END-TO-END SINGLE-RATE MULTICAST CONGESTION DETECTION USING SUPPORT VECTOR MACHINES

By

Xiaoming Liu

A Thesis Submitted in fulfilment of

the Requirements for the Degree of

MAGISTER SCIENTIAE

in the Department of Computer Science,

University of the Western Cpae.

Dr. Christian Omlin
Thesis Supervisor

University of the Western Cape
Bellville, Cape Town

June 2007
(For Graduation March 2008)

Keywords

Multicast

Congestion Detection

Machine Learning

Accumulation Measurement

Support Vector Machines

Declaration

I, Xiaoming Liu, declare that this thesis "*End-to-End Single-rate Multicast Congestion Detection Using Support Vector Machines*" is my own work, that it has not been submitted before for any degree or assessment in any other university, and that all the sources I have used or quoted have been indicated and acknowledged by means of complete references.

Signature: ..... Date: .....

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENT

I, as well as this thesis, am a compilation of efforts from many people to help develop and guide me through the years. I would first like to thank my advisor Prof. Christian Omlin for supporting and encouraging me during my Master's study. Without his help, this work would not have been possible. At this time I would like to extend a very special thanks to Prof. IM. Venter, James Connan and Verna Connan. Without their help I would certainly not be where I am today. Thank you.

I would like to thank my colleagues Yaw Nkansah-Gyekye and Eihab Bashier Mohammed Bashier, who gave me a lot of valuable suggestion for my project. They were more than just colleagues, but they were dear friends whom I will always cherish. Support does not always come in the form of time in front of a machine; many of my friends that I had no opportunity to work with also played a part in my success, including Shan Ji, Lixiang Lin, Haili Jia, Hongze Luo, Changhong Wu and Long Yi.

I would like to thank Ying Ma, a friend who give me a lot of support.

Finally, I give respect and love to my family. Although they were thousands of miles away from me, I always felt their presence in my heart.

ABSTRACT

IP multicast is an efficient mechanism for simultaneously transmitting bulk data to multiple receivers. Many applications can benefit from multicast, such as audio and videoconferencing, multi-player games, multimedia broadcasting, distance education, and data replication. For either technical or policy reasons, IP multicast still has not yet been deployed in today's Internet. Congestion is one of the most important issues impeding the development and deployment of IP multicast and multicast applications. Many congestion control schemes have been proposed to tackle multicast congestion problem. However, few of the schemes focus on using machine learning to detect multicast congestion in advance.

Machine learning has already been successfully applied in a number of areas without much background information, and gives useful results. Because we tackle the multicast congestion problem with the end-to-end assumptions, we cannot obtain opportune and accurate congestion information directly from inside the network. Therefore, machine learning is particularly appropriate due to the absence of congestion information and the unpredictable variance of network congestion. To detect end-to-end multicast congestion, we propose an end-to-end multicast congestion detection scheme using support vector machines. Support vector machines are able to detect incipient congestion with great accuracy in an end-to-end multicast network after training by using structural information about the multicast network.

To verify the performance of our scheme, we ran several `ns-2` simulations and statistical experiments. Our simulations have shown that support vector machine is an appropriate mechanism for decision making in proactive multicast congestion detection.

CHAPTER 1
Introduction

1.1 Motivation

Since IP multicast was introduced by Steve Deering in 1989 in RFC 1112 [30], many solutions have been proposed for multicast congestion management. Two distinct strategies of congestion management have been suggested to tackle the multicast congestion problem: *congestion avoidance* [21] schemes detect and respond to network congestion without necessarily inducing packet loss, while *congestion control* schemes only detect congestion after packet loss occurs. Consequently, congestion avoidance mechanisms that detected incipient congestion may be more efficient than congestion control. In addition, we study multicast congestion management issues based on end-to-end assumption since only an end-to-end congestion management frame can be used in the Internet, i.e., we can only collect needed information from source and destination nodes. In view of these, it is a challenge to detect incipient multicast congestion without network support.

Machine learning has already been applied in a number of areas without much background information, and has given useful results. While machine learning can attain the correct output by learning the input/output functionality, it is particularly appropriate to address more complex problems where traditional approaches cannot be used to compute the desired outcome from a set of inputs, or where that computation may be very expensive. In the case of learning to detect multicast congestion, the output is a simple yes/no tag, i.e. as a binary output value. Thus, multicast congestion detection can be viewed as a binary classification problem.

Classification methods used in machine learning have been applied extensively in scientific research, such as Fisher's linear discrimination [41], Rosenblatt's perceptron [91], back-propagation [92] and neural networks. Support vector machines (SVMs) [25] are a new type of learning method proposed by Vapnik and Cortes for classification problems. SVM maps a problem's input space into some high dimensional feature space. It is a powerful method; since its introduction, it has already

outperformed most other systems in a variety of applications.

In this thesis, we propose a new end-to-end single-rate scheme using SVM to detect multicast congestion before packet loss occurs. Our scheme focuses on learning to detecting the congestion problem in advance.

1.2 Assumptions

To resolve the congestion management problem of Internet Protocol (IP) multicast, we make the following assumptions which are similar to Li and Kalyanaraman's multicast congestion avoidance (MCA) scheme [62].

1.2.1 SSM Multicast Model

We assume that only one node can become a sender and forward data to all the others in a multicast group (source-specific multicast model, i.e. SSM model) [52]. We also presume the source of a group may open and close the group. Any receiver should be able to freely join or leave the group without informing others.

1.2.2 End-to-End Multicast Network Model

The model of our multicast network is end-to-end, i.e. it only requires support from source and receivers. According to the IP multicast protocol RFC 1112 [30], our multicast routers only forward multicast packets to all the nodes. Therefore, we do not require any information about the underlying network topology, network traffic model and routing status.

1.2.3 Background Flow

We do not have any information about the parameters of the background flows, including the flavor of TCP, the size of the maximum congestion window, and the estimated round trip time (RTT). We also assume that every multicast packet only passes through the same link once, and our routes can remain unchanged for a long time.

1.2.4 Data Transmission

We do not provide any mechanism to guarantee the reliability in data transmission. Consequently, we assume that there is another module which ensures reliable data transmission.

1.3 Problem Statement

IP multicast is an efficient mechanism for simultaneously transmitting bulk data to multiple receivers. But it still has not yet been deployed in today's Internet. Congestion is one the most important issues impeding the development and deployment of IP multicast and multicast applications. Many congestion control approaches have been proposed to handle the multicast congestion problem. However, few of the approaches focus on using machine learning to detect multicast congestion in advance. In this thesis, we use accumulation measurement and support vector machines to detect incipient congestion in an end-to-end multicast network. We develop theoretical models of intelligent multicast congestion detection and verify their performance by ns-2 simulations and statistical experiments.

1.4 Research Hypothesis

This thesis is designed to test the hypothesis that incipient multicast congestion can be detected by support vector machines (SVMs). Machine learning has already been successfully applied in a number of areas without much background information, and gives useful results. Because of tackling multicast congestion problem of the end-to-end assumptions, we cannot obtain the opportune and accurate congestion information directly from inside the network. Therefore, machine learning is particularly appropriate due to the absence of congestion information and the unpredictable variance of network congestion.

1.5 Technical Goals

Since multicast congestion detection can be viewed as a binary classification problem, the main technical goal of the research is to detect end-to-end multicast

congestion before packet loss occurs using support vector machine classification. We need to establish a training set that includes structural information about the multicast network for SVM. We also need to collect the statistics of the multicast flow as the working set of SVM, and label these unlabeled data using SVM for congestion detection. In addition, we will evaluate the performance of our scheme by comparing it with another approach using accumulation measurement.

1.6 Research Methodology

In this section, we describe the specific tasks that will be performed to address the principal scientific questions and the objective: detect incipient end-to-end multicast congestion using SVM. Simulations and analyses are used for the objective. A small multicast network will be simulated. A detection agent containing SVM will be created at each receiver. The detection agent gathers the statistics of the multicast flow, and labels these unlabeled data for congestion detection using SVM classifier. Analyses of the ns-2 simulation results will be performed to verify the basic performance of our scheme.

1.7 Contributions

Since only end-to-end congestion control protocols are suitable for the current Internet [93, 12, 35], we focus on issues in multicast congestion management based on end-to-end assumptions, i.e., it only obtains support from source and receiver nodes. Therefore, we cannot obtain the opportune and accurate congestion information such as buffer size and bottleneck bandwidth directly from inside the network. Machine learning is particularly appropriate due to the absence of congestion information and the unpredictable variance of network congestion. We study the situation where the support is provided from the receiver side. And only one group is allowed for each multicast session under the situation we consider.

For this situation, we propose multicast congestion detection (MCD) scheme, an end-to-end multicast congestion detection scheme using support vector machines. It is a single-rate scheme in the sense that only one multicast group is allowed for a multicast session. Since all receivers have the same throughput rate, the sender

adjusts the transmission rate according to the slowest receiver. MCD provides a proactive detection mechanism when backlog is being built up in bottleneck queue, whereas other reactive detection schemes used for congestion control respond when bottleneck queues are full and packets are beginning to be dropped. In our scheme, receivers first collect the needed information from the received packet pattern. A training set containing congestion status is constructed in accordance with this labeled information. When a decision function is established based on the training data, our SVM can detect incipient congestion on the receiver side. As shown by simulations, SVM can achieve great accuracy in predicting congestion.

1.8 Thesis Organization

In Chapter 2, we present an overview of Internet Protocols. We will briefly discuss these protocols and standards in a top-down manner, i.e from the application layer towards the physical layer.

In Chapter 3, we will first discuss single-rate network multicast congestion control solutions. we then discuss multicast avoidance solutions which are the foundation of our work. Since network congestion solutions with learning mechanisms are similar to our work in this thesis, we discuss them in more details.

In Chapter 4, we present an overview on Support Vector Machines for classification and regression. We also discuss the mathematical details of SVMs in this chapter.

In Chapter 5, we develop a proactive end-to-end multicast congestion detection scheme using support vector machines on the receiver side. At first, a training set is generated by measuring accumulation. After off-line SVM training, we can detect incipient congestion using SVM classification on the receiver side.

In Chapter 6, we illustrate the effectiveness of the multicast congestion detection solution developed in this thesis. Since we assume our multicast network to be a blackbox, it is difficult to analyze the performance of multicast congestion management scheme. Therefore, we use simulations and statistical experiments as the major methodology for performance evaluation.

We conclude our work in Chapter 7 and discuss the future research briefly.

All the abbreviations used in this thesis are explained in a Glossary which is in Appendix C.

CHAPTER 2
Internet Protocols

2.1 Introduction

Internet communication is one of the most exciting and important technologies of our time. The Internet interconnects millions of computers, providing various information and services, such as electronic mail, online chat, file transmission, and other documents of the World Wide Web. All of these devices are called hosts or end systems. End systems are connected together by communication links. End systems, routers, and other devices run protocols that control the sending and receiving of information within the Internet. A protocol defines the format and the order of messages exchanged between two or more communicating entities, as well as the actions taken on the transmission and/or receipt of a message or other event [59]. The Internet makes widespread use and expansion of protocols. Different protocols are used to implement different communication tasks.

Today's Internet traces its beginnings back to the early 1960s. After much work, an overall plan for the so-called ARPAnet (Advanced Research Projects Agency Network of the US Department of Defense) was introduced in [87], the first packet-switched computer network and a direct ancestor of today's public Internet. Following on from this, ARPAnet had grown to approximately 15 nodes by 1972. The first end-to-end network-control protocol (NCP) between ARPAnet and end systems was completed by Steve Crocker [28]. The ARPAnet host protocol was transited from NCP to TCP/IP (Transmission Control Protocol and Internet Protocol) as of January 1, 1983 [80]. In the late 1980s, a new host-based scheme [30] was introduced for TCP congestion control. The Domain Name System (DNS) was also developed for mapping between Internet names and their corresponding 32-bit IP address. The Web was invented at CERN by Tim Berners-Lee in 1989–1991 [7], which brought the Internet into the homes and businesses of millions of people worldwide. The recent development and widespread deployment of the World Wide Web have brought with it a new community. A new coordination organization, the

World Wide Web Consortium (W3C), has taken on the responsibility for evolving the various protocols and standards associated with the Web. Thus, over more than two decades of Internet activity, we have seen a steady evolution of organizational structures designed to support and facilitate an ever-increasing community working collaboratively on Internet issues [118].

The Internet is an extremely complicated system. There are many pieces to the Internet: numerous applications and protocols, various types of end systems and connections between end systems, routers, and various types of link-level media. To reduce design complexity, network designers organize protocols—and the protocols are implemented by network hardware and software—in layers. Each protocol belongs to one of the layers. Each layer may implement one or more of the generic set of tasks, such as error control, flow control, segmentation and reassembly, multiplexing and connection setup. When taken together, the protocols of the various layers are known as the protocol stack [59]. The Internet stack consists of five layers: the physical, data link, network, transport and application layers. We will briefly discuss the layers in the following.

2.2 Application Layer

The application layer is responsible for supporting network applications. It defines how an application running on one system passes messages to another. The application layer includes many protocols, including HTTP (Hyper Text Transfer Protocol) to support the Web, SMTP (Simple Mail Transfer Protocol) to support electronic mail, FTP (File Transfer Protocol) to support file transfer, and DNS (Domain Name System) to support directory service.

2.2.1 The Web and HTTP

The HyperText Transfer Protocol (HTTP) is defined in [8] and [40]. It is used for exchanging messages between a Web client and a Web server or between intermediate machines and Web servers. HTTP uses TCP as its underlying transport protocol. It does not provide reliability or retransmission. Once a connection is established, the Web client sends HTTP requests to the server and receives HTTP

responses from it. Because an HTTP server does not keep any information about the clients, HTTP is said to be a stateless protocol. HTTP allows bidirectional transfer and capability negotiation between clients and servers. To improve response time, HTTP supports caching of Web pages on the client side. A proxy server can be used for Web pages caching and request response between a client and a server.

