Mohammad Hovaidi Ardestani

Congestion Control in Information Centric Networking using Neural Networks

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo, July 2014

Thesis supervisor:

Prof. Jörg Ott

Thesis instructor:

Ph.D. Pasi Sarolahti

Aalto University
School of Electrical Engineering

AALTO UNIVERSITY SCHOOL OF ELECTRICAL ENGINEERING	ABSTRACT OF THE MASTER'S THESIS

Author: Mohammad Hovaidi Ardestani

Title: Congestion Control in Information Centric Networking using Neural Networks

Date: July 2014 Language: English Number of pages:11+77

Department of Communications and Networking

Professorship: Networking Technology Code: S-38

Supervisor: Prof. Jörg Ott

Instructor: Ph.D. Pasi Sarolahti

Nowadays, the Internet suffers from several problems that are driving computer experts to develop the architect of innovative computer networks in the near future. Information Centric Networking(ICN) has been designed to address the Internet issues by replacing the current communication model, which is based on host names, with a communication model based on content names.
Content Centric Networking (CCN) is a clean slate future network architecture, and an exemplary model of ICN in shifting the current Internet transport paradigm by addressing content instead of host locations. In contrast to TCP that applies a flow-based end-to-end communication model between a pair of hosts, CCN content distribution session may involve multiple sources and multiple destinations not identified in advance, making traditional end-to-end approach to congestion control impossible. Thus, CCN nodes need to apply local measures to detect congestion in advance that can lead to relieve congestion. In IP, the efficiency of TCP relies on the rapid detection of packet loss through out-of-sequence packet delivery, however in CCN the packet sequence is not respected and an analogy of "triple duplicate ACK" to detect loss is not possible therefore, loss of data chunks because of buffer overflow is considered as a main indicator for congestion.
This thesis proposes a congestion control algorithm based on early detection of congested links in content-centric networks. The approach accelerates congestion recovery and can make dramatic decrease in the rate of packet drops and possible retransmissions. A neural network technique is used to realize this goal to avoid deterioration of network throughput. In various network scenarios, we demonstrate the advantage of early detection of congested links and provide a performance analysis using ndnSIM simulation environment. Our simulation results show that the proposed method is efficient and effective in controlling congestion in terms of applied performance metrics.

Keywords: Information-Centric Networking, Content-Centric Networking, Congestion Control Algorithms, Neural Networks

Acknowledgments

I would like to convey my sincere appreciation to my supervisor, *Professor Jörg Ott*, for his great support and encouragement throughout my master's thesis research. I have been amazingly fortunate to have an advisor who gave me the freedom to explore on my own, and at the same time the wisdom to finish this thesis.

I am also truly grateful and privileged to my instructor *Pasi Sarolahti* whose encouragement, supervision and support from the preliminary to the concluding level enabled me to develop an understanding of the subject.

I wish to express my gratitude to my family in my motherland, who encouraged me to continue my education; without them this thesis would not have been possible to finish.

And to my wife *Behnaz*, who has given me her unequivocal support for which my mere expression of thanks likewise does not suffice.

Mohammad Hovaidi Ardestani
Espoo, 01.07.2014

Contents

Abbreviations and Acronyms

ACK	Acknowledgement
AIMD	Additive Increase Multiplicative Decrease
ANN	Artificial Neural Networks
CCN	Content-Centric Networking
COMET	COntent Mediator architecture for contentaware nETworks
CS	Content Store
DTN	Delay Tolerant Networking
DONA	A Data-Oriented Network Architecture
ICMP	Internet Control Message Protocol
ICP	Interest Control Protocol
ICN	Information-Centric Networking
IP	Internet Protocol
LAN	Local Area Network
LMS	Least mean square
MAC	Media Access Control
MSS	Maximum Segment Size
NACK	Negative Acknowledgement
NDN	Named-Data Networking
NN	Neural Networks
PSIRP	Publish-Subscribe Internet Routing Paradigm
PURSUIT	Publish-Subscribe Internet Technology
RTT	Round Trip Time
SAIL	Scalable and Adaptive Internet soLutions
TCP	Transmission Control Protocol
TRIAD	Translating Relaying Internet Architecture integrating Active Directories
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VoIP	Voice over IP
WLAN	Wireless Local Area Network
WPAN	Wireless Personal Area Network
WSN	Wireless Sensor Network

