
3.3 Markov Model and Transition Probability Matrix Computation

Similar to [5], the system is modeled at both a macroscopic and a microscopic level. At the macroscopic level, which is the main model, the system is observed at the instants when a new round trip interval begins. That is, the system is modeled as an embedded Markov chain. The transition probabilities between two successive embedding points are then computed at the microscopic level, by taking the transitions between the systems at the slot boundaries between these two embedding points.

The system state at the embedding points (macroscopic model) consists of the tuple \( \Psi = (q_{in}, a_{in}, q, a) \), where \( q_{in} \) and \( a_{in} \) are the instantaneous queue size and the average queue size for the IN packets. We refer this model as "completed model". The variables \( q \) and \( a \) are
the instantaneous queue size and the average queue size for all the packets. And the values of these four variables satisfy:

\[ 0 \leq q_{in} \leq B \]

\[ 0 \leq q \leq B \]

\[ 0 \leq a_{in} \leq \text{max\_in} \]

\[ 0 \leq a \leq \text{max\_out} \]

Based on the above, the state space contains \([(B+1)\times(B+1)\times(\text{max\_in}+1)\times(\text{max\_out}+1)]\) states. When the values of \(B, \text{Max\_in}\) and \(\text{Max\_out}\) are high, the computational complexity is very high. To decrease the computational complexity, we need to decompose the system state. In most common situations, the number of the OUT packets is much smaller than that of the IN packets. We can always make \(\text{max\_out} < \text{max\_in}, \text{min\_out} < \text{min\_in}\) and \(P_{\text{max\_out}} > P_{\text{max\_in}}\). That is, OUT packets are dropped earlier than IN packets with higher probability, so both the instantaneous queue size and the average queue size for OUT packets should not be very high and should not have significant impact on the queue for IN packets. On the other hand, the instantaneous queue size and the average queue size for IN packets do have great impacts on the queue for the OUT packets. Therefore the OUT queue need not be taken into account when the transition probability matrix for the IN queue is constructed, while the effect of the IN queue on the OUT queue should be considered when constructing the transition probability matrix for the OUT queue.
Based on the above observations, the system state is decomposed into two tuples: $\Psi_1 = (q_{in}, a_{in})$ and $\Psi_2 = (q_{out}, a_{out})$, and the state transition probabilities are computed separately. We refer this model as “simplified model”. Here we define $q_{out}$ and $a_{out}$ to be the instantaneous queue size and the average queue size for the OUT packets, and they satisfy:

$$a_{out} = (1 - w) \times a_{out} + w \times q_{out}.$$ 

The transition probability matrix for the IN queue can be constructed independently and the mean of the average queue size $\tilde{a}_{in}$ and of the instantaneous queue size $\tilde{q}_{in}$ for the IN queue can be obtained by solving the matrix for the steady state probabilities of $a_{in}$ and $q_{in}$. It should be noted that the state variable $q_{in}$ can be any integer between 0 and $B$, and $a_{in}$ can be any integer between 0 and $\max_{in}$.

Similar to the above, the matrix for the OUT queue is constructed and the mean values of the average queue size, $\tilde{a}_{out}$, and the instantaneous queue size $\tilde{q}_{out}$ are obtained. The variable $q_{out}$ can be any integer between 0 and $B - q_{in}$, while $a_{out}$ can be any integer between 0 and $\max(0, \max_{out} - a_{in})$.

We now comment on the accuracy of this approximation. Since the computation of $a_{in}$ does not depend on OUT packets, the state variables $q_{in}$ and $a_{in}$ are sufficient to calculate $\tilde{a}_{in}$. Therefore, this approximation does not affect the accuracy of $\tilde{a}_{in}$. We just need to consider how this approximation affects the steady state of the queue for OUT packets.

We already have the following relations.
\[ a_{in} = (1 - w) \times a_{in} + w \times q_{in} \]

\[ a_{out} = (1 - w) \times a_{out} + w \times q_{out} \]

\[ a = (1 - w) \times a + w \times q \]

\[ q = q_{in} + q_{out} \]