2.2.2 File Transfer Protocol: FTP

File transfer is among the most frequently used Internet applications. FTP [83] is the major TCP/IP transfer protocol. FTP provides interactive access and authentication control. In a typical FTP session, if a user wants to access a remote account, he must provide a user identification and a password. After that, he can transfer files from the local file system to the remote file system and vice versa.

2.2.3 Electronic Mail in the Internet

Electronic mail has been around since the beginning of the Internet. Email is the most popular application of the Internet because it offers a fast, convenient method of transferring information. The Internet mail system has three major components: user agents, mail servers, and the Simple Mail Transfer Protocol (SMTP). SMTP is defined in RFC (Request for Comments) 2821 [57]. It uses the reliable data transfer service of TCP to transfer mail from the sender's mail server to the receiver's mail server. SMTP does not normally use intermediate mail servers for sending mail. Communication between a client and server consists of simple ASCII (American Standard Code for Information Interchange) text. To send content different from ASCII text, the sending user agent must include additional headers in the message. These extra headers are defined in RFC 2045 [45] and RFC 2046 [46], the MIME (Multipurpose Internet Mail Extensions) extension to RFC 822 [29]. If a remote user wants to retrieve mail from a permanent mailbox, two mail access protocols, POP3 [74] and IMAP [26] allow the user to manipulate the mailbox content.

2.2.4 DNS–The Internet's Directory Service

There are two ways to identify a host–by its hostname and by its IP address. Users prefer to assign machines pronounceable, easily remembered names, while

routers prefer fixed-length, hierarchically structured IP addresses. For mapping between hostnames and IP addresses, we use the Internet's Domain Name system (DNS) [70, 71] to provide the directory service. The DNS is an application-layer protocol that allows hosts and name servers to communicate in order to provide the translation service. It implements a distributed database in a hierarchy of name servers [59]. DNS is usually employed by other application-layer protocols, such as HTTP, SMTP and FTP.

2.3 Transport Layer

The transport layer provides the service of transporting application-layer messages from one application program to another. There are two transport protocols in the Internet, TCP and UDP. TCP provides a connection-oriented service to its applications. This service includes reliable transport and flow control. TCP also divides the stream of data being transmitted into small pieces and provides a congestion control mechanism, so that a source may adjust its transmission rate when the network is congested. The UDP protocol provides its application an unreliable, connectionless service. There is no handshaking between sender and receiver before sending a datagram.

2.3.1 User Datagram Protocol: UDP

The User Datagram Protocol or UDP, defined in RFC 768 [82], provides the primary mechanism that application programs use to send datagrams to other application programs. After obtaining messages from the application process, UDP attaches source and destination port number fields and other small fields, and passes the resulting segment to the network layer. The network layer encapsulates the segment into an IP datagram and then delivers the segment to the receiving host. UDP uses the destination port number to transfer the segment's data to the correct application process. However, UDP does not use acknowledgements to make sure messages arrive, and it does not provide feedback for flow control. For this reason, UDP is known as connectionless. The lack of congestion control in UDP is a potentially serious problem [43].

2.3.2 Transmission Control Protocol: TCP

TCP (Transmission Control Protocol) is defined in RFC 793 [81], RFC 1122 [15], RFC 1323 [55], RFC 2018 [67], and RFC 2581 [1], which provides a reliable, connection-oriented service to the invoking application. TCP guarantees to deliver a stream of data from sending process to receiving process without duplication or data loss by using flow control, sequence numbers, acknowledgments and timers technique. TCP also provides congestion control, which prevents any one TCP connection from swamping the links and switches between communicating hosts with an excessive amount of traffic [59].

Unlike UDP, TCP is a connection-oriented protocol that requires both endpoints to send preliminary segments to each other to establish the parameters of the ensuing data transfer before one endpoint can begin to send data to another. Once a TCP connection is established, the two application processes can send data to each other. TCP connections can provide full-duplex data transfer between sender/receiver pairs.

TCP views data as an ordered stream of bytes that it divides into segments with a TCP header for transmission. The TCP segment consists of header fields and a data field. The TCP header carries the expected identification and control information, such as source port, destination port, sequence number and acknowledgement number.

To handle timeout and retransmission, TCP estimates the round-trip time (RTT) between sender and receiver [53, 76]. The weighted average and the variance of RTT, which are defined in RFC 2988 [76], are required for computing TCP timer management. When timeout occurs, TCP responds to the timeout event by retransmitting the segment that caused the timeout. TCP then restart the timer.

TCP provides a flow-control service to its applications to reduce the possibility of the sender overflowing the receiver's buffer. Using flow control, TCP can match the rate at which the sender is sending to the rate at which the receiving application is reading. TCP provides flow control by having the sender maintain a receive window, which specifies how much free buffer space is available on the receiver's end. Because TCP is full-duplex, the sender maintains a distinct receive window at

each side of the connection.

To establish a connection, TCP uses a three-way handshake. It is both necessary and sufficient for correct synchronization between the two ends of the connection. Some operations are required to accomplish the handshake. The client-side TCP first sends a special TCP segment to the server-side TCP. The special segment can be identified by its SYN (synchronization) bit set in the code field. Once the IP datagram containing the TCP SYN segment arrives at the server host, the server allocates the TCP buffers and variable to the connection and sends a connection-granted segment to the client TCP. This connection-granted segment has both the SYN bit and ACK bits set, indicating that it acknowledges the first SYN segment as well as continuing the handshake. Upon receiving the connection-granted segment, the client also allocates buffers and variables to the connection and sends the server another segment which is merely used to acknowledge the server's connection-granted segment. Once the preceding three steps have been completed, the client and server hosts can send segments containing data to each other.

TCP provides congestion control to handle congestion in the Internet. Since the IP layer provides no explicit feedback to the end systems regarding network congestion, TCP must use end-to-end congestion control rather than network-assisted congestion control [59]. To control congestion, TCP maintains a congestion window on either side of connection. The congestion window is used to restrict data flow not to exceed the receiver's buffer size when congestion occurs. The sender can adjust its transmission rate by adjusting the size of the congestion window.

TCP uses "lost event" as the indication of congestion when a datagram is lost. An additive-increase, multiplicative-decrease (AIMD) congestion control algorithm is then used for transmission rate adjustment. The algorithm has three major components: additive-increase, multiplicative-decrease, slow start and reaction to timeout event. The TCP sender continues to increase its sending rate exponentially fast until there is a loss event, at which time the congestion window is cut in half. Whenever initiating traffic on a new connection or increasing traffic after a period of congestion, the TCP sender begins by transmitting at a slow rate but increases its sending rate exponentially. In addition, TCP maintains a threshold variable which

determines the window size at which slow start will end and congestion avoidance will begin.

2.4 Network Layer and Routing

The transport layer that provides the process-to-process communication services relies on the network layer which provides host-to-host communication service. The Internet's network layer has three major components. The network layer uses routing protocols to determine the route or path that packets take through the network from source to destination. The Internet protocol defines network-layer addressing, the fields in the datagram, and packet handling. The Internet Control Message Protocol (ICMP) is used for error reporting by hosts, routers, and gateways. In addition, we also discuss briefly the Internet Group Management protocol (IGMP) and Dynamic Host Configuration Protocol (DHCP) in the subsection.

2.4.1 Routing Protocols

The network layer uses routing protocols to determine the path or route when a source host transfers packets to the destination host. A route algorithm used by the routing protocol finds a "least cost" path for source to destination routing. Two types of routing algorithms are used in the Internet: the link state algorithm and the distance vector routing algorithm.

The link state algorithm [34] calculates the least-cost path from the source node to all other nodes in the network. After the nth iteration of the algorithm, the least-cost paths are computed to n destination nodes. When the link state algorithm terminates, a routing table can be constructed according to the information by storing, for each destination, the next-hop node on the least-cost path from the source to the given destination.

The distance vector algorithm [5, 44] is asynchronous in that it does not require all of the nodes to operate in lockstep with each other. A distance table is maintained at each node. A node receives some information from its neighbors. After performing a calculation with the distance vector algorithm, it transmits the results of its calculation back to its neighbors. This process continues until no more

information is exchanged between neighbors.

2.4.2 The Internet Protocol: IP

The network protocol in the Internet is called the Internet Protocol, which defines network-layer addressing, the fields in the datagram, and the actions taken by routers and end systems on a datagram based on the values in these fields. There are two versions of the IP in use today, IP version 4 [79] and IP version 6 [50, 31].

2.4.2.1 Internet Protocol version 4: IPv4

The widely deployed Internet Protocol version 4 (IPv4) is defined in RFC 791 [79]. Because each interface on every host and router in the global Internet must have an IP address for sending and receiving IP datagrams, IPv4 provides the addressing services by classful addressing technique. Every IP datagram has a source address field and a destination address field. Each IP address is 32 bits long. The data field of the datagram is filled with a TCP or UDP segment. There are also other key fields in the IPv4 datagram, such as version number, header length, type of service, datagram length, identifier, flags, fragmentation offset, time-to-live, protocol, header checksum, options, and so on. In addition, IPv4 uses fragmentation to divide a large datagram into smaller pieces when the datagram needs to traverse a network that has a small maximum transfer unit (MTU).

2.4.2.2 Internet Protocol version 6: IPv6

Since the 32-bit IP address space of IPv4 was approaching its limit, a new IP protocol, IPv6 [50, 31] was developed to respond to this need for a large IP address space. IPv6 increases the size of the IP address from 32 to 128 bits. In addition to unicast and multicast addresses, IPv6 uses a new type of address, known as anycast address, to deliver datagrams to any one of a group of hosts. A streamlined 40-byte fixed-length header defined by IPv6 allows for faster processing of the IP datagram. A new encoding of options allows for more complicated options processing. IPv6 supports flow labeling, which allows the routers to associate a datagram with a specific flow and priority. In addition, IPv6 uses a new ICMP protocol (ICMPv6) [24] for error reporting.

2.4.3 Internet Control Message Protocol: ICMP

The Internet Control Message Protocol (ICMP) is specified in RFC 792 [78], which is used by hosts, routers, and gateways to communicate network-layer information to each other. ICMP can report error conditions back to the original source of the datagram. ICMP is considered part of IP but a higher level protocol. ICMP messages are carried as IP payloads in order to travel across several physical networks to reach their destinations. ICMP messages have a type and a code field, and also contain the first 64 data bits of the datagram causing the problem.

2.4.4 Internet Group Management Protocol: IGMP

Multicast routers and hosts use the Internet Group Management Protocol version 2 (IGMPv2) [39] to communicate group membership information. A multicast router can use IGMP to determine that one or more hosts on the local network have decided to join a specific multicast group before propagating multicast membership information. IGMPv2 [39] has three message types, *membership_query*, *membership_report* and *leave_group*. A general *membership_query* message is sent by a router to all hosts to determine the set of all multicast groups that have been joined by the hosts. A router can determine whether a specific multicast group has been joined by hosts using a specific *membership_query*. The specific query includes the multicast address of the group being queried in the multicast group address field of the IGMP *membership_query* message. In addition, IGMP provides feedback suppression for performance optimization. To do so, each *membership_query* message sent by a router includes a "maximum response time" field. A host waits a random amount of time between zero and the maximum response time value. If the host observes a *membership_report* message from some other attached host for that given multicast group, it suppresses its own pending *membership_report* messages [59]. The final type of IGMP, *leave_group*, is optional. According to the Internet multicast service model [30], any host can join a multicast group at the network layer. A host simply delivers a *membership_report* IGMP message to its attached router. The router will soon begin transferring multicast datagrams to the host. Joining a multicast group is receiver-driven.

2.4.5 Dynamic Host Configuration Protocol: DHCP

The Dynamic Host Configuration Protocol (DHCP) [37] is often used to assign IP addresses to hosts dynamically. DHCP is a client-server protocol. A newly arriving host can obtain network configuration information, including an IP addresses from a DHCP server or a DHCP relay agent of the network. The newly arriving host firstly broadcasts its DHCP discover message. A DHCP server responds back to the client with a DHCP offer message when it receives a DHCP discover message. The client will respond to the DHCP server with a DHCP request message, and echo back the configuration parameters. The server confirms the requested parameters with a DHCP ACK message. Once the client receives the DHCP ACK message, it can use the DHCP-allocated IP address for the lease duration. DHCP also provides a mechanism which allows a client to renew its lease on an IP address.

2.5 Link Layer

A link-layer protocol is used to deliver a datagram over an individual link. The link-layer protocol defines the format of the packets exchanged between the nodes at the ends of the link, and the actions taken by these nodes when sending and receiving these packets [59]. A link-layer protocol provides the basic services for datagram transmission, including framing, link access, reliable delivery, flow control, error detection, error correction, half-duplex, and full-duplex. Examples of link-layer protocols include Ethernet, PPP (Point-to-point Protocol), ATM (Asynchronous Transfer Mode), and so on.

2.5.1 Ethernet

The original Ethernet local area network [69] was introduced in the mid 1970s by Bob Metcalfe and David Boggs. There are many different Ethernet technologies on the market today, but they all use the same frame structure. The sending adapter encapsulates the IP datagram within an Ethernet frame and delivers the frame to the physical layer. The receiving adapter receives the frame from the physical layer, extracts the IP datagram, and delivers the IP datagram to the network layer. The Ethernet frame has six key fields, including data field, destination address, source

address, type field and cyclic redundancy check (CRC).

Every adapter of nodes in an Ethernet local area network (Ethernet LAN) has an Ethernet address for transmitting frames to each other on the LAN. Because there are both Internet IP addresses and LAN addresses, the address resolution protocol (ARP) [77] is implemented for translation between them. Every Internet host and router has an ARP module on a LAN. The ARP module maintains an ARP table which contains the mappings of IP addresses to LAN addresses, and a time-to-live (TTL) entry for each address mapping. An ARP packet also needs to be constructed on the sending node for address resolution. The ARP packet queries all the other nodes on the LAN to determine the LAN address corresponding to the IP address that is being resolved [59].