List of Figures

List of Tables

1 Introduction

Today's Internet was designed and created to make connection to scarce and valuable mainframe computers in the 1960s and 70s. In this networking architecture, machines are uniquely located and identified with an Internet protocol address in order to establish a session between any pair of devices.
In the 50 years since the creation of packet switched networking, computers and their peripherals have become cheap and pervasive. Moreover, networks users are more willing to share information rather than their resources and are oblivious to locations of data providers [1]. In fact, although the focus of computer networking was resource sharing at the very beginning, the pressure of massive content delivery has changed data communication over the past decade.
Consequently, network use has dramatically evolved to be dominated by content distribution and retrieval, whilst the underlying infrastructure is still based on interconnection of hosts by means of their IP addresses. Accessing content and services requires naming methods including URL and URI, which tie the content to the Internet hosts.
Nowadays, the Internet suffers from several problems that are driving researchers to develop the innovative computer networks architecture for the future. Security issues are among the biggest imminent problems facing the Internet. Denial of service attack and spam are the most common threats which have tilted the balance of power in favour of attackers [2]. On the other hand, the current Internet was designed to make connections between fixed nodes. Thus, locating roaming mobile terminals for call delivery and maintaining their connections regardless of their points of attachment is another important issue to be solved. Besides, the number of IP addresses is limited and while IPv6 deployment is still under investigation, some parts of the world have already exhausted their IPv4 allocations [14]. In general, there has not been remarkable evolution in the Internet architecture for almost twenty years, in spite of a revolution in the use of Internet and it has been leading toward ossification [2].
The aforementioned problems are some of the main Internet issues that raise the question about reconsidering the design of this networking architecture. In order to address these problems and many other immediate issues, there is no clear consensus in the Internet research community. One side believes that the current Internet infrastructure should be improved gradually and defends an evolutionary approach which is backwards-compatible. The other side argues a clean-slate approach and investigates a revolutionary transformation to a next-generation Internet architecture. Among the revolutionary approaches, there has been considerable amount of carried out research in recent years to replace the current communication model, which is based on host names, with a communication model based on content names.
Information Centric Networking(ICN) [15] proposals have recently emerged to rethink Internet foundations and design a natively data-oriented networking environment. The main rationale behind the design of ICN is to focus on the network's main mechanisms on content("what")names instead of content locations("where"). Shifting from IP addresses to named data will tackle naturally most of the issues

faced by the current IP infrastructure in terms of mobility management, security, scalability, and content delivery. [3]
In the seminal paper of Gritter and Cheriton [16], the very first ideas about transition from location-based network paradigm to information-centric design were introduced about a decade ago in the context of TRIAD project at the Stanford University [15]. Since then, various ICN projects which focus on replacing the current communication model by a type of Content-centric network have been proposed.
ICN is currently being investigated by a number of research projects mainly in the US and Europe. Research projects conducted in the US are the DONA project, A Data-Oriented Network Architecture [17] at UC Berkeley, Named Data Networking (NDN) [24] and its predecessor Content Centric Networking (CCN) [25], and MobilityFirst [26]. *Hereafter in this thesis the terms Named Data Networking(NDN) and Content Centric Networking(CCN) will be used interchangeably.*
European projects include the EU funded projects like the PSIRP project Publish-Subscribe Internet Routing Paradigm [18] and its successor Publish-Subscribe Internet Technology (PURSUIT) [19], Scalable and Adaptive Internet soLutions (SAIL) [20] and its predecessor 4WARD [21], COMET [22], and CONVERGENCE [23], as well as the French funded project ANR Connect [24] which adopts the NDN architecture. At the same time, the Delay Tolerant Networking(DTN) [28] community has developed a message-oriented architecture that has been used along with information-centric addressing and routing concepts.
Amongst current plans in ICN, Content-Centric Networking (CCN) is a promising architecture proposed by PARC [1]. This approach is designed to deal with today's trend and proposes a new paradigm to address the aforementioned problems. It follows a receiver-based communication model, and introduces generalised caching in potentially every network device. In CCN, Data Packets is sent in reply to request called Interest Packet and Data chunks gets cached along the route back to the original requester. In simple words, its communication model is based on the idea of Publish/Subscribe [19] paradigm and focuses on content dissemination and retrieval.