If the linear relation \( a = a_{in} + a_{out} \) holds, then the state space of \( \Psi \) can be linearly decomposed into the two disjoint subspaces \( \Psi_1 \) and \( \Psi_2 \) without any loss of accuracies. We already know that \( a_{in} \) is computed when an IN packet arrives at the router, while \( a_{out} \) is computed when an OUT packet arrives. When a packet arrives, whether it is an IN packet or an OUT packet, \( a \) is computed. Regardless of the model that is used, the dropping decision for an IN packet is always based on \( a_{in} \). For an OUT packet, this is not the case. If we use an exact representation model, we need to make the dropping decision based on \( a \) as explained earlier. However, if we use the simplified model, we need to make the dropping decision based on \( a_{out} \). That is, the packet is accepted if enough buffer space exists; if \( a_{out} \geq \text{max}_{out} - a_{in} \), drop the packet; if \( \text{min}_{out} - a_{in} \leq a_{out} < \text{max}_{out} - a_{in} \), the packet is accepted with a certain probability. Let's consider two extreme cases: (i) all the packets are IN packets; and (ii) all the packets are OUT packets. In case (i), only \( a_{in} \) is computed and \( a_{out} \) is always 0. Therefore \( a = a_{in} + a_{out} \). In case (ii), only \( a_{out} \) is computed and \( a_{in} \) always equals to 0. Also \( a = a_{in} + a_{out} \) is satisfied. When the arriving packets are a mix of IN packets and OUT packets, the times when \( a_{out} \) is computed is less than the times when \( a \) is computed. So \( a_{out} \) will be increasing much slower than \( a \).
In this section, we show how to evaluate the transition probabilities. As mentioned above, we do this at two levels, the microscopic, i.e., the slot-by-slot level, and the macroscopic, which is between the start of two successive round trip times.

**Microscopic Analysis:**

First, we consider the transition between two successive slot boundaries within the embedding interval. Because the computation processes of the transition probabilities for $\Psi_1$ and $\Psi_2$ are similar, we will consider the computation for $\Psi_1$ first, then consider the difference between them.

We define the ordered pair $(q_{in}, a_{in})$ as the state at the beginning of a slot and the pair $(q^{i}_{in}, a^{i}_{in})$ as the state after the $i^{th}$ packet’s arrival within this slot, while the third pair $(q_{in}, a_{in})$ is the state at the end of a slot. The values of $q_{in}$, $q^{i}_{in}$ and $q_{in}$ are from 0 to $B$, and the values of $a_{in}$, $a^{i}_{in}$ and $a_{in}$ are from 0 to $\text{max}_{in}$. We refer the number of arriving IN packets within the $k^{th}$ slot as $n_{k}$, $1 \leq i \leq n_{k}$, and we have $q_{in}^{i} = q_{in}$, $a_{in}^{i} = a_{in}$, $q_{in}^{n_{k}} = q_{in}$, and $a_{in}^{n_{k}} = a_{in}$.

For each of these packets, we need to compute the new average queue size $a_{new}$ at the time the packet is processed by the router with equation (2.1) in Section 2.2. A special case occurs when $q_{in}^{i} = 0$, and we use the average number of packets that might have been transmitted during the period of time when the queue is empty to be equal to $m$ in equation
(2.1) as follows: suppose the number of slots within one RTT is $n_s$. If $\sum_{k=1}^{n_s} n_k > n_s$, we let $m = 1$; otherwise, we let $m = \left\lfloor \frac{n_s}{n_s} \right\rfloor$.

We assume that both the instantaneous queue size and the average queue size take integer values. However, since in practice, the average queue size can be any non-negative real number, after calculating the average queue size with equation (2.1) and getting a non-integer real number, it is rounded to the closest integers with a certain probability that is dependent on how close that actual average value to the integer approximation. For example, suppose after calculation, the average queue size equals 4.6, then it is rounded to 4 with probability 0.4 and to 5 with probability 0.6.

With the new average queue size and the threshold configurations, the state transition probabilities are calculated as follows, and according to different cases.

**Case 1:** $n_k = 0$, i.e., no IN packets arrive during the current time slot. The new average queue size is not calculated. We have the following relations:

- If $q_{in} = 0$,
  \[ p(q_{in}, a_{in} | q_{in}, a_{in}) = 1 \]

- If $q_{in} > 0$,
  \[ p(q_{in} - 1, a_{in} | q_{in}, a_{in}) = P_{serv} \]
  \[ p(q_{in}, a_{in} | q_{in}, a_{in}) = (1 - P_{serv}) \]
  \[ process\_in(q_{in}, a_{in}) = process\_in(q_{in}, a_{in}) + P_{serv} \]

where $P_{serv}$ is the probability that an IN packet will be processed.
So we define the transition probability between slot boundaries:

$$P = \left| p(q_{in}, a_{in} | q_{in}, a_{in}) \right|$$

whose elements are the transition probabilities from state \((q_{in}, a_{in})\) to state \((q_{in}, a_{in})\).

**Case 2:** \(n_k > 0\), i.e., the number of arriving IN packets is greater than zero. We need to compute the transition probabilities according to different situations. For the \(i^{th}\) IN packet \((1 \leq i \leq n_k)\), the computation process is as following:

If \(a_{in}^i < \min_{in}\), the arriving IN packet will be accepted as long as the queue is not full. So we calculate the transition probabilities as follows:

- If \(B - q_{in}^i > 0\), i.e., the queue is not full, then accept the arriving packet. After accepting the packet, the queue processes an IN packet with probability \(P_{serv} / n_k\) (\(P_{serv}\) is the probability with which the queue processes an IN packet within one time slot).