2.5.2 Point-to-point Protocol: PPP

The point-to-point protocol (PPP) [2, 101] operates over a point-to-point link which is a link directly connecting two nodes, on each end of the link. PPP contains some basic functions for data transmission on the point-to-point link; including packet framing, transparency, multiple network-layer protocols, multiple types of links, error detection, connection liveness, network-layer address negotiation and simplicity. In addition, PPP's link-control protocol (LCP) and family of PPP network-control protocols are used to accomplish the initialization, maintenance, error reporting and shutdown of a PPP link.

2.5.3 Asynchronous Transfer Mode: ATM

The standards for ATM were first developed in the mid-1980s. The ATM standards call for cell switching with virtual circuits. ATM encodes data traffic into small (53 bytes; 48 bytes of data and 5 bytes of header information) fixed-sized cells. To achieve high transfer speeds, an ATM network consists of one or more high-speed switches, and uses optical fibers for connections. In addition, the lowest layers of an ATM network use fixed-size frames known as cells. ATM switch can process cells quickly because each cell is the same size. ATM provides connection-oriented service. Before a host connected to an ATM switch can send cells, a connection must be established manually or the host must first interact with the switch to

specify a destination. Once a connection succeeds, the local ATM switch chooses an identifier for the connection, and passes the connection identifier to the host along with a message that informs the host of success. The host uses the connection identifier when sending or receiving cells. When a connection is no longer needed, the host again communicates with the ATM switch to request that the connection be broken. The switch then disconnects the two hosts. After a disconnection, the hosts cannot communicate until they establish a new connection. Furthermore, identifiers used for a connection can be recycled. Once a disconnection occurs, the switch can reuse the connection identifier for a new connection [23].

2.6 Other Protocols

There are some protocols for network management and multimedia networking applications.

2.6.1 Simple Network Management Protocol: SNMP

The Simple Network Management Protocol version 2 (SNMPv2) [17] is used to deliver MIB (Management Information Base) information among managing entities and agents executing on behalf of managing entities. SNMP uses request-response mode and trap messages for network management. In a request-response mode, the SNMP managing entity sends a request to an SNMP agent who receives the request, performs some actions, and sends a reply to the request. The request will be used to query or modify MIB object values associated with a managed device. Trap messages are used to notify a managing entity of an exceptional situation that has resulted in changes to MIB object values. SNMPv3 [18] and SNMPv2 use the same general framework, but SNMPv3 provides additional services for security and administration. SNMPv3 security is known as user-based security. A user can be identified by a username, with which security information such as password, key value, or access privileges are related. SNMPv3 provides for encryption, authentication, protection against playback attacks, and access control [59].

2.6.2 Protocols for Multimedia Networking Applications

Real-time streaming protocol (RTSP), defined in RFC 2326 [97], allows a media player to control the transmission of a media stream. Some protocols, such as RTP [96], SIP [90], and H.323 [114], are used to transmit multimedia data for real-time interactive applications which include Internet phone and video conferencing.

2.7 The Future of the Internet

The clearest part of the future of the Internet is that of nomadic computing and smart spaces. The availability of lightweight, inexpensive, high-performance, portable computing devices plus the ubiquity of the Internet has enabled users to access the Internet and data on their home or work computers from anywhere in the world. However, nomadic computing is only one step. The next step will enable us to move out from the netherworld of cyberspace to the physical world of smart spaces. Our environments will come alive with artificial intelligence and embedded technology. These technologies will allow our environment to provide the IP services we want [59]. Future Internet possibly includes the following additional key components [59]:

- The intelligent software agents will be deployed across the network whose function it is to mine data, act on that data, observe trends, and implement dynamically and adaptively.
- More network traffic generated by the embedded devices and the intelligent software agents.
- This vast fast network will be controlled by large collections of self-organizing systems.
- Amounts of information flash across this network instantaneously with this information undergoing enormous processing and filtering. The Internet will essentially be a pervasive global nervous system.

CHAPTER 3
Multicast Congestion Management

Various solutions have been proposed for congestion management. In the following, we will first discuss single-rate network multicast congestion control solutions. We will then discuss multicast avoidance solutions which are the foundation of our work. Since network congestion solutions with learning mechanism are similar to our work in this thesis, we discuss them in more details.

3.1 Multicast Congestion Control

IP multicast was first proposed by Steve Deering in 1989 in RFC 1112 [30]. In IP multicast, data is delivered from one host to multiple hosts simultaneously. Multicast routers replicate the data when they forward the packets to other networks. Consequently, to support the various applications such as video conference and distance learning, IP multicast is considered a very efficient mechanism. However, due to many technical and marketing reasons, IP multicast is still far from being widely deployed in the Internet. Congestion is one of the most important problems impeding the deployment of multicast. Much research has been done on multicast congestion control. In these congestion control schemes, the sender adjusts the transmission rate or congestion window relying on the reports from the congestion detection mechanism. Many of today's strategies for detecting congestion use positive acknowledgments (ACKs) or negative acknowledgments (NAKs) to send congestion indications after packet loss occurs. Round-trip time (RTT) measurement could be an alternative way for congestion detection. But what RTT to select is a problem since multicast network is based on tree instead of path.

There are four categories of multicast congestion control: Firstly, we will briefly discuss end-to-end unicast congestion control. Then we will discuss end-to-end single-rate and network supported multicast congestion control solutions.

3.1.1 End-to-End Unicast Congestion Control

TCP [53] is the most common end-to-end unicast congestion control protocol in the Internet today. Other variants of TCP have been introduced to improve its performance, such as Reno [54], NewReno [51], SACK [38], Vegas [16], and Westwood [19]. Other generalized end-to-end unicast congestion control algorithms have also been proposed, such as MCFC [47] and Binomial [4]. However, because of the reasons discussed in [60], the unicast schemes cannot be migrated directly to multicast situation.

3.1.2 End-to-End Single-rate Multicast Congestion Control

Since MCD is a single-rate scheme, we compare it with some of the well known schemes in this class.

3.1.2.1 DeLucia and Obraczka's Scheme Using Representatives

DeLucia and Obraczka's work in [32] is an early single-rate multicast congestion control scheme. It requires two types of feedback messages from representatives for congestion detection: Congestion Clear (CC) and Congestion Indication (CI). CC is equivalent to ACK; CCs are sent with the worst receive rate as using rate-based metrics, or with the worst RTT as using delay-based metrics. CI is equivalent to NAK; it is sent in the case of packet loss. In this scheme, the source needs to determine if congestion occurs in the network using continuous feedback packets from each receiver. The transmission rate of the data packets will be reduced and the computation complexity is O(N) where N is the number of receivers.

3.1.2.2 Pragmatic General Multicast Congestion Control Scheme: PGMCC

Rizzo's PGMCC [86] is based on the Pragmatic General multicast (PGM) protocol [103]. It needs two types of feedback, ACK and NAK. All receivers send the NAKs to the source when the data packet losses occur. After computing the estimated throughput, an acker, a representative receiver with the worst throughput will be selected for the rate adaptation at source. A simplified TCP average throughput formula [75, 68] is used to compute the lowest estimated throughput.

The source requires positive ACKs from the acker for running a windows-based congestion control scheme similar to TCP.

3.1.2.3 TCP-Friendly Multicast Congestion Control Scheme: TFMCC

Widmer's TFMCC [111] develops the equation-based unicast congestion control scheme TFRC [42] to the multicast domain. The current limiting receiver (CLR) with the lowest expected throughput is selected for the transmission rate adaptation at source using the full TCP throughput formula [75, 68]. Each receiver is required to measure packet loss event rate and RTT. The packet losses are aggregated into loss events at all receivers. Therefore, TFMCC detects congestion after packet losses occurs at receivers.

3.1.2.4 Multicast Dissemination Congestion Control Scheme: MDP-CC

In Macker et al's MDP-CC scheme [66], a list of receivers with the worst estimated throughput is selected as congestion control representatives (CCRs) using TCP throughput formula [75, 68]. To compute estimated throughput, the source gathers the feedback with loss event estimates and RTT measurements. The source dynamically chooses one worst path representative (WPR) amongst the CCRs. The transmission rate is adjusted exponentially towards the predicted rate of the WPR. In MDP-CC scheme, all receivers require to do the congestion detection after packet loss occurs for loss event estimates.

3.1.2.5 Linear Proportional Response Filter: LPRF

Bhattacharya et al's Linear Proportional Response Filter (LPRF) scheme [9] uses LPRF filter to pass loss indications (LIs) with a probability for rate adaptation. But LPRF cannot ensure that the selected receiver is the worst case receiver.

3.1.2.6 Loss-Event Oriented Source-based Multicast Congestion Control Scheme: LE-SBCC

Thapliyal et al's source-based LE-SBCC scheme [105] is built upon Bhattacharya et al's LPRF [9]. LE-SBCC uses Max-LPRF filter to pass each loss event

(LE) with a probability where is the number of LEs from receivers, and estimates RTT for rate adaptation (AIMD algorithm [21]).

3.1.2.7 Output Rate Multicast Congestion Control Scheme: ORMCC

In Li and Kalyanaraman's ORMCC scheme [63], all receivers send feedback with Throughput Rate At Congestion (TRAC) to source when packet losses are detected. The slowest receiver will be chosen as the Congestion Representative (CR) for rate adjustment at the source.

3.1.2.8 Other Schemes

There are also several other schemes in this class. In Shi's work [100], the most congested receiver is selected using a simplified TCP throughput formula. The slowest receiver sends a congestion notification (CN) to the source. AIMD algorithm [21] is used for the source rate adaptation.

Bouras's work [13] fits ATM or DiffServ networks better. Each receiver passes the loss rate and delay jitter to the source for the AIMD rate adjustment algorithm [21].

Other schemes such as RLA (Random Listening Algorithm) [110], TCP-SMO (SMO version of Transmission Control Protocol) [64], SRM-TFRC (SRM based Congestion Control Scheme for Reliable Multicast) [112], LNM (Loss Notification Mechanism) [49], BMTP (Bulk Multicast Transport Protocol) [73], and SNMCC (Self-suppressed Nack-based Multicast Congestion Control)[65], require all receivers sending ACK or NAK feedback to the source after packet losses. With these feedbacks, the source adjusts the transmission rate using different types of algorithm.

3.1.3 Network Supported Single-rate Multicast Congestion Detection

Network supported schemes are different from end-to-end source/receiver-based algorithms. They need support from routers or nodes other than source and receiver.

Siu's work [108] is a single-rate ATM scheme which extends unicast congestion control protocols to multicast and preservers max-min fairness characteristic.

Each "resource management" (RM) computes the transmission rate for congestion detection.

Sedano's work [98] is a single-rate scheme which is based on active networks. In an active network, the functions of routers can be modified using active service [104]. Receivers send feedbacks with "proper rate" and ACK to upstream routers toward the source. All routers adjust their transmission rate according to the feedbacks. It uses a hop-by-hop congestion control mechanism.

Chiu et al's scheme [22] proposed a tree-based and window-based protocol. It assumes a repair tree topology [56]. Each receiver sends ACKs which are aggregated by interior nodes of the repair tree. The congestion window of the source is adjusted in accordance with the reports.

MTCP (Multicast TCP) [85] assumes a logical tree topology. All nodes act both as a source and a receiver in the tree. They collect the congestion feedbacks (ACKs or NAKs) from their subtree and adjust the transmission rate using window-based congestion control mechanism.

3.2 Multicast Congestion Avoidance

Congestion avoidance [21] is simply the process of detecting and responding to network congestion when accumulation is being built up in bottleneck queues. As a proactive measure, congestion avoidance defers from congestion control in that it takes effect when bottleneck queues are full and packets are beginning to be dropped. Compared to a "congestion control" strategy which simply detects the congestion and necessarily induced packet loss, we easily see the advantages of congestion avoidance.

An early single-rate multicast congestion control scheme by DeLucia and Obraczka [32] detects incipient congestion at source for the paths between the source and representatives. DeLucias's scheme uses the congestion detection mechanism proposed in TCP Vegas [16], a unicast congestion avoidance scheme. Since other receivers still detect congestion with packet loss measurement, Delucia's scheme is not a fully congestion avoidance scheme.

Li and Kalyanaraman's MCA [62] is an end-to-end single-rate scheme. Their

MCA scheme detects incipient congestion using simple thresholding techniques and "accumulation concept", which is defined as the number of buffered bits of a flow inside the network. In the MCA scheme, accumulation-based congestion control [113] is extended from unicast to multicast. On the receiver side, an accumulation measurement algorithm is performed to detect congestion and responds to incipient congestion. The congestion representative (CR) sends feedbacks with congestion indications (CIs) to source when incipient congestion is detected. AIMD rate control policy [21] is used for rate adaptation. Both the slowest receiver ("Congestion Representative") selection and receiver side feedback suppression use G-TRCA rate formula instead of continual RTT measuring at all receivers.

In some "congestion control" schemes, such as PGMCC [86], TFMCC [111], LE-SBCC [105] and references therein, the receivers send feedback with congestion indications to the source when packet loss is detected. While packet marking support is provided by network components (like TCP-ECN [84]), the above schemes can also prevent packet loss.

3.3 Applications of Machine Learning for Network Congestion Management

Machine learning has been successfully applied to tackle network congestion management problem. The following discussions illustrate how to approach network congestion management issues with learning methods. They show that a learning mechanism can be of great value for network congestion management.