1.1 Problem Statement

In [1] Jacobson outlined the layout of the overall CCN architecture. Nevertheless, many functionalities are still in the early stage and need to trigger an enormous amount of research. Traffic control mechanism is one of the most important design specifics that has not been studied in the context of ICN to significant extent. CCN as a networking architecture need to define concise methods to control traffic when multiple users contend for access to the same resources (bandwidth, buffers, and queues).
There are some similarities between traffic model in CCN and Publish/Subscribe paradigm [8] in which senders *publish* what they want to send and receivers interested in specific data *subscribe* to their desired publication. In CCN receivers send their requests for particular data content regardless of where senders are located. Although by decoupling senders from receivers in this model CCN has alleviated

problems with unsolicited traffic, engineering and shaping traffic have not been addressed effectively [7].
In [1] flow balance, one Interest packets retrieves at most one data packet, has been barely suggested to avoid congestion, while there is no further technique to manage congested traffic flow and saturated transmission buffers. Therefore, The definition of a suite of traffic control mechanisms adapted to CCN still lacks in the literature.

1.2 Objectives and Scope

Content-centric networking is still a very young research area and, as such, there are a number of weaknesses in existing designs. The approach discussed in this thesis is to focus on congestion issue in Information Centric Networking in a broader sense and propose a novel idea to tackle this problem. The main objective is to provide an intelligent algorithm to detect congestion in early phase of occurrence and to propose possible response decisions to realize congestion recovery. This approach is suggested for Content Centric Networking as a great representation of Information Centric Networking.

1.3 Contribution of the Thesis

This Thesis proposes a novel approach to control congestion in CCNs by early detection of congested links based on computed probability of data packet drops for next time instances. A Neural Network technique is used to realize this goal to avoid deterioration of network throughput.
In various network scenarios, we demonstrate the advantage of early detection of congested links and provide a performance analysis using ndnSIM simulation environment. Our simulation results show that the proposed method is efficient and effective in early congestion detection in terms of applied performance metrics. The technique also provides multiple options for CCN-routers to decide in respond to the impending congestion in advance. Moreover, the proposed technique can be considered as a general solution for congestion problem of ICNs in a more broad sense.

1.4 Outline

Chapter 2, provides a background overview of the Internet and CCN architecture as two packet-switched networks from two different generations of computer networks. chapter 3, presents a detailed description of taxonomy of congestion control algorithms and highlights the most appreciated one implemented in the current networks. It also provides recent related researches on congestion control algorithm in CCNs. Neural Networks overview and the rationale behind the proposed technique used in this thesis is described in the next chapter. This is followed by Chapter 5 which starts the solution part where the system that has been developed and a new congestion control algorithm using Neural Network are described elaborately. After the solutions part, testing and evaluation is explored in Chapter 6. Moreover, this section compares the suggested solution with related researches discussed in the third chapter. Finally, in the last chapter, conclusions about the work are made and observations on possible future challenges are projected.

2 Internet and CCN overview

This chapter serves as an introduction to the principles of the Internet architecture as the most prevalent Packet-switched network and Content-Centric Networking as a prospective candidate of Information-Centric Networking.
The first section presents an introductory description about the Internet architecture and begins with a brief overview of the Internet protocol suite layers as it relates to hosts. The next and more detailed part of this chapter describes CCN principles, compares its protocol stack with the TCP/IP, and highlights the main similarities and discrepancies between these networking structure. The rest of this chapter describes CCN communication model elaborately and discusses its related issues.