\[
p(q_{in}^i + 1 | a_{new}^i \geq q_{in}^i, a_{in}^i) = (1 - P_{serv} / n_k) \times (1 - \Delta)
\]

\[
p(q_{in}^i + 1 | a_{new}^i + 1 \geq q_{in}^i, a_{in}^i) = (1 - P_{serv} / n_k) \times \Delta
\]

\[
p(q_{in}^i | a_{new}^i \geq q_{in}^i, a_{in}^i) = (P_{serv} / n_k) \times (1 - \Delta)
\]

\[
p(q_{in}^i | a_{new}^i + 1 \geq q_{in}^i, a_{in}^i) = (P_{serv} / n_k) \times \Delta
\]

\[
\text{process } \_ \text{in}(q_{in}^i, a_{in}^i) = \text{process } \_ \text{in}(q_{in}^i, a_{in}^i) + P_{serv} / n_k
\]

Here, \(\lfloor a_{new}^i \rfloor\) is the maximum integer that is smaller than \(a_{new}^i\), and \(\Delta = a_{new} - \lfloor a_{new}^i \rfloor\).

- If \(B - q_{in}^i = 0\), i.e., the queue is full, the packet has to be dropped, and an IN packet is processed with probability \(P_{serv} / n_k\).

\[
drop \_ \text{in}(q_{in}^i, a_{in}^i) = drop \_ \text{in}(q_{in}^i, a_{in}^i) + 1
\]
If $a_{in}^i \geq \text{max}_\text{in}$, the arriving IN packet is dropped with probability 1. We therefore have the following relations:

- If $q_{in}^i = 0$, i.e., the queue is empty, it does not process any IN packet.
  
  \[
  \text{drop}_\text{in}(q_{in}^i, a_{in}^i) = \text{drop}_\text{in}(q_{in}^i, a_{in}^i) + 1
  \]
  
  \[
  p(q_{in}^i, a_{new}^i | q_{in}^i, a_{in}^i) = 1 - \Delta
  \]
  
  \[
  p(q_{in}^i, a_{new}^i + 1 | q_{in}^i, a_{in}^i) = \Delta
  \]

- If $q_{in}^i > 0$, i.e., the queue is not empty, it processes an IN packet with probability $P_{\text{serv}} / n_k$. Then, we have:

  \[
  \text{drop}_\text{in}(q_{in}^i, a_{in}^i) = \text{drop}_\text{in}(q_{in}^i, a_{in}^i) + 1
  \]
  
  \[
  p(q_{in}^i, a_{new}^i | q_{in}^i, a_{in}^i) = (1 - P_{\text{serv}} / n_k) \times (1 - \Delta)
  \]
  
  \[
  p(q_{in}^i, a_{new}^i + 1 | q_{in}^i, a_{in}^i) = (1 - P_{\text{serv}} / n_k) \times \Delta
  \]
  
  \[
  p(q_{in}^i - 1, a_{new}^i | q_{in}^i, a_{in}^i) = (P_{\text{serv}} / n_k) \times (1 - \Delta)
  \]
  
  \[
  p(q_{in}^i - 1, a_{new}^i + 1 | q_{in}^i, a_{in}^i) = (P_{\text{serv}} / n_k) \times \Delta
  \]
  
  \[
  \text{process}_\text{in}(q_{in}^i, a_{in}^i) = \text{process}_\text{in}(q_{in}^i, a_{in}^i) + P_{\text{serv}} / n_k
  \]
If \( \min_{in} \leq a_{in}^i < \max_{in} \), we need to compute the dropping probability \( P_{drop} \) using the following equation:

\[
P_{drop} = \frac{P_{\max_{in}} \times (a_{in}^i - \min_{in})}{(\max_{in} - \min_{in})}
\]

thus the probability of accepting the arriving packet is \( 1 - P_{drop} \). Then we have the following relations:

- If \( B - q_{in}^i = 0 \), i.e., the queue is full, the arriving IN packet is dropped with probability 1. Since \( q_{in}^i = B > 0 \), the queue processes an IN packet with probability \( P_{serv} / n_k \). So we have:

  \[
drop_{in}(q_{in}^i, a_{in}^i) = drop_{in}(q_{in}^i, a_{in}^i) + 1
\]

  \[
p(q_{in}^i, a_{new}^i | q_{in}^i, a_{in}^i) = (1 - P_{serv} / n_k) \times (1 - \Delta)
\]

  \[
p(q_{in}^i, a_{new}^i + 1 | q_{in}^i, a_{in}^i) = (1 - P_{serv} / n_k) \times \Delta
\]

  \[
p(q_{in}^i - 1, a_{new}^i | q_{in}^i, a_{in}^i) = (P_{serv} / n_k) \times (1 - \Delta)
\]

  \[
p(q_{in}^i - 1, a_{new}^i + 1 | q_{in}^i, a_{in}^i) = (P_{serv} / n_k) \times \Delta
\]

  \[
process_{in}(q_{in}^i, a_{in}^i) = process_{in}(q_{in}^i, a_{in}^i) + P_{serv} / n_k
\]