In Thottan's work [107, 106], network fault is detected using a sequential Generalized Likelihood Ratio (GLR) test. Time series MIB variable data is collected by SNMP [14], and divided into 2.5 minute windows. Relying on these windows, the statistical deviation between two adjacent time windows is computed by a sequential hypothesis test using the Generalized Likelihood Ratio. Then, two techniques can be used to correlate the different alarms with the values of several MIB variables: in the first scheme, a Bayesian belief network based on a directed graph can provide the hierarchical structure information of the MIB variables. The second technique is a duration filter, which correlates the propagation of many alarms to the dependencies

of the MIB variables during a certain duration period [107, 106]. The authors detect faults using statistical data from one source on the network. They also detect the patterns leading to faults in the network file system (NFS) using statistical methods. The interface statistics of a single router are collected as the input of their detection algorithm. A Generalized Likelihood Ratio (GLR) test is used to process the faults patterns.

Bivens' scheme [11] proposes using neural networks to predict the source or sources responsible for the congestion. A control agent containing a neural network is created to collect information from each managed node. It determines if network problems occur. The neural networks are trained off-line using a pattern file. Two types of pattern are used to train the neural network: one contains no network problems and another has congestion problems at various locations.

CHAPTER 4
Support Vector Machines

The classification problem has been studied widely since the notion of linear discrimination was introduced by Fischer in mid 1930s. In the 1960s, Rosenblatt introduced the perceptron as a new approach of machine learning. Later, Rumelhart, Hinton, and Williams use back-propagation technique [92] to improve the perceptron method in the mid 1980s. Artificial neural networks (ANNs) have been applied extensively in the field of pattern recognition and machine learning since the mid 1980s. ANNs implement piece-wise linear type decision functions while the perceptron constructs a linear decision function. Since it was introduced as a new type of learning method by Vapnik and Cortes, support vector machines [25] have become one of the standard tools in the Machine Learning community for classification, regression and density estimation tasks. We present the details in the following.

4.1 The Basic Concept

We begin by considering the binary classification problem. The aim of SVMs is to optimize the separating hyperplanes in a high dimensional feature space. A hyperplane is an affine subspace of dimension $n-1$ which divides the data points into two distinct classes. Especially, an n-dimensional inner product space X can be split into two parts by the hyperplane defined by the equation $w \cdot x + b = 0$.

In the simplest case, the SVM algorithm constructs a hyperplane (if possible) from which the distance of all data points of the positive class is greater than zero, and of the negative class is lesser than zero. If there exits a hyperplane which correctly classifies the training data, we call this hyperplane a separating hyperplane.

The given training set $(x_1, y_1), \ldots, (x_l, y_l)$ is for binary classification $Y = \{-1, 1\}$, where l is the number of training examples. We assume that the training data are linearly separable, i.e. there exits a separating hyperplane. Such, with $x_i \in \mathbb{R}^n$, $w \in \mathbb{R}^N$ and $b \in \mathbb{R}$, we have

$$y_i(w \cdot x_i + b) \geq 1, \;\; i \in 1, \ldots, l. \tag{4.1}$$

The hyperplane defined by Equation (4.1) is said to be in canonical form. Since the hyperplane does not change if we scale (w, b), the set of hyperplanes $(w \cdot x_i + b) = 0$ is the same as the set of all separating hyperplanes. However, it is easy to check out that the existence of a single such hyperplane implies the existence of an infinite number of canonical separating hyperplanes.The best canonical separating hyperplane is then selected by measuring the minimum distance of the hyperplane from the closest data point.

Now, the Euclidean distance of a point x_i from the plane $(w \cdot x_i + b) = 0$ is $\frac{|w \cdot x_i + b|}{\|w\|}$. According to Equation (4.1), the problem can be reduced to maximizing $\frac{1}{\|w\|}$ for all the canonical separating hyperplanes since the minimum of the numerator is 1. The value, known as the margin, measures the moving distance of the hyperplane without affecting the separation. From the data point perspective, the margin computes how much it can be moved without changing correct classification.

The maximization of the distance of the nearest data point from the hyperplane (the margin)$\frac{1}{\|w\|}$ is equivalent to the minimization of $\|w\|$ or $\frac{1}{2}\|w\|^2 = \frac{1}{2} w \cdot w$. For the construction of the optimal hyperplane (i.e. the one with maximal margin), we have a constrained optimization problem:

$$\begin{aligned} & \min_{w,b} \frac{1}{2} w \cdot w \\ \text{subject to} \quad & y_i(w \cdot x_i + b) \geq 1, \;\; i \in 1, \ldots, l, \end{aligned} \tag{4.2}$$

The constraints can be tackled by introducing Lagrange multipliers $\alpha \geq 0$, also called the dual variables. The primal Lagrangian is

$$L(w, b, \alpha) = \frac{1}{2}(w \cdot w) - \sum_{i=1}^{l} \alpha_i \{y_i(w \cdot x_i + b) - 1\}. \tag{4.3}$$

The corresponding dual is found by differentiating with respect to w and b, imposing stationarity,

$$\frac{\partial L(w,b,\alpha)}{\partial w} = w - \sum_{i=1}^{l} y_i \alpha_i x_i = 0, \tag{4.4}$$

$$\frac{\partial L(w,b,\alpha)}{\partial b} = \sum_{i=1}^{l} y_i \alpha_i = 0, \tag{4.5}$$

and resubstituting the relations obtained

$$w = w - \sum_{i=1}^{l} y_i \alpha_i x_i, \tag{4.6}$$

$$b = \sum_{i=1}^{l} y_i \alpha_i, \tag{4.7}$$

into the primal to obtain

$$\begin{aligned} L(w,b,\alpha) &= \frac{1}{2}(w \cdot w) - \sum_{i=1}^{l} \alpha_i \{y_i((w \cdot x_i) + b) - 1\} \\ &= \frac{1}{2} \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j (x_i \cdot x_j) - \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j (x_i \cdot x_j) + \sum_{i=1}^{l} \alpha_i \\ &= \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j (x_i \cdot x_j). \end{aligned} \tag{4.8}$$

The optimal (^{o}) solution of Equation (4.8) is in the form of:

$$\alpha^o = (\alpha_1^o, \ldots, \alpha_l^o) \tag{4.9}$$

and the optimal hyperplane (w^o, b^o) is determined by:

$$w^o = \sum_{i=1}^{l} \alpha_i^o y_i x_i \tag{4.10}$$

$$b^o = -\frac{max_{y_i=-1}((w^o \cdot x_i)) + min_{y_i=1}((w^o \cdot x_i))}{2} \tag{4.11}$$

Since the value of b does not appear in the dual formulation, b^o is usually taken to be the mean of the values calculated using Equation (4.11) for each non-zero α_i^o. After the separating hyperplane has been determined, it can be used to classify new data points. The relative decision function is:

$$sgn(\sum_{i=1}^{l} \alpha_i^o y_i x_i \cdot x + b). \tag{4.12}$$

4.2 Kernel-Induced Feature Spaces

In order to learn non-linear relations with a linear machine, SVM maps the input space to an higher dimensional space, in which the linear machine can be used. Consider a finite input space $X = \{x_1, \ldots, x_n\}$ with $K(x, y)$ a symmetric function on X. $K(x, y)$ is a kernel function if and only if the matrix $\mathbf{K} = (K(x_i, x_j))_{i,j=1}^n$ is positive semi-definite (has non-negative eigenvalues). A weighting λ_i for each dimension is introduced for a slight generalization of an inner product in a Hilbert space,

$$K(x, y) = \sum_{i=1}^{\infty} \lambda_i \phi_i(x) \phi_i(y) = \langle \phi_i(x), \phi_i(y) \rangle, \tag{4.13}$$

so that the feature vector becomes

$$\phi_(x) = (\phi_1(x), \phi_2(x), \ldots, \phi_n(x), \ldots). \tag{4.14}$$

According to Mercer's theorem, a continuous symmetric function $K(x, y)$ can be an inner product in the feature space $G \supseteq \phi(X)$. We have,

$$K(x, y) = \langle \phi_i(x), \phi_i(y) \rangle. \tag{4.15}$$

In particular, with the kernel extension, the dual form of the Lagrangian multipliers becomes:

$$\begin{aligned} \text{maximise} \quad & W(\alpha) = \sum_{i,j=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j K(x_i, x_j), \\ \text{subject to} \quad & \sum_{i,j=1}^{l} y_i \alpha_i = 0, \\ & \alpha_i \geq 0, \qquad i = 1, \ldots, l. \end{aligned} \tag{4.16}$$

The solution remains in the form:

$$w^o = \sum_{i=1}^{l} \alpha_i^o y_i x_i \tag{4.17}$$

$$b^o = y_i - K(w^o, x_i) \tag{4.18}$$

The resulting classification function becomes:

$$f(x) = sgn(\sum_{i=1}^{l} \alpha_i^o y_i K(x_i, x) + b^o). \tag{4.19}$$

We solve the quadratic optimization problem by calculating the parameters α^o and b^o in the feature space implicitly defined by the kernel $K(x, y)$.

We now consider some common kernels in use:

$$\text{Polynomial} \qquad K(x, y) = ((x, y) + 1)^d, \tag{4.20}$$

$$\text{Gaussian} \qquad K(x, y) = \exp(-\frac{\|x - y\|^2}{\sigma^2}), \tag{4.21}$$

$$\text{Sigmoid} \qquad \tanh((x, y) - \sigma). \tag{4.22}$$

The choice of a particular kernel is often difficult. There is a lot of work on making better kernels which implicitly define a complicated feature space. Nevertheless, regularization theory has helped in this regard [94], and has allowed the design of kernel functions to embed a priori knowledge [102].

4.3 Soft Margin Optimization

For the non-linearly separable case, the maximal margin classifier cannot be used. If very powerful kernels are used, overfitting occurs. This problem can be approached by using the soft margin optimization algorithm [25]. This approach can tolerate some misclassification of the training data by relaxing the margin constraints Equation (4.1) and optimizing the complete bound. In order to optimize the margin slack vector we need to introduce slack variables to allow the margin constraints to be violated

$$\begin{aligned} \text{subject to} \quad & y_i((w \cdot x_i) + b) \geq 1 - \xi, i = 1, \ldots, l, \\ & \xi_i \geq 0, i = 1, \ldots, l. \end{aligned} \tag{4.23}$$

The minimization problem (Equation 4.2) becomes

$$\begin{aligned} & \min_{w,b,\xi} \frac{1}{2} w \cdot w + C \sum_{i=1}^{l} \xi_i \\ \text{subject to} \quad & y_i((w \cdot x_i) + b) \geq 1 - \xi, \forall i = 1, \ldots, l \\ & \xi_i \geq 0, \forall i = 1, \ldots, l \end{aligned} \tag{4.24}$$

where the second term C is a regularization parameter. It is usually found by cross-validation. In Equation (4.24), the value of C can give the optimal bound by finding the minimum of ξ with the given value for w. Furthermore, the value of C also corresponds to the optimal choice of w.

The primal Lagrangian for the problem of Equation (4.24) is

$$L(w, b, \xi, \alpha) = \frac{1}{2}(w \cdot w) + C \sum_{i=1}^{l} \xi_i - \sum_{i=1}^{l} \alpha_i \{y_i(w \cdot x_i + b) - 1 + \xi_i\} - \sum_{i=1} l \eta_i \xi_i \tag{4.25}$$

where $\alpha_i \geq 0$ are the Lagrange multipliers. The corresponding dual is found by differentiating with respect to w, ξ and b,

$$\frac{\partial L(w,b,\xi,\alpha)}{\partial w} = w - \sum_{i=1}^{l} y_i \alpha_i x_i = 0, \tag{4.26}$$

$$\frac{\partial L(w,b,\xi,\alpha)}{\partial \xi} = C\xi - \alpha = 0, \tag{4.27}$$

$$\frac{\partial L(w,b,\xi,\alpha)}{\partial b} = \sum_{i=1}^{l} y_i \alpha_i = 0. \tag{4.28}$$

Using these constraints, the relative dual form of the Lagrangian multipliers becomes:

$$\sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) \tag{4.29}$$

$$\text{subject to} \quad 0 \leq \alpha_i \leq C, i = 1, \ldots, l$$

$$\sum_{i=1}^{l} \alpha_i y_i = 0.$$

This is still a quadratic programming problem, and the solution remains in the form:

$$w^o = \sum_{i=1}^{l} y_i \alpha_i^o x_i \tag{4.30}$$

$$b^o = y_i - (w^o, x_i) \tag{4.31}$$

The value of b^o is chosen using the relation $\alpha_i = C\xi_i$ and the Karush-Kuhn-Tucker (KKT) complementarity conditions which imply that if $C > \alpha_i^o > 0$ both $\xi_i^o = 0$ and $y_i((w, x_i) + b) - 1 + \xi_i^o = 0$. Thus, all the α_i are upper bounded by C. The constraint ensures only those data points x_i closest to the hyperplane can have non-zero Lagrange multipliers α_i. The margin defined by such a hyperplane is known as soft margin.

4.4 Support Vector Regression

The support vector method can also be applied to the case of regression. We can optimize the generalization bounds given for regression by defining a loss function of the regularization problem. In feature space, the hyperplane is an estimator of the loss function, and penalize deviation of data points from the hyperplane [58]. $\|w\|$ reflects the smoothness of the hyperplane, so that the objective function is

$$\frac{1}{2}w \cdot w + C\sum_{i=1}^{l} l(y_i, w \cdot x_i + b). \tag{4.32}$$

where the loss function is defined by

$$l(y_i, y) = (y_i - y)^2, \tag{4.33}$$

we can obtain the ridge regression which is a more stable modification of linear regression in many cases. The standard loss function applied for SV regression is the ε-insensitive loss function

$$l_\varepsilon(y_i, y) = \max(0, |y_i - y| - \varepsilon). \tag{4.34}$$

Thus, the training points are penalized which differ from the estimator by more than ε, and then in a linear fashion. As a consequence, the regression function relies only on a subset of the training data. Since there exist additional Lagrange multipliers resulting from the absolute value in the loss function, the derivation of the resulting optimization differs slightly from what we have seen earlier [58]. However, we can solve the Wolfe dual problem:

$$\begin{aligned} \text{maximise} \quad & \sum_{i=1}^{l} y_i(\alpha_i^* - \alpha_i) - \varepsilon \sum_{i}^{l}(\alpha_i^* + \alpha_i) \\ & -\frac{1}{2}\sum_{i,j=1}^{l}(\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j)(x_i \cdot x_j), \\ \text{subject to} \quad & 0 \leq \alpha_i, \alpha_i^* \leq C, i = 1, \ldots, l, \\ & \sum_{i}^{l}(\alpha_i^* - \alpha i) = 0, i = 1, \ldots, l. \end{aligned} \tag{4.35}$$

where w^o can be expanded as a linear combination of the x_i's, and b^o can be calculated from the data points with Lagrange multipliers in the open interval (0, C) [58].