2.1 Internet overview

The name of Internet stems from its purpose of the "interconnection of computer networks". It is a connectionless universal system of packet-switched communication networks. An enormous number of services are implemented over the Internet, including e-mail, file transfer, remote computer control, web browsing, and video streaming.
The communications infrastructure of the Internet consists of its hardware components both in end systems (Internet hosts), and intermediate systems including, LAN switches, and routers. Moreover, the Internet runs a system of software layers called protocol stack that control various aspects of the architecture [44]. All hardware components in the Internet need to use the same communication protocol called standard Internet protocol suite (TCP/IP) and its main goal is to facilitate data transmission and exchange among several billion users worldwide.

2.1.1 Internet Hosts

A network host, or simply "host" is the ultimate consumer of communication services and it is called Internet host if it participates in the Internet. In networking jargon Internet hosts are also referred to as end systems and they span a wide range of size, speed, and function. An internet host has one or more network layer host address called IP address assigned to its network interfaces. It generally executes application programs on behalf of user(s) by applying Internet communication services for its support [46].

2.1.2 Routers

Internet hosts are usually connected to each other using switching devices known as routers rather than using single communication link. A router is a device that uses the address information (IP address) to forward data packets along networks. Figure 1 shows two end systems that communicate over the Internet via routers. The client-server communication model depicted in this figure is a prevalent model in which one host called client requests a service or resource and the other end system called server shares the requested information with its client.

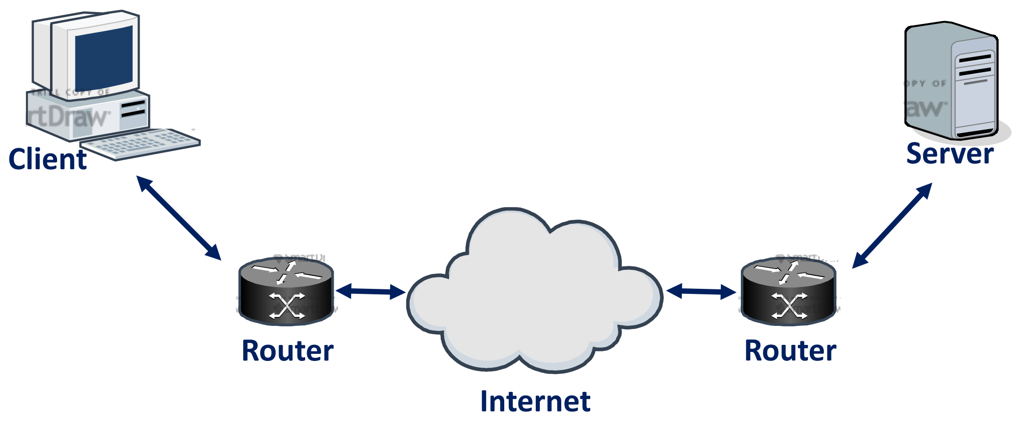


Figure 1: Communication between a service requester(Client) and the provider of a resource or service(Server) via Internet.

2.1.3 Internet Protocol suite

The communications process between end systems and intermediate systems is defined in terms of the TCP/IP. It creates heterogeneous network of communication nodes with various types of operating systems and hardware architecture. It provides end-to-end connectivity and defines how data should be formatted, addressed, transmitted, routed and received at the destination.
The TCP/IP functionality has been organized into four abstraction layers which are applied to characterize all related protocols according to the scope of networking involved. A layer does not define a single protocol, but it clusters several data communication functions which are performed by any number of protocols [47]. The idea behind layering is that each layer is responsible for providing a service to the layer above by using the services of the layer below. It is worth mentioning that, when Internet nodes send and receive data, the individual layers do not need to know how the layers above and beneath function; they only need to know how to pass data to them. The protocol stack layers used in the TCP/IP suite are as follows:

Application Layer

The application layer is the top layer of the Internet protocol suite. It consists of protocols that focus on process-to-process communication across an IP network and provides a firm communication interface and end-user services. All application processes use the service elements provided by the application layer. This is where the most common Internet user protocols such as SMTP, FTP, SSH, and HTTP operate [44] [46].