- If \( B - q_{in}^i > 0 \), we may accept \( k(k = 0, 1) \) packets,
  
  - If \( q_{in}^i + k = 0 \), the queue does not process any IN packet. So we have:

    \[
drop_{in}(q_{in}^i, a_{in}^i) = drop(q_{in}^i, a_{in}^i) + (1 - k) \times P_{accept} \times (1 - P_{accept})^{1-k}
\]

    \[
p(q_{in}^i + k, a_{new}^i | q_{in}^i, a_{in}^i) = (1 - \Delta) \times P_{accept} \times (1 - P_{accept})^{1-k}
\]

    \[
p(q_{in}^i + k, a_{new}^i + 1 | q_{in}^i, a_{in}^i) = \Delta \times P_{accept} \times (1 - P_{accept})^{1-k}
\]

  - If \( q_{in}^i + k > 0 \), the queue processes one IN packet with probability \( P_{serv} / n_k \).
\[\text{drop} \_\text{in}(q_{in}^i, a_{in}^i) = \text{drop}(q_{in}^i, a_{in}^i) + (1 - k) \times P_{\text{accept}}^k \times (1 - P_{\text{accept}})^{1-k}\]

\[p(q_{in}^i + k, a_{new}^i \mid q_{in}^i, a_{in}^i) = (1 - P_{\text{serv}} / n_k) \times (1 - \Delta) \times P_{\text{accept}}^k \times (1 - P_{\text{accept}})^{1-k}\]

\[p(q_{in}^i + k, a_{new}^i + 1 \mid q_{in}^i, a_{in}^i) = (1 - P_{\text{serv}} / n_k) \times \Delta \times P_{\text{accept}}^k \times (1 - P_{\text{accept}})^{1-k}\]

\[p(q_{in}^i + k - 1, a_{new}^i \mid q_{in}^i, a_{in}^i) = (P_{\text{serv}} / n_k) \times (1 - \Delta) \times P_{\text{accept}}^k \times (1 - P_{\text{accept}})^{1-k}\]

\[p(q_{in}^i + k - 1, a_{new}^i + 1 \mid q_{in}^i, a_{in}^i) = (P_{\text{serv}} / n_k) \times \Delta \times P_{\text{accept}}^k \times (1 - P_{\text{accept}})^{1-k}\]

\[\text{process} \_\text{in}(q_{in}^i, a_{in}^i) = \text{process} \_\text{in}(q_{in}^i, a_{in}^i) + (P_{\text{serv}} / n_k) \times P_{\text{accept}}^k \times (1 - P_{\text{accept}})^{1-k}\]

Based on the above, we define the transition probability matrix between packet arrivals within the same slot:

\[P_i = \left[p(q_{in}^{i+1}, a_{in}^{i+1} \mid q_{in}^i, a_{in}^i)\right]\]

whose elements are the transition probabilities from state \((q_{in}^i, a_{in}^i)\) to state \((q_{in}^{i+1}, a_{in}^{i+1})\) for \(i = 1, \ldots, n_k - 1\).

For each IN packet within the \(k^{th}\) slot, we repeat the above process. Thus the transition probability matrix between the slot boundaries is:

\[P = \left[p(q_{in}^i, a_{in}^i \mid q_{in}^i, a_{in}^i)\right] = \prod_{i=1}^{n_k} P_i\]

Macroscopic Analysis:

Now we consider an embedding interval that starts at the beginning of a round trip interval. We define the transition probability from state \((q_{in}^i, a_{in}^i)\) to state \((q_{in}^i, a_{in}^i)\) as

\[p(q_{in}^i, a_{in}^i \mid q_{in}^i, a_{in}^i)\]. The transition probability matrix is denoted by \(R\) and

\[R = \left[p(q_{in}^i, a_{in}^i \mid q_{in}^i, a_{in}^i)\right].\]
41

\( R \) can be calculated using the equation \( R = P^n \).

Considering the similarity between the computation of IN queue and OUT queue, we may use the similar computation process for OUT queue as used for IN queue to construct the transition probability matrix with the following exceptions:

- The value range of \( q_{out} \) is from 0 to \( B - q_{in} \) and the value range of \( a_{out} \) is from 0 to \( \max_{out} - a_{in} \).

- If \( \max_{out} - a_{in} < 0 \), then let \( a_{out} = 0 \).