CHAPTER 5
MCD: An End-to-end Multicast Congestion Detection Scheme using Support Vector Machines

In our scheme, we consider the multicast congestion detection problem for the following situation:

> Support is provided on the receiver side, and a multicast session only uses one multicast group.

Consider a scenario, where a company only has one class–D address from IANA (Internet Assigned Numbers Authority), but some multicast data is required to transmit over the public network (assuming multicast is supported) using the open group model [30].

We present an end-to-end multicast congestion detection (MCD) scheme as the solution. Our scheme detects multicast network congestion before packet loss occurs using support vector machines. The scheme is purely receiver-based in the sense that it operates on a stream of multicast data packets from the source without any other support from network elements, source or in the packet format of underlying multicast transport protocols. One earlier work by Li and Kalyanaraman. [62] uses simple thresholding techniques and "accumulation concept" to detect multicast network congestion without necessarily inducing packet loss. However, the accumulation measurement algorithm is based on aggregated flow information, whereas our work uses detailed flow information for congestion detection. Another single-rate multicast congestion control work by DeLucia et al. [32] uses the congestion detection method proposed in TCP Vegas [16], a unicast congestion avoidance scheme. DeLucia et al.'s scheme only detects incipient congestion on the source side for the paths between the source and representative receivers, when other receivers still detect congestion by packet losses. For comparison, our scheme detects incipient congestion on the receiver side for all paths. In other "congestion control" schemes including PGMCC (Pragmatic General Multicast Congestion Control

Scheme) [86], TFMCC (TCP-Friendly Multicast Congestion Control Scheme) [111] and references within, congestion detection can be implemented by monitoring the number of dropped packets.

In our scheme, SVMs [25] detect incipient congestion on the receiver side (Figure 5.1). We gather statistics from the stream of multicast data packets as the input data of SVM. The outputs of SVM are collected as congestion indications. Before using SVM classifier to detect multicast congestion, the classifier is trained off-line. For training SVM, labeled data with two labels "congestion" and "uncongestion" are generated by accumulation measurement algorithm and regression techniques.

Simulation will show that MCD can achieve great accuracy in predicting incipient congestion.

5.1 Algorithm Description

In MCD, four algorithms are implemented on the receiver side, including accumulation measurement algorithm, regression estimation algorithm, statistics generation algorithm and SVM classification algorithm. Training dataset is generated by accumulation measurement algorithm [62] and regression estimation algorithm.

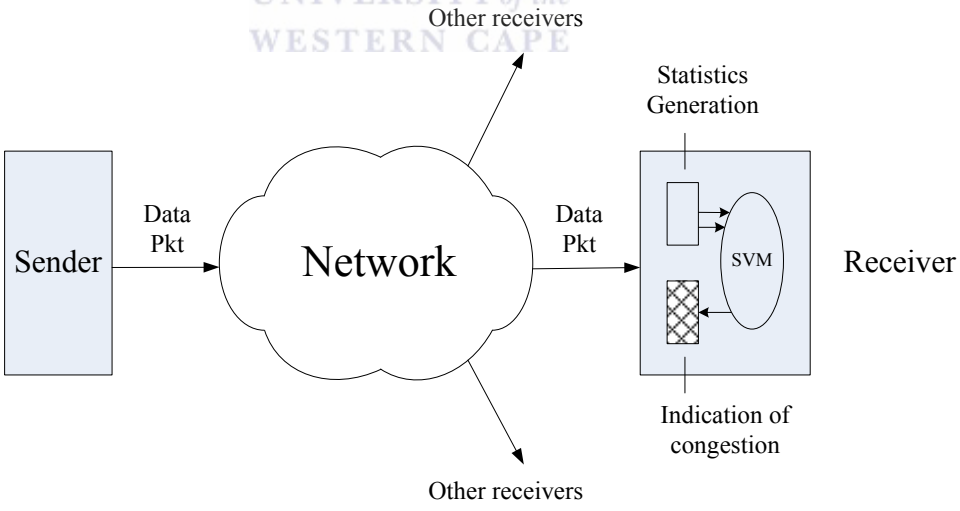

Figure 5.1: MCD Model.

We gather statistics from the stream of multicast data packets for training dataset generation and SVM classification using statistics generation algorithm. After off-line SVM training, incipient congestion is detected using SVM classification. We present the details in the following.

5.1.1 Accumulation Measurement

An accumulation measurement algorithm is performed to obtain the training samples used to train the SVM for incipient congestion. We borrow the ideas from the MCA scheme [62]. In Li and Kalyanaraman's MCA scheme, the concept of accumulation based on unicast is extended to multicast. The accumulation measurement algorithm is explained by Figure 5.2 and the following specifications.

Similar to Li and Kalyanaraman's MCA scheme, the control packets are multicast to all receivers at a fixed time interval. A normal data packet can be easily changed to a control packet by turning on some one-bit flag and adding its sending time into the optional field. On the receiver side, accumulation is measured relying on the control packets. If accumulation is larger than two packets, congestion occurs, and then the time is recorded. After that, we use the regression technique to obtain the estimation value of the statistics about the multicast stream when incipient congestion is detected.

5.1.2 Regression Estimation

In the period of training set generation, we gather two types of labeled data ("congested" and "uncongested") as training sample for SVM classification. On the receiver side, accumulation is measured to detect congestion as every control packet arrives. Using the regression technique, we can calculate the estimation values of the statistic about the multicast stream according to the control packet arrival time.

Although the control packet arrival time can be collected by measuring accumulation, it is difficult to measure directly the statistics about the multicast stream at the moment. Thus we use the regression estimation algorithm to obtain the estimation values of the statistic. During training set generation, the number of packets received at fixed intervals (40 ms) is recorded. Two variables can be obtained from the sampling, namely the number of packets received at fixed intervals

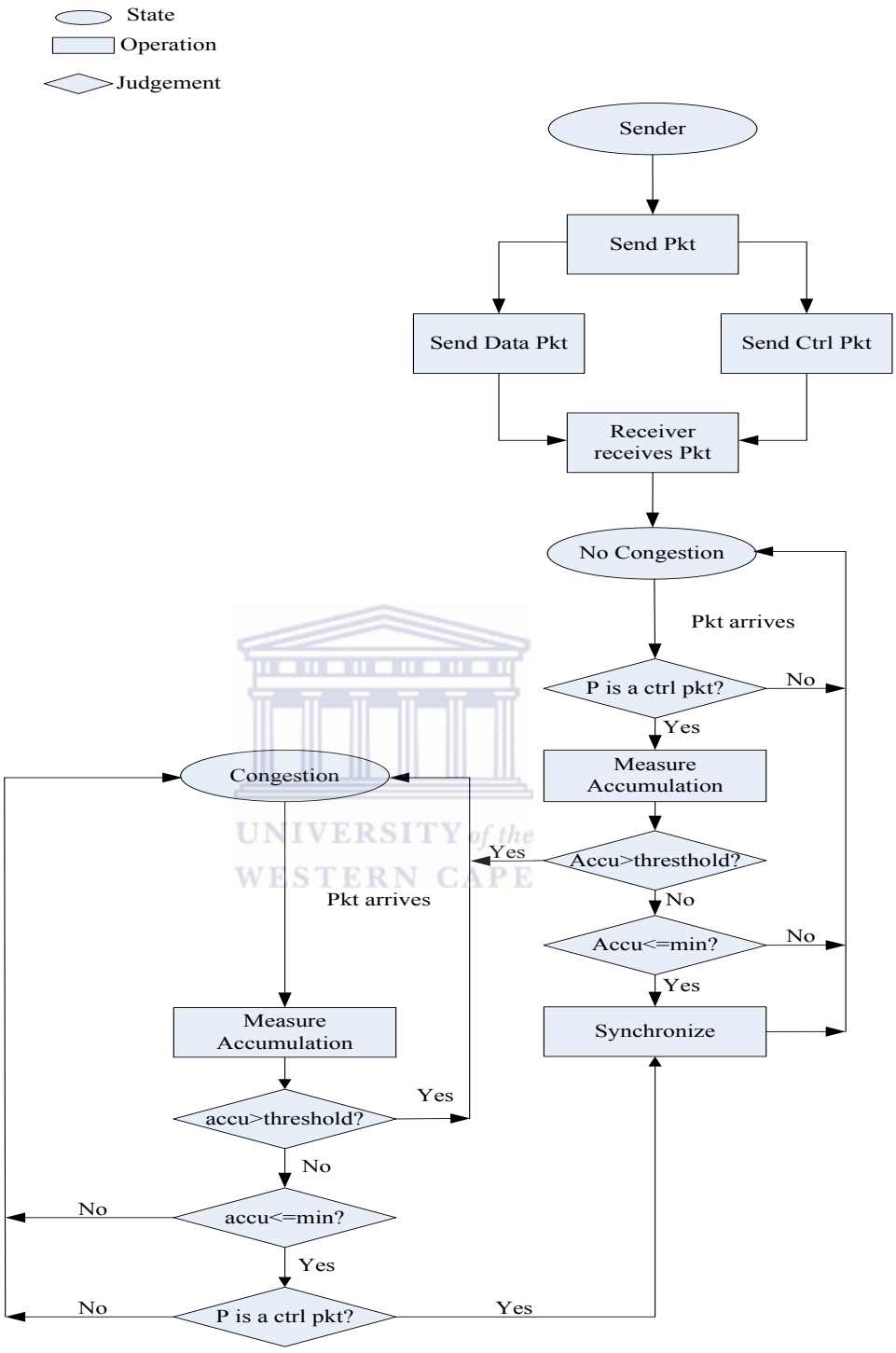

Figure 5.2: Accumulation Measurement.

$Y_i = \{Y_1, Y_2, \ldots, Y_n\}$ and the time of sampling $X_i = \{X_1, X_2, \ldots, X_n\}$ where n is sampling number before the control packet arrives. We assume that the regression line of variable Y, on variable X has the form $\beta_0 + \beta_1 X$. Then we can write the linear regression model

$$Y = \beta_0 + \beta_1 X + \varepsilon, \tag{5.1}$$

where β_0 and β_1 are the parameters of the model, and ε is the increment by which any individual Y may fall off the regression line.

We use estimates b_0 and b_1 instead of β_0 and β_1, and ε is difficult to discover since it changes for each sample Y; thus we can write

$$\hat{Y} = b_0 + b_1 X, \tag{5.2}$$

where $\hat{Y}$ denotes the predicted value of Y for a given X. In our scheme, $\hat{Y}$ is the estimation value of the number of packets received as the control packet arrives, and X is the control packet arrival time.

Assume that we have available n sets of samples $(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)$, where X is the time of sampling; Y is the number of packets at fixed intervals; n is sampling number as the control packet arrives. Then by Equation (5.2), we can write

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i, \tag{5.3}$$

for $i = 1, 2, \ldots, n$, so that the sum of squares of deviations from the true line is

$$S = \sum_{i=1}^{n} \varepsilon_i^2 = \sum_{i=1}^{n} (Y_i - \beta_0 - \beta_1 X_i)^2. \tag{5.4}$$

We use the estimates b_0 and b_1 instead of β_0 and β_1 to produce the least possible value of S. We can determine b_0 and b_1 by differentiating Equation (5.4) first with respect to β_0 and then with respect to β_1 and setting the results equal to zero. Now, we have

$$\begin{aligned}\frac{\partial S}{\partial \beta_0} &= -2\sum_{i=1}^{n}(Y_i - \beta_0 - \beta_1 X_i) \\ \frac{\partial S}{\partial \beta_1} &= -2\sum_{i=1}^{n} X_i(Y_i - \beta_0 - \beta_1 X_i)\end{aligned} \quad . \tag{5.5}$$

So that the estimates b_0 and b_1 are given by

$$\begin{aligned}\sum_{i=1}^{n}(Y_i - b_0 - b_1 X_i) &= 0 \\ \sum_{i=1}^{n} X_i(Y_i - b_0 - b_1 X_i) &= 0\end{aligned} \quad . \tag{5.6}$$

where we substitute b_0 and b_1 for β_0 and β_1, when we equate Equations (5.5) to zero. From Equations (5.6) we have,

$$\begin{aligned}\sum_{i=1}^{n} Y_i - nb_0 - b_1\sum_{i=1}^{n} X_i &= 0 \\ \sum_{i=1}^{n} X_i Y_i - b_0\sum_{i=1}^{n} X_i - b_1\sum_{i=1}^{n} X_i^2 &= 0\end{aligned} \tag{5.7}$$

or

$$\begin{aligned} b_0 n + b_1\sum_{i=1}^{n} X_i &= \sum_{i=1}^{n} Y_i \\ b_0\sum_{i=1}^{n} X_i + b_1\sum_{i=1}^{n} X_i^2 &= \sum_{i=1}^{n} X_i Y_i\end{aligned} \quad . \tag{5.8}$$

The solution of Equations (5.8) for b_1, the slope of the fitted straight line, is

$$b_1 = \frac{\sum X_i Y_i - [(\sum X_i)(\sum Y_i)]/n}{\sum X_i^2 - (\sum X_i)^2/n} = \frac{\sum(X_i - \overline{X})(Y_i - \overline{Y})}{\sum(X_i - \overline{X})^2} \tag{5.9}$$

where all summation are from $i = 1$ to n. The solution of Equations (5.8) for b_0, the intercept at $X = 0$ of the fitted straight line, is

$$b_0 = \overline{Y} - b_1\overline{X}. \tag{5.10}$$

Substituting Equation (5.10) into Equation (5.2) gives the estimated regression equation

$$\hat{Y} = \overline{Y} + b_1(X - \overline{X}) \tag{5.11}$$

where $\hat{Y}$ is the estimation value of Y (the number of packets received at a fixed interval as the control packet arrives); X is the time of sampling in the period of statistics generation; b_1 is given by Equation (5.9). And we have

$$\begin{aligned} \overline{X} &= \frac{1}{n}\sum_{i=1}^{n} X_i, \\ \overline{Y} &= \frac{1}{n}\sum_{i=1}^{n} Y_i. \end{aligned} \tag{5.12}$$

According to Equations (5.9, 5.11, 5.12, 5.13), we can calculate the estimated number of packets received at a fixed interval ($\hat{Y}$) when the control arrives.