Transport Layer

The transport layer provides end-to-end communication services for applications. It is responsible for delivering data to the appropriate application process on the host computers and provides convenient services such as connection-oriented data stream support, reliability, flow control, and multiplexing [46]. The most well-known transport protocol is the Transmission Control Protocol(TCP) which is used for connection-oriented transmissions [44].

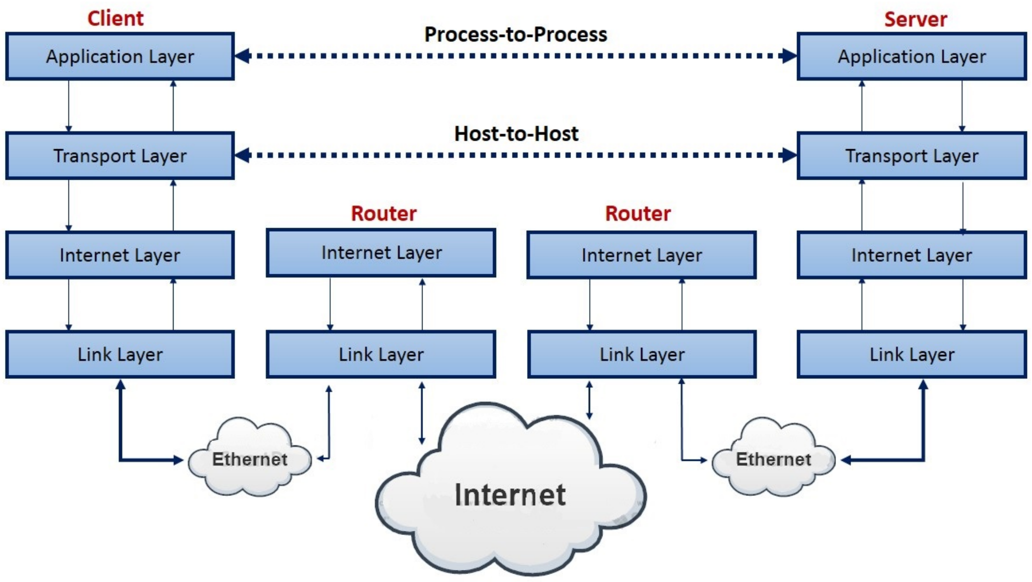


Figure 2: This figure projects data flow in the client-server model.

Network Layer

The network layer provides the functional and procedural means of transferring variable length data sequences from a source to a destination host via one or more networks, while maintaining the quality of service functions [46]. This layer defines the addressing and routing structures used for the TCP/IP protocol suite. All Internet transport protocols use the Internet Protocol(IP) which defines IP addresses to carry data from source host to destination host [44].

Link Layer

As it can be observed in the figure 2 the link layer is the lowest layer in the Internet Protocol Suite and it provides communication on its directly-connected network. This is a descriptive realm of networking protocols that operate only on the local network segment(link) which provides communication among hosts without intervening routers.

2.2 Content-Centric Networking

Content-Centric Networking [1] is a clean slate future network infrastructure that introduces a new Internet architecture. The main idea in CCN is to change the current internet transport paradigm by addressing content instead of host locations. In stark contrast to TCP/IP in which communication session is established based on host addresses, sessions in CCN are concerned with content names. In other words, location addresses are not integrated with a CCN packet and it merely addresses hierarchical human readable names of data packet over the standard form of URI.