3.4 Performance Measures

Denote the steady state probability vector for IN queue by

\[ \Pi_{in} = \{\pi_{in,1}, \pi_{in,2}, \ldots, \pi_{in,B} \} \]

where \( \pi_{in,i} \) is the steady state probability vector of the IN queue having the instantaneous queue size \( i \). This vector in turn has several elements,

\[ \pi_{in,i} = \{\pi_{in,i,0}, \pi_{in,i,1}, \ldots, \pi_{in,i,max_{in}} \} \]

Similarly the steady state probability vector for OUT queue can be denoted by

\[ \Pi_{out} = \{\pi_{out,1}, \pi_{out,2}, \ldots, \pi_{out,\Delta} \} \]

where \( \Delta = B - q_{in} \), and \( \pi_{out,i} \) has the following elements:

If \( a_{in} \leq \max_{out} \), let \( \Delta = \max_{out} - a_{in} \). Then,
If $a_{in} > \text{max}_\text{out}$, then

$$\pi_{out,i,j} = \begin{cases} 0 & (i \neq 0 \text{ or } j \neq 0) \\ 1 & (i = 0 \text{ and } j = 0) \end{cases}$$

After the vector $\Pi_{in}$ has been obtained by using the equation $\Pi_{in} \times R = \Pi_{in}$, and the condition that the sum of all probability elements equals to 1, i.e., $\sum_{i=0}^{B_{\text{max}_\text{in}}} \sum_{j=0}^{B} \pi_{in,i,j} = 1$, several performance measures can be obtained. These include:

1. The mean value of the average queue size $a_{in}$:

$$a_{in} = \sum_{j=0}^{B_{\text{max}_\text{in}}} \sum_{i=0}^{B} j \times \pi_{in,i,j}$$

2. The mean value of the instantaneous queue size $q_{in}$:

$$q_{in} = \sum_{i=0}^{B_{\text{max}_\text{in}}} \sum_{j=0}^{B} i \times \pi_{in,i,j}$$

3. The average dropping probability for IN packets $\text{drop}_\text{in}$:

$$\text{drop}_\text{in} = \left( \sum_{i=0}^{B_{\text{max}_\text{in}}} \sum_{j=0}^{B} \text{drop}_\text{in}(i,j) \times \pi_{in,i,j} \right) / \sum_{k=1}^{n} n_k$$

Using the similar approach, we can obtain $a_{out}$, $q_{out}$ and $\text{drop}_\text{out}$.

4. Throughput: The system throughput can be obtained using the following definition:

$$\text{Throughput} = E \left( \frac{\text{number of packets processed in one RTT}}{\text{RTT}} \right)$$

$$= E \left( \frac{\text{number of IN packets processed in one RTT}}{\text{RTT}} \right) +$$
E (number of OUT packets processed in one RTT) / RTT

E (number of IN packets processed in one RTT) can be obtained using the following equation:

\[
E \text{ (number of IN packets processed in one RTT)} = \sum_{i=0}^{b_{\text{max, in}}} \sum_{j=0}^{\text{process _in}(i, j) \times \pi_{\text{in}, i, j}}
\]

E (number of OUT packets processed in one RTT) can be obtained similarly.

3.5 Summary

This chapter has presented two contributions. First, Section 3.1, introduces a strategy for the selection of the maximum threshold selection in the RED algorithm, which can be extended to RIO and other active queue management techniques using thresholds. Secondly, Sections 3.2 and 3.3, proposed an analytical model for RIO algorithm. The model is based on the discrete time Markov chain. Two levels of Markov process modeling are used: a microscopic level, at the packet transmission time boundaries, and a macroscopic level, at the start of the new round trip time. A few performance measures were obtained in Section 3.4.
CHAPTER 4 SIMULATION RESULTS

In this chapter, performance results from the models of the thesis are compared to simulation results to verify the accuracy of the model. In Section 4.1, we will compare the results from an OPNET simulator to the calculation of the maximum threshold in RED algorithm based on our strategy. In Section 4.2 we present the network model used to assess the accuracy of our model for RIO. In Section 4.3 we evaluate the effect of the thresholds on the performance of RIO. In Section 4.4 we study the effect of the ratio of out of profile packets, namely OUT packets, on the performance of RIO. In Section 4.5 we summarize the above results.

4.1 Evaluation of The Maximum Threshold of RED

In this section, we present numerical results based on the maximum threshold selection strategy introduced in Section 3.1, and compare it to the results from an OPNET simulator. From Section 3.1, we derived the following relations:

\[
q^k_n = q_0^1 + (k - 1) 	imes (n - 1) + n
\]

\[
a^k_n = (1 - n \times w)^k \times a_0^1 + n \times w \times q_0^1 \times \sum_{i=0}^{k-1} (1 - n \times w)^i + 0.5 \times n \times (n - 1) \times w \times \sum_{i=0}^{k-1} (1 - n \times w)^i \times (k - 1 - i)
\]

\[
a_0^1 = (1 - w)^n \times a_0^1 + [1 - (1 - w)^n] \times q_0^1 + n \times (n - 1) \times w / 2
\]