5.1.3 Statistics Generation

In our scheme, the statistics are collected on the receiver side for constructing the training set as well as the working set. We can obtain the statistics—the sample mean and variance—using simple statistical techniques. The details of this process are presented in the following discussion.

5.1.3.1 Statistics Generation for Training Set

The training set is generated by using two types of training sample labeled as "congested" or "uncongested". As shown in Figure 5.3, incipient congestion is detected by measuring accumulation when the control packet arrives; the estimated number of packets, relating to the control packet arrival time, is then computed using the regression estimation algorithm. Depending on the estimated number of packets, the estimated statistics can be obtained for training set generation when every control packet is received. The estimated statistics are labeled as "congested" or "uncongested" according to the detection results from the accumulation measurement.

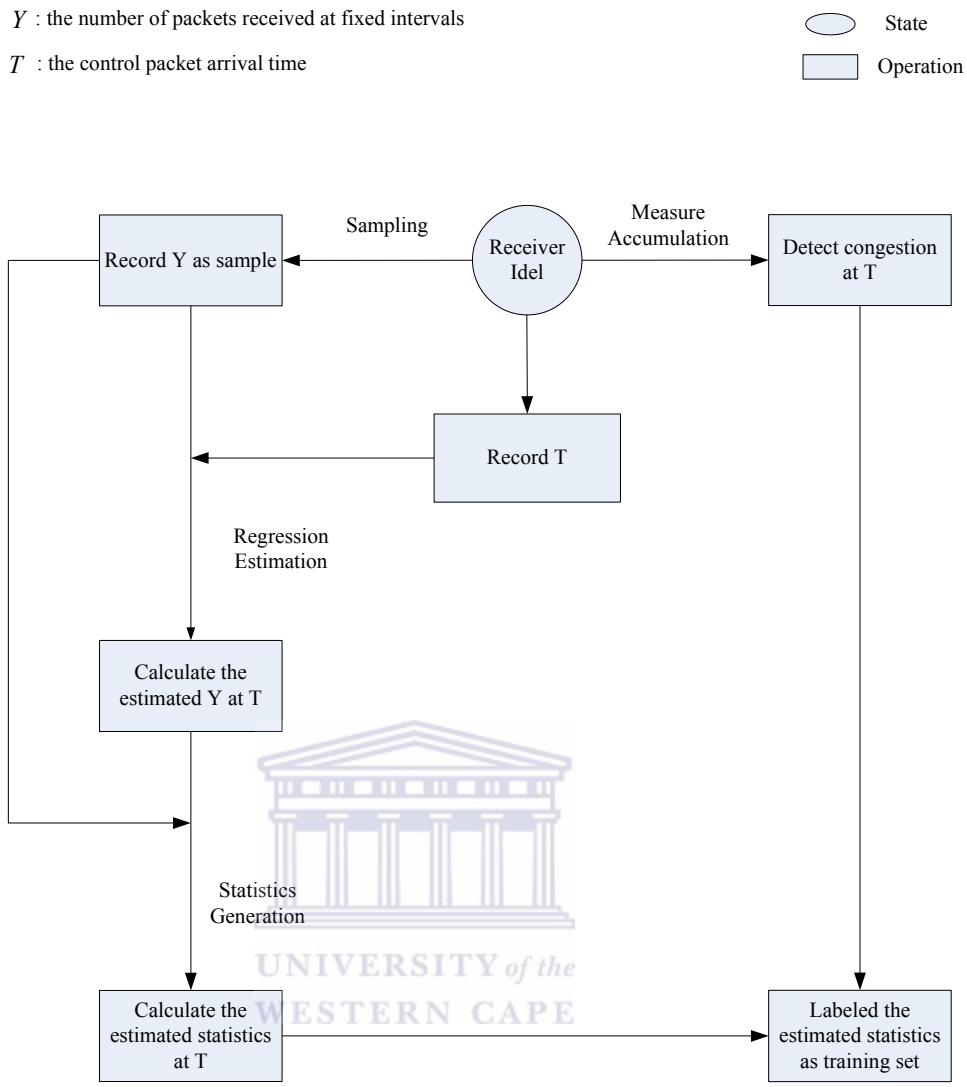

Figure 5.3: Training Set Generation.

On the receiver side, the number of packets received at fixed intervals (40 ms) is recorded as sample during training set generation. Two parameters $\overline{Y_k}$ and S_k are computed as the estimated statistics when every control packet is received. We have,

$$\overline{Y_k} = \frac{1}{n+1}\sum_{i=1}^{n+1} Y_i + \frac{\hat{Y}}{n+1} \tag{5.13}$$

where $\overline{Y_k}$ is the estimated mean of sample (the number of packets received at fixed intervals); Y_i is the number of packets received at fixed intervals before the control packet arrives; n is sampling number before the control packet arrives. According to the unbiased formula

$$S = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(X_i - \overline{X})^2}, \tag{5.14}$$

we have,

$$S_k = \sqrt{\frac{1}{n}\sum_{i=1}^{n+1}(\hat{Y} - \overline{Y_k})^2} \tag{5.15}$$

where S_k is the estimated standard deviation (variance); $\hat{Y}$ is the estimation value as the control packet arrives; $\overline{Y_k}$is the estimated mean of sample (the number of packets received at fixed intervals); n is sampling number before the control packet arrives. The estimated mean of sample $\overline{Y_k}$ and the estimated standard deviation (variance) S_k will be computed to create the training dataset where our SVM learns about incipient congestion.

5.1.3.2 Statistics Generation for Working Set

During SVM classification, the number of packets received at fixed intervals (40 ms) is recorded as sample at the receiver side. Two parameters $\overline{Y_t}$ and S_t are computed as the statistics of the multicast stream when every sample is recorded. The unlabeled data is also the working set of SVM classification. We have,

$$\overline{Y_t} = \frac{1}{n}\sum_{i=1}^{n} Y_t \tag{5.16}$$

where $\overline{Y_t}$ is the sample mean (the number of packets received at fixed intervals); Y_t is the number of packets received at fixed intervals; n is sampling number.

The unbiased formula is also used for sample variance. We have,

$$S_t = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(Y_t - \overline{Y_t})^2} \tag{5.17}$$

where S_t is the sample standard deviation (variance); $\overline{Y_t}$ is the sample mean (the number of packets received at fixed intervals), Y_t is the number of packets received at the fixed intervals and n is sampling number.

5.1.4 Support Vector Machines

Support Vector Machines [25] is a new type of learning method constructed by Vapnik and Cortes. It is used in our scheme to detect incipient congestion that occurs within a time series. We chose to use SVM since it has been successfully applied in many fields, such as word sense disambiguation, text classification, part-of-speech tagging, web page classification, and question classification [109]. SVM first maps input space into some high dimensional feature space, then constructs a linear decision surface in this feature space that relates to a non-linear decision surface in the original input space. The comparison between SVM and other classification methods is given in [95, 48, 6]. In our scheme, Vapnik's C-Support Vector Classification algorithm [109] is used to classify the statistics of multicast stream and arbitrate the congestion according to the training sample. We summarize the algorithm here.

Our training data consists of N pairs $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, with $x_i \in R^2$ and $y \in \{-1, 1\}$. The primal form considered is

$$\begin{aligned} & \min_{w,b,\xi} \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i \\ \text{subject to} \quad & y_i(w^T \phi(x_i) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, \qquad i = 1, \ldots, l. \end{aligned} \tag{5.18}$$

It can be rephrased as

$$\begin{aligned} & \min_{\alpha} \frac{1}{2}\alpha^T Q\alpha - e^T\alpha \\ & 0 \le \alpha_i \le C, \qquad i = 1, \ldots, l, \\ \text{subject to} \quad & y^T\alpha = 0, \end{aligned} \tag{5.19}$$

where e is the vector of all ones, $C > 0$ is the upper bound, Q is an $l \times l$ positive semi-definite matrix, $Q_{ij} \equiv y_i y_j K(x_i, x_j)$, and $K(x_i, x_j) \equiv \phi(x_i)^T \phi(x_j)$ is the kernel. Here training samples x_i are mapped into a higher dimensional space by the function ϕ. The decision function is

$$sgn(\sum_{i=1}^{l} y_i \alpha_i K(x_i, x) + b). \tag{5.20}$$

5.2 Scheme Description

As it is depicted in Figure 5.4, the receivers collect the training samples for estimating the classifier function using accumulation measurement and regression estimation algorithm. After off-line SVM training, we calculate the statistics of the multicast stream at fixed intervals as a working set (unlabeled data) for SVM classification, and finally by using the estimated function, we classify the unlabeled data (working set) when every statistic generates. According to the result of SVM classification, we can detect incipient congestion on the receiver side. Therefore, the MCD scheme can be split into two operating parts, viz. training set generation and SVM Classification. We present the details in the following.

5.2.1 Training Set Generation

To obtain the training samples where our SVM learns about incipient congestion, accumulation measurement and regression computation are executed on the receiver side. The sender multicasts data and control packets to the receivers, and the receivers obtain the congestion time before packet loss occurs using an accumulation measurement algorithm. Meanwhile, the number of packets received at

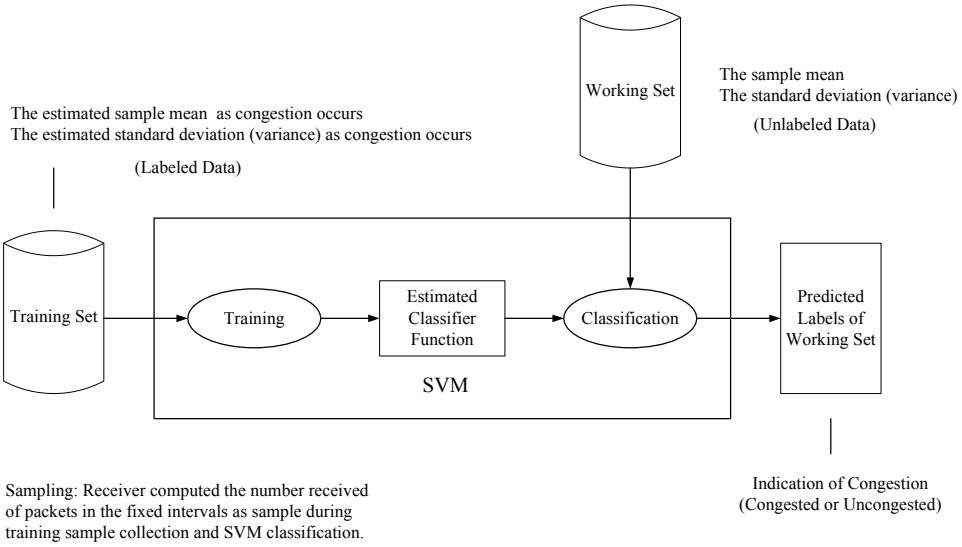

Figure 5.4: SVM Learning.

fixed intervals (40 ms), (it is same with the transmission interval of control packets) is recorded as the sample on the receiver side. The estimation value of the number of packets received is calculated using the regression algorithm while every control packet arrives. And then, we can compute the sample mean and variance depending on the estimated number of packets. After that, the estimated statistics are labeled as "congested" or "uncongested" based on the detection result of accumulation measurement. Our training set can be generated using the labeled statistics.

5.2.2 SVM Classification

The sender multicasts data packets to the receivers after our SVM is trained. On the receiver side, the number of packets received at fixed intervals (40 ms) is sampled. Two parameters, the average of number of packets received at fixed intervals and variance, are computed as the statistics of the multicast stream when every sample is recorded. The unlabeled data is also the working set of SVM classification. The C-Support Vector Classification algorithm [109] is used to classify the working set and arbitrate the congestion.

CHAPTER 6
Simulations and Experiments

To verify the performance of our scheme, we ran several `ns-2` [3] simulations and statistical experiments. We collect samples and record every control packet arrival time on `ns-2`, and generate the training set by statistical experiments. LIBSVM is used to train and optimize our SVM. After off-line training, we use the SVM classifier to detect incipient multicast congestion.

6.1 Sample Collection

In all `ns-2` simulations, the data packet size is 1,000 bytes, the bottleneck buffer size is 10K bytes, the initial RTT (round trip time) is 100 milliseconds. The simulation time is 60 seconds. The random number generator is Pareto. We first collect samples and record every control packet arrival time on the simple topology in Figure 6.1. We used the star topology to generate asynchronous and independent congestion on different paths. There are 16 ends nodes in the topology. Between each pair of source i and receiver i ($i = 1 \cdots 16$), there are three TCP Reno flows. Furthermore, there is a multi-receiver MCD flow from source 17 to all 16 receivers. Therefore, on a path between the router and any receiver, the multi-receiver MCD flow competes with three TCP flows. On the receiver side, the number of packets received at fixed intervals—40 ms, it is same with the transmission interval of control packets—is recorded as the sample; every receiver obtains 1,476 samples. 16,925 control packets are multicasted to receivers, and every control packet arrival time is gathered for training set generation. The time samples are labeled as "congested" or "uncongested" based on the detection result of accumulation measurement. And their statistics are as shown in Table 6.1.