By solving the following equations, we can derive the maximum threshold in RED:
As an example, if $B = 50$, $n = 2$, $w = 0.002$, $q_0^1 = 0$ and $a_0^1 = 0$, we have:

$$a_1^1 = n \times (n-1) \times w / 2 = 2 \times 1 \times 0.002 / 2 = 0.002$$

From $q_k^i \leq B$, the following inequality must hold:

$$q_k^i = (k-1) \times (n-1) + n = (k-1) \times (2-1) + 2 \leq 50,$$

which yields $k \leq 49$. By choosing $k = 49$, we have:

$$a_n^1 = 0.5 \times n \times (n-1) \times w \times \sum_{i=0}^{k-2} (1-n \times w)^i + n \times (n-1) \times w \times \sum_{i=1}^{k-2} (1-n \times w)^i \times (k-1-i) = 4.5115$$

To guarantee that the average queue size reaches the maximum threshold before the instantaneous queue size reaches the buffer size under all traffic conditions, we choose $\max = 4$. To verify our computation, we conducted a few experiments with OPNET simulator. The network topology is shown in Figure 4.1. The link between routers A and B is the only bottleneck in the network. RED algorithm is implemented in router A.

![Figure 4.1 The network topology in RED simulations](image)

In our simulation, packet size is set to 1500 bytes. The bandwidth of the link between the source and router A is 10 Mbps, and the bandwidth of the link between routers A and B is 3 Mbps. The source is configured to send 500 packets per second. Therefore, the incoming
traffic rate to router A is $500 \times (1500 \times 8) = 6 \times 10^6$ bps, and the router A's processing rate is $3 \times 10^6$ bps. The ratio between the incoming traffic rate and the processing rate is therefore $2$. The buffer size at router A is 50 packets. We also set the minimum threshold to zero. Figure 4.2, 4.3 and 4.4 show the instantaneous queue size obtained from the OPNET simulator with the maximum threshold equal to 30, 5, and 4, respectively. We find that when the maximum threshold is greater than 4, for example, 30 and 5, there is a chance that the instantaneous queue size exceeds the buffer size. This likelihood increases with the increasing maximum threshold. When the maximum threshold decreases to 4 or less, the instantaneous queue size is always below the buffer size, and no packets are ever dropped due to buffer overflow. This conforms to the strategy in Section 3.1. We also find that when the maximum threshold is greater than that evaluated by our strategy, the instantaneous queue size may exceed the buffer size only at the very beginning of the simulation. This is because in this region, the average queue size is very small and no packets are dropped, and the instantaneous queue size can therefore increase quickly. This is the possible worst case, which is considered in our strategy. As the simulation progresses, the average queue size increases and packets start to be dropped, and the instantaneous queue size starts to decrease. This worst case may happen when a large amount of traffic arrives at the router after the router has been empty for some time. This is possible due to the bursty characteristic of traffic generated by the applications which use TCP, such as HTTP, FTP, etc.
Figure 4.2 The instantaneous queue size (max=30)

Figure 4.3 The instantaneous queue size (max=5)
4.2 Network Model

We consider a simple network topology with ten sources connecting to their respective destinations via one common link. The topology is shown in Figure 4.5. The link between A and B is the only bottleneck in the network. RIO algorithm is implemented at router A. All the packets are 1500 bytes. TCP connection $i$ is made from source $i$ to destination $i$ ($1 \leq i \leq 10$). The round trip time for all the links is 20ms. All the sources are configured to send traffic based on Poisson distribution with different average rate. The buffer size of router A is 50 packets. The transmission rate of the links between sources and router A and the links between router B and destinations is 10 M bps. The transmission rate of the link between router A and B is 33 Mbps.
In our experiment, the rate at which packets arrive at the router A is denoted by $r_s$ (packet/sec), and the service rate at which the router A processes the arriving packets is $r_o$ (packet/sec). We can therefore define the offered load $r$ to router A as the ratio between $r_s$ and $r_o$. We also define ratio as the ratio between the number of OUT packets, and the number of all the packets within one round trip time. In our experiment, we control the traffic sent by the sources to provide different offered load levels to the router A, and change the parameters used in RIO. We consider the effect of different parameters, such as the thresholds, the offered load, and the maximum dropping probabilities. By comparing the results obtained from OPNET and our model under different parameters, we can assess the accuracy of our model.
4.3 Effect of Threshold Values on RIO Performance

In this section, we study the effect of the thresholds in RIO, \( Max_{in} \), \( Min_{in} \), \( Max_{out} \) and \( Min_{out} \), on the performance, and compare results from our model to results from an OPNET simulator. The following combinations for the thresholds were selected.

1. \( max_{in} = 8, min_{in} = 4, max_{out} = 4, min_{out} = 2 \)
2. \( max_{in} = 20, min_{in} = 10, max_{out} = 10, min_{out} = 5 \)
3. \( max_{in} = 30, min_{in} = 15, max_{out} = 15, min_{out} = 7 \)

Each combination is tested under different offered load levels from 0.1 to 2, and different values of ratio. In this experiment we used, \( P_{max_{in}} = 0.1 \) and \( P_{max_{out}} = 0.5 \). The results are shown in Figure 4.6 - 4.17.