6.2 Training Set Generation

After obtaining the samples from `ns-2` [3], we generate the training set for our SVM using the statistical methods. The estimation value of the number of packets

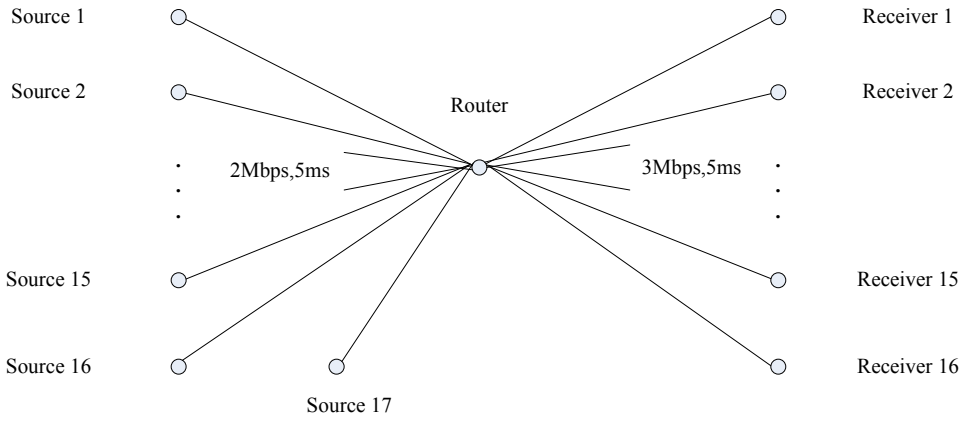

Figure 6.1: 16-Receiver Star Topology.

Table 6.1: Statistics of the Time Sample during Sample Collection.

Receiver No.	Control Pkts Received	Uncongested	Congested
18	1,066	811	255
19	1,025	788	237
20	1,043	798	245
21	1,084	821	263
22	1,046	811	235
23	1,049	810	239
24	1,070	826	244
25	1,057	820	237
26	1,032	796	236
27	1,086	838	248
28	1,060	822	238
29	1,074	815	259
30	1,070	822	248
31	1,053	812	241
32	1,081	840	241
33	1,029	785	244
Sum	16,925	13,015	3,910

received is calculated using the regression algorithm when every control packet arrives. We then compute the sample mean and variance depending on the estimated number of packets. The estimated statistics are labeled as "congested" or "uncongested" depending on the labeled time sample (control packet arrival time). Our training set can be generated using the labeled statistics. There are 16,925 records in the training set, which fall into two classes. Each record has two attributes, viz. the estimated sample mean and variance.

6.3 SVM Training

Chang et al's LIBSVM [20] is used to train and optimize our SVM. LIBSVM is an integrated software for support vector classification (C-SVC, nu-SVC), regression (epsilon-SVR, nu-SVR) and distribution estimation (one-class SVM). To choose the best parameters of our SVM, `grid.py` [117], a model selection tool, is used for C-SVM classification. It uses cross validation (CV) technique to estimate the accuracy of each parameter combination in the specified range. During training period, parameter C was set to 32,768 and parameter γ was set to 8. After training is finished, the cross validation accuracy is 90.9%.

6.4 Detect Congestion using SVM Classification

After off-line training, an `ns-2` simulation is run to test the performance of MCD. LIBSVM is used for SVM training and classification. The same `ns-2` configuration used by sample collection is implemented during SVM classification. The simulation time is again 60s. But the sender only multicasts data packets to receivers, and then the receivers detect incipient congestion using SVM classification instead of accumulation measurement. Every receiver collects 1,466 samples during the simulation. The sampling time is labeled as "congested" or "uncongested" depending on the classification result of SVM. And their statistics are as shown in Table 6.2. The congestion time gathered by measuring accumulation is compared with collecting by MCD, as shown in Table 6.3.

We compare the detection results (the number of congestions and the time of congestion) from MCD with the results collected by measuring accumulation.

Table 6.2: Statistics of MCD Sampling

Receiver No.	Sampling Number	Uncongested	Congested
18	1,466	1,115	351
19	1,466	1,144	322
20	1,466	1,217	249
21	1,466	1,135	331
22	1,466	1,175	291
23	1,466	1,142	324
24	1,466	1,170	296
25	1,466	1,170	296
26	1,466	1,121	345
27	1,466	1,162	304
28	1,466	1,154	312
29	1,466	1,138	328
30	1,466	1,157	309
31	1,466	1,064	402
32	1,466	1,134	332
33	1,466	1,136	330
Sum	23,456	18,334	5,122

Figure 6.2–6.17 clearly demonstrates that MCD can achieve great accuracy in predicting incipient congestion by SVM classification. However, at the Receiver 20 and Receiver 22, SVM detects incipient congestions before using accumulation measurement. The data packets will not be lost because we assume that there is another congestion avoidance module which should be able to adjust transmission rate while incipient congestion occurs. But at the Receiver 31, it will possibly suffer congestion while SVM detects incipient congestions with delay. The accuracy of our SVM classification is about 90%. Failing includes either missing congestion or predicting congestion while there is none.

Chang et al's LIBSVM [20] is used to train and optimize our SVM. LIBSVM is an integrated software for support vector classification (C-SVC, nu-SVC), regression (epsilon-SVR, nu-SVR) and distribution estimation (one-class SVM). We use a model selection tool, `grid.py` [117], for our C-SVM classification. During training period, parameter C was set to 32,768 and parameter γ was set to 8. After training

is finished, the cross validation accuracy is 90.9%.

We changed the seed for the random number generator (RNG) of `ns-2` simulation. The experiment was repeated nine times. We estimated the effect of MCD scheme by comparing between two congestion curves. Their statistics are as shown in Table 6.4. Some receivers will possibly suffer congestion while SVM detects incipient congestions with a clear delay. The percentage of the receivers which can detect incipient congestions in time by using SVM will be calculated for each experiment. We then calculate its confidence interval. We also calculate the cross validation accuracy for each experiment.

Table 6.3: Number of Congestions Compared with MCD and Accumulation Measurement

Receiver No.	MCD	Accumulation Measurement
18	351	255
19	322	237
20	249	245
21	331	263
22	291	235
23	324	239
24	296	244
25	296	237
26	345	236
27	304	248
28	312	238
29	328	259
30	309	248
31	402	241
32	332	241
33	330	244
Sum	5,122	3,910

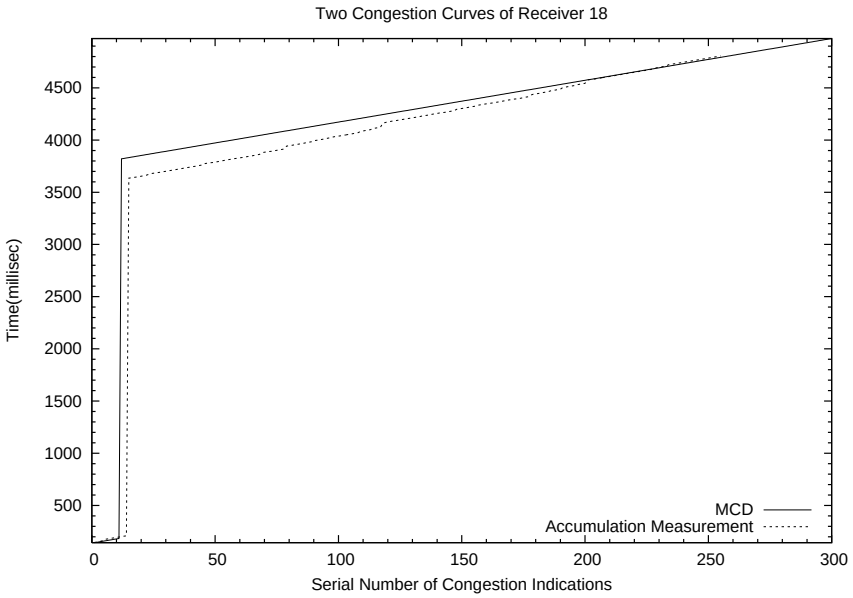

Figure 6.2: Two Congestion Curves from MCD and Accumulation Measurement at Receiver 18

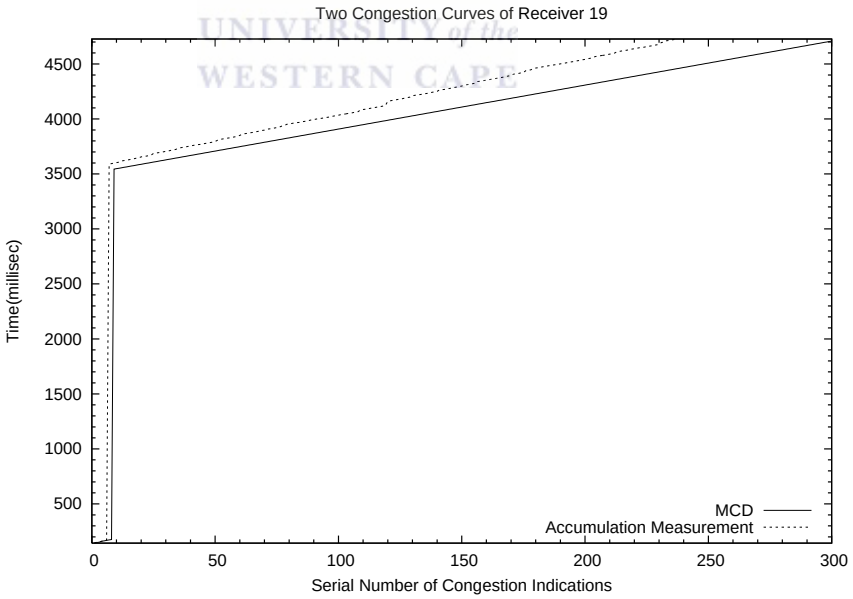

Figure 6.3: Two Congestion Curves from MCD and Accumulation Measurement at Receiver 19

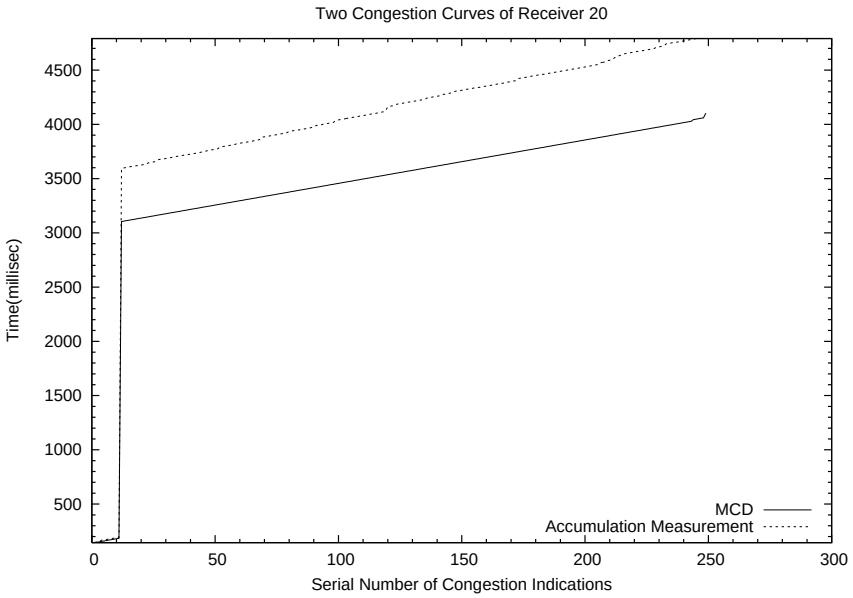

Figure 6.4: Two Congestion Curves from MCD and Accumulation Measurement at Receiver 20

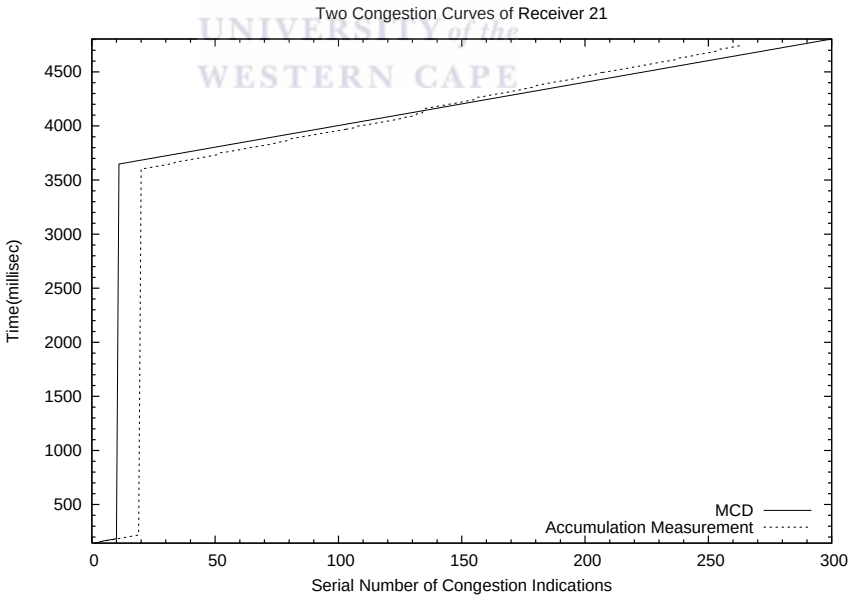

Figure 6.5: Two Congestion Curves from MCD and Accumulation Measurement at Receiver 21