In these figures, the average queue sizes, \( a_{in} \) and \( a_{out} \), are shown versus the offered load. The results from our model and the results from the OPNET simulator with the same parameters are plotted on the same figure for the sake of comparison. As expected, when the thresholds increase, \( a_{in} \) and \( a_{out} \) also increase. We also notice that as the offered load increases, \( a_{in} \) also increases, while \( a_{out} \) first increases then decreases. This can be explained as follows: with low offered load, the queue has enough space for the arriving packets, so both \( a_{in} \) and \( a_{out} \) increase as the offered load increases. However, when the offered load exceeds a certain value, the space in the queue is limited, and the RIO algorithm favors IN packets, i.e., it always drops OUT packets before IN packets when it has to drop packets, and
therefore $a_{in}$ keeps increasing while $a_{out}$ begins to decrease. Our model captures this behavior accurately.

4.4 Effect of Traffic Mix

Figure 4.6 - 4.17 show the effect of the traffic mix on the performance. This can be done by controlling the parameter ratio. When the ratio increases, i.e., the proportion of OUT packets increases, the average queue size for IN packets decreases under the same offered load, and the average queue size for OUT packets increases. We notice that the lower the value of ratio, the closer are the results for the IN queue from our model to the results from OPNET. For example, with ratio=0.1, the results based on our model are very close to those from the simulation. With ratio=0.5, the difference between the results based on our model and those from the simulation increases to around 15%. This is because in most common cases the number of OUT packets is far less than that of IN packets, therefore when we construct the transition probability matrix for the IN queue, we don’t take the number of OUT packets into account. The value of the instantaneous queue size for the IN queue ranges from 0 to $B$ instead of $B - a_{out}$, and the value of the average queue size for the IN queue ranges from 0 to max\_in instead of max(0, max\_in – $a_{out}$). So the values of the average queue size for the IN queue from the model is always greater than those from the simulator. When the ratio increases, the number of OUT packets increases as well, and the effect of OUT packets becomes more important. However, since the number of OUT packets is usually not high, our model is accurate in most common cases.
4.5 Effect of the Maximum Dropping Probability

In this section, we study the effect of the maximum dropping probabilities $P_{\text{max}_{\text{in}}}$ and $P_{\text{max}_{\text{out}}}$ in RIO. We selected the following combinations for our experiment.

1. $\text{max}_{\text{in}} = 8, \text{min}_{\text{in}} = 4, \text{max}_{\text{out}} = 4, \text{min}_{\text{out}} = 2, P_{\text{max}_{\text{in}}} = 0.1, P_{\text{max}_{\text{out}}} = 0.5$

2. $\text{max}_{\text{in}} = 8, \text{min}_{\text{in}} = 4, \text{max}_{\text{out}} = 4, \text{min}_{\text{out}} = 2, P_{\text{max}_{\text{in}}} = 0.8, P_{\text{max}_{\text{out}}} = 0.9$

Each combination is tested with the offered load from 0.1 to 2 and ratio equal to 0.1. The results from our model and from the OPNET simulator are shown in Figure 4.18. From the figure, we find that when the maximum dropping probabilities increase, the average queue sizes decrease as expected. When $P_{\text{max}_{\text{in}}} = 0.1$, only a small number of IN packets are dropped when the average queue size for IN packets is between $\text{min}_{\text{in}}$ and $\text{max}_{\text{in}}$. Therefore the mean of the average queue size for IN packets almost reaches the buffer size. When $P_{\text{max}_{\text{in}}} = 0.8$, on the average, about 40% IN packets are dropped when the average queue size is between $\text{min}_{\text{in}}$ and $\text{max}_{\text{in}}$. So the mean of the average queue size for IN packets is slightly greater than 6, which is the middle point between $\text{min}_{\text{in}}$ and $\text{max}_{\text{in}}$.

We also notice that the effect of the maximum dropping probabilities under low offered load is not as obvious as that under high offered load. This is because the maximum dropping probabilities affect the average queue sizes only when the average queue sizes exceed the corresponding minimum threshold. For example, when the offered load is so low that $\tilde{a}_{\text{in}}$ is lower than $\text{min}_{\text{in}}$, $\tilde{a}_{\text{in}}$ is almost not affected by $P_{\text{max}_{\text{in}}}$. 
4.6 Summary

In this chapter, we gave an example for the maximum threshold selection strategy for RED algorithm, which was developed in Chapter 2. We also showed the results from the OPNET simulator. The simulation results verify our strategy.

We then presented the results from the model and from the OPNET simulator to show the effect of different parameters, such as thresholds, the ratio, and the dropping probabilities, on the RIO performance and to assess the accuracy of the model. The results show that our model can capture the effect of the parameters on the RIO performance fairly well.
Figure 4.6 Average queue sizes for IN and OUT queues (max_in=8, min_in=4, max_out=4, min_out=2, Pmax_in=0.1, Pmax_out=0.5, ratio=0.1)

Figure 4.7 Average queue sizes for IN and OUT queues (max_in=8, min_in=4, max_out=4, min_out=2, Pmax_in=0.1, Pmax_out=0.5, ratio=0.3)
Figure 4.8 Average queue sizes for IN and OUT queues (max_in=8, min_in=4, max_out=4, min_out=2, Pmax_in=0.1, Pmax_out=0.5, ratio=0.5)

Figure 4.9 Average queue sizes for IN and OUT queues (max_in=8, min_in=4, max_out=4, min_out=2, Pmax_in=0.1, Pmax_out=0.5, ratio=0.9)
Figure 4.10 Average queue sizes for IN and OUT queues (max_in=20, min_in=10, max_out=10, min_out=5, Pmax_in=0.1, Pmax_out=0.5, ratio=0.1)

Figure 4.11 Average queue sizes for IN and OUT queues (max_in=20, min_in=10, max_out=10, min_out=5, Pmax_in=0.1, Pmax_out=0.5, ratio=0.3)
Figure 4.12 Average queue sizes for IN and OUT queues (max_in=20, min_in=10, max_out=10, min_out=5, Pmax_in=0.1, Pmax_out=0.5, ratio=0.5)

Figure 4.13 Average queue sizes for IN and OUT queue (max_in=20, min_in=10, max_out=10, min_out=5, Pmax_in=0.1, Pmax_out=0.5, ratio=0.9)
Figure 4.14 Average queue sizes for IN and OUT queues (max_in=30, min_in=15, max_out=15, min_out=7, Pmax_in=0.1, Pmax_out=0.5, ratio=0.1)

Figure 4.15 Average queue sizes for IN and OUT queues (max_in=30, min_in=15, max_out=15, min_out=7, Pmax_in=0.1, Pmax_out=0.5, ratio=0.3)
Figure 4.16 Average queue sizes for IN and OUT queue (max_in=30, min_in=15, max_out=15, min_out=7, Pmax_in=0.1, Pmax_out=0.5, ratio=0.5)

Figure 4.17 Average queue sizes for IN and OUT queue (max_in=30, min_in=15, max_out=15, min_out=7, Pmax_in=0.1, Pmax_out=0.5, ratio=0.9)
Figure 4.18 Average queue sizes for IN and OUT queues under different dropping probabilities (max_in=8, min_in=4, max_out=4, min_out=2, ratio=0.1).
(i) P_{max\_in}=0.1, P_{max\_out}=0.5  (ii) P_{max\_in}=0.8, P_{max\_out}=0.9)
CHAPTER 5  SUMMARY AND CONCLUSIONS

Since active queue management is a key component in the Differentiated Services architecture, analytical approaches to evaluate the performance of different active queue management techniques and the effects of their parameters are needed. Only a few studies have proposed such models. However, due to the complexity of these techniques, these models have made several simplifying assumptions and are inaccurate in many cases. This thesis has considered RIO which is an important active queue management technique. No analytical model has been proposed for RIO and its performance is therefore not clear.

5.1 Thesis Contributions

In this thesis, we propose an analytical model to evaluate the performance of RIO algorithm. We modeled the RIO algorithm using a discrete time Markov chain. By analyzing the properties of the RIO algorithm, we decomposed the state vector into two sub-vectors, thus simplifying the computation. In our model, we considered the average queue size as one of the state variables, which is more accurate than previous studies, which modeled RED.

We also proposed a strategy to decide the maximum threshold for the RED algorithm. This strategy is helpful when the router experiences very heavy traffic after it has been empty for a period of time. We considered the cases in which the heavy traffic starts when the instantaneous and average queue sizes are at certain levels. This strategy can be used to dynamically adjust the maximum threshold such that the maximum threshold is always
reached when the instantaneous queue size reaches, or is just about to reach the maximum buffer size.

The RIO algorithm has also been implemented in the OPNET simulator and compared results from the simulator to results from the proposed analytical model. The model results compared to the simulation results. The model was used to study the effect of several parameters on the performance.

5.2 Future Directions

The following are some of the future research directions in the context of analytical modeling for the active queue management techniques.

Currently we only consider the model of the router only. To describe the algorithms in more details, we need to incorporate the TCP sources with our model, and use the feedback mechanism to control the sources. TCP sources may be also modeled based on Markov chain. The traffic generated by the sources is fed into the model of the thesis. Then the dropping probability obtained from our model can be used to control the TCP sources.

In this thesis, we simplify the system state, and ignore OUT packets when the transition probability matrix for IN queue is constructed. To make our model suitable for the situation where the number of OUT packets is non-negligible, parameters for the OUT queue state are needed when the transition probability matrix for the IN queue is constructed.

In RIO, only two classes of traffic are considered. In other algorithms, more than two classes may be used. The model of this thesis can be extended straightforwardly to include multiple classes. However, the complexity of the model will increase significantly. It is
therefore required that the extension to multiple classes be done in a manner that results in a moderate state space increase.
REFERENCES