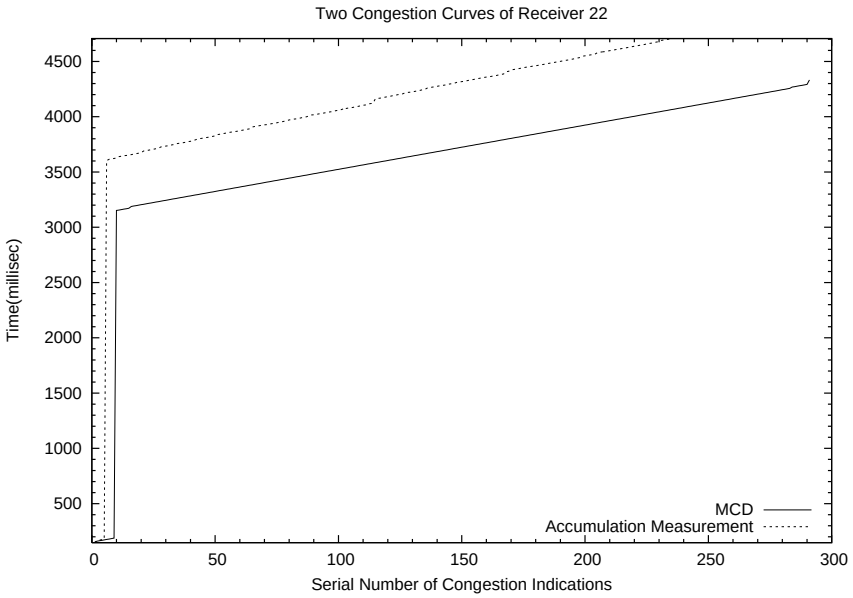

Figure 6.6: Two Congestion Curves from MCD and Accumulation Measurement at Receiver 22

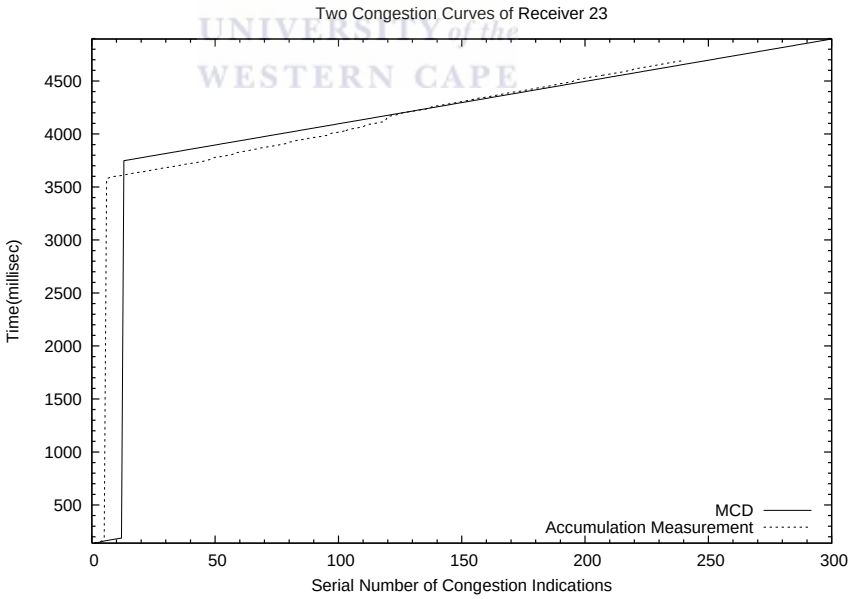

Figure 6.7: Two Congestion Curves from MCD and Accumulation Measurement at Receiver 23

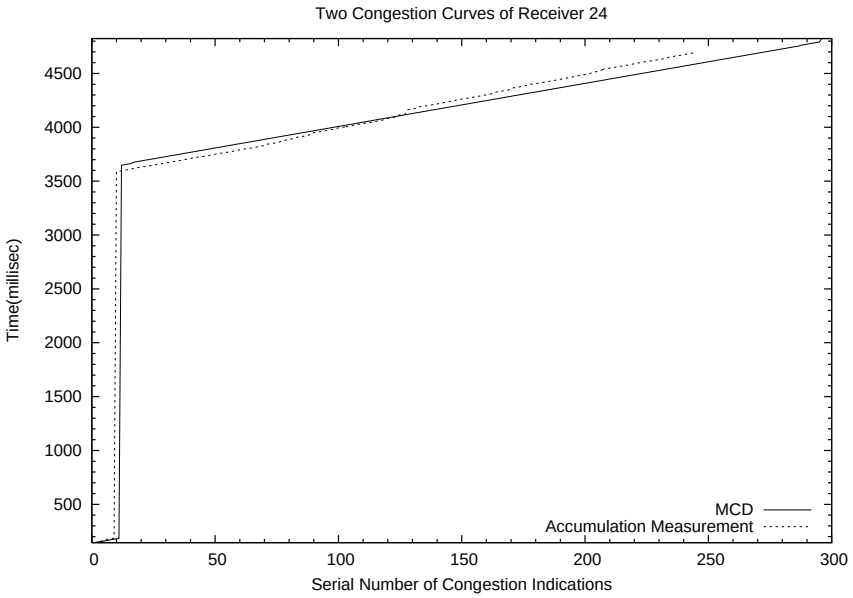

Figure 6.8: Two Congestion Curves from MCD and Accumulation Measurement at Receiver 24

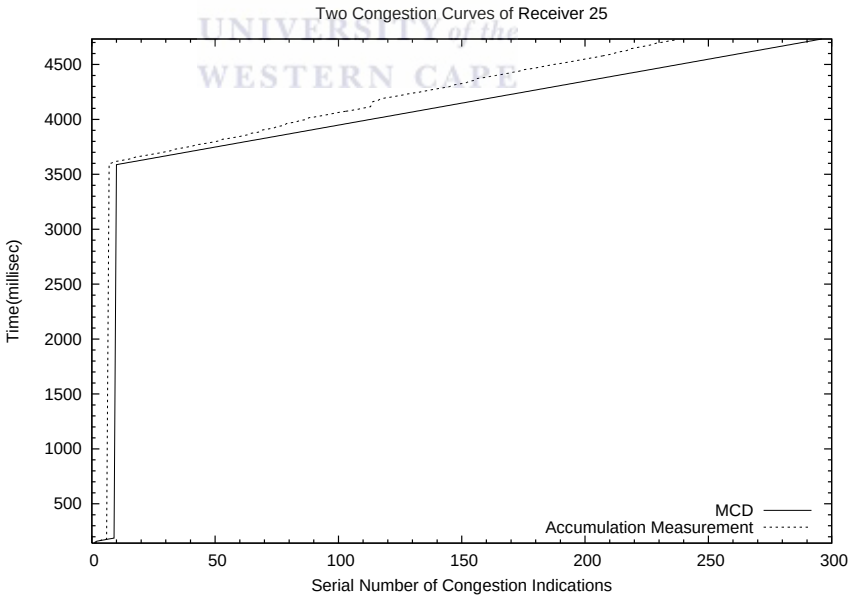

Figure 6.9: Two Congestion Curves from MCD and Accumulation Measurement at Receiver 25

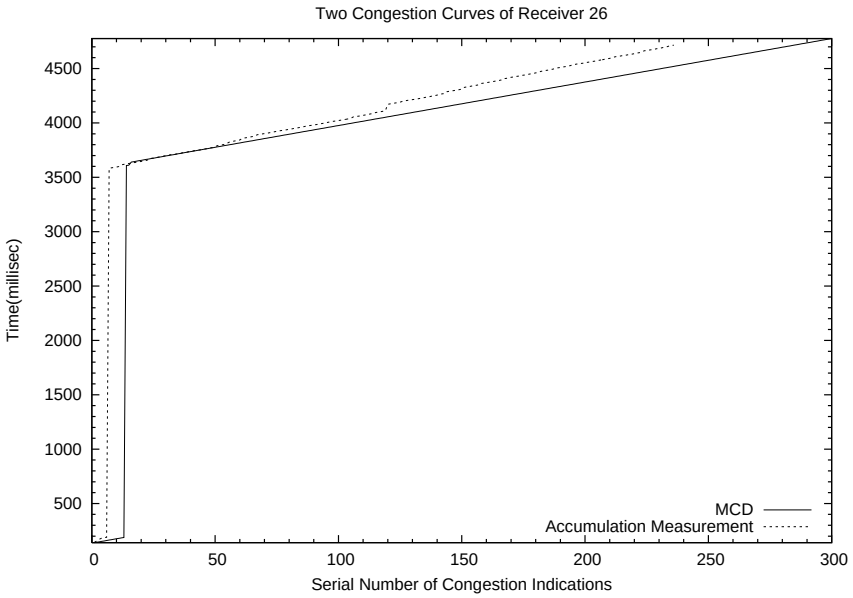

Figure 6.10: Two Congestion Curves from MCD and Accumulation Measurement at Receiver 26

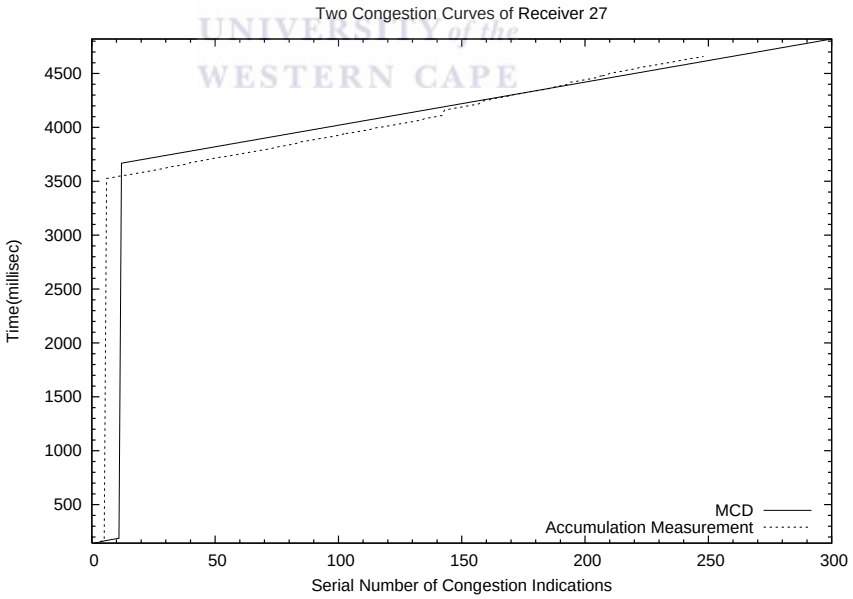

Figure 6.11: Two Congestion Curves from MCD and Accumulation Measurement at Receiver 27

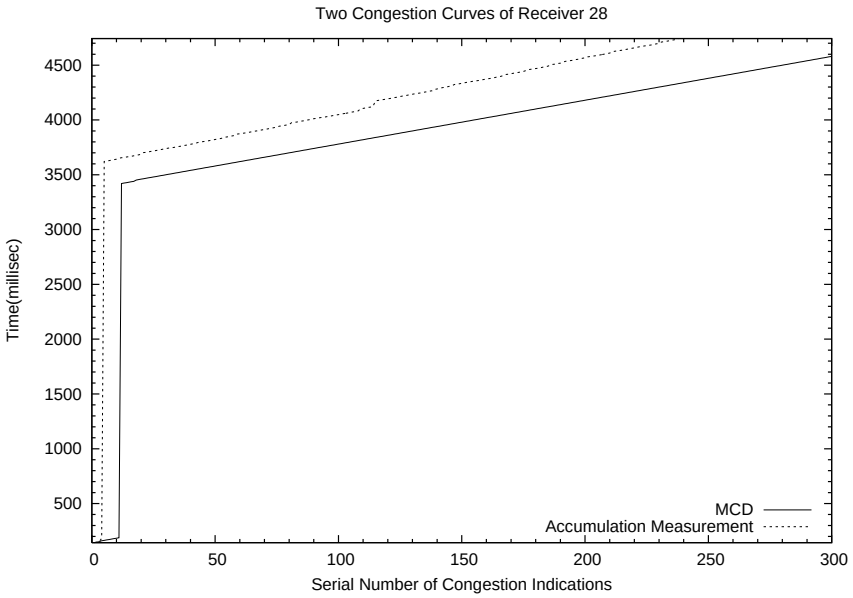

Figure 6.12: Two Congestion Curves from MCD and Accumulation Measurement at Receiver 28

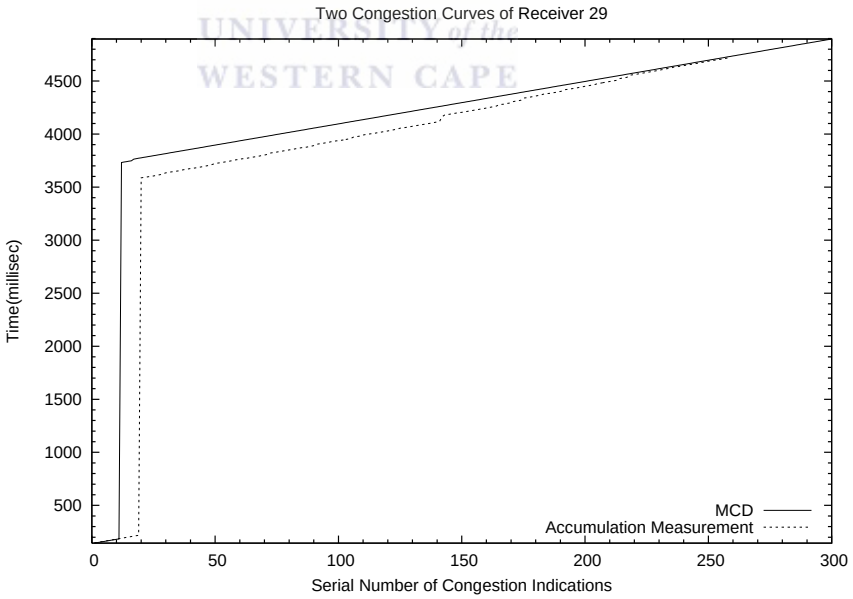

Figure 6.13: Two Congestion Curves from MCD and Accumulation Measurement at Receiver 29