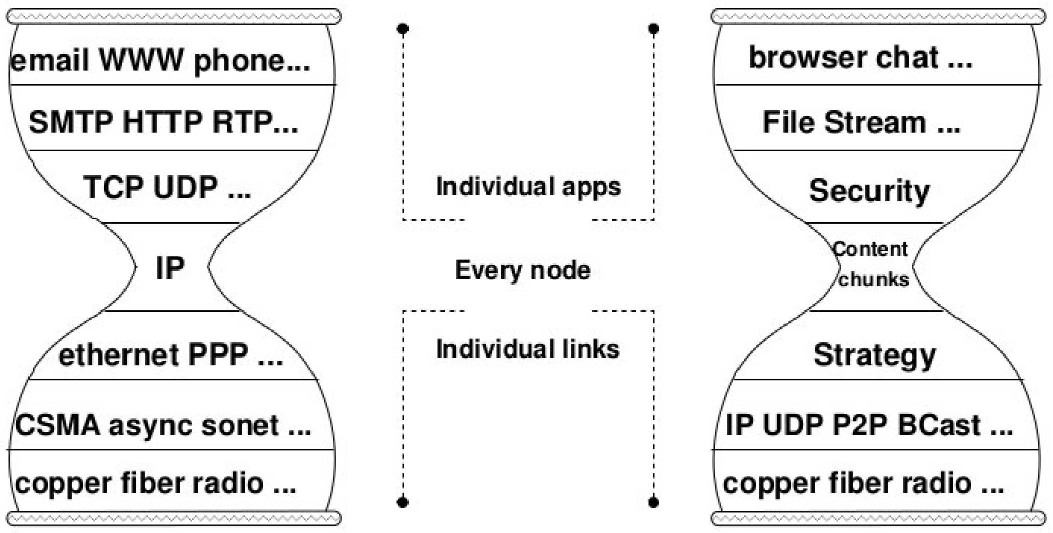


Figure 3: Protocol Stack,TCP/IP Vs. CCN. *Adopted from* [1].

Figure 3 shows similarities and discrepancies between the IP and CCN protocol stacks. [1] It can be observed that most layers of the TCP/IP stack imply mutual agreements; e.g., link layer framing protocol is designed for reliable and efficient communication between two adjacent machines; transport layer handle communication between data requester and provider. The network layer is the only one that requires universal agreement. CCN's network layer struggles to take advantages of IP layer attractive properties and in the meanwhile, alleviate number of demands on layer two.
On the other hand, there are number of substantial discrepancies to discriminate CCN from TCP/IP stack. CCN shifts the main focus from IP addresses to content names, and introduces the strategy and security layers which deal with data dissemination and securing data itself rather than the connection path respectively.

Strategy layer can make multi-homing more feasible and security layer makes network more immune to host-based vulnerabilities.

2.3 CCN Node Model

CCN communication is driven by the consumers of data. There are two type of messages exchanged between network nodes: Interest and Data. A consumer asks for a content by disseminating an Interest packet to the network over the all available interfaces. Any node receiving the Interest and having the desired data replies with a Data packet. The data packet is transmitted only in response to an Interest and consumes that Interest and nodes which are not either consumer or provider just forward the Interest message on the overlay network [1].

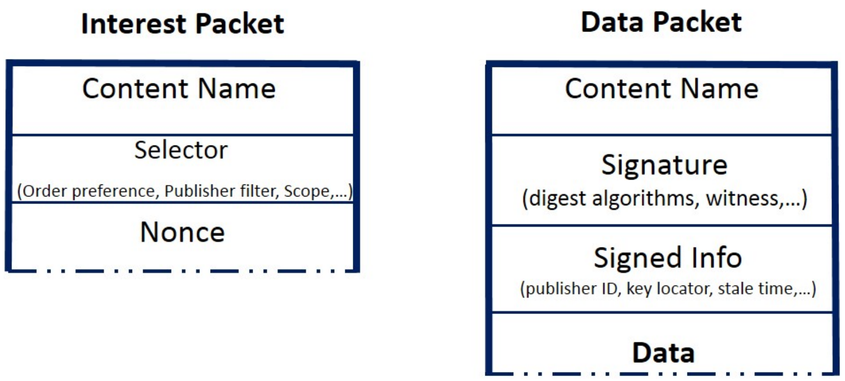

Interest Packet
Content Name
Selector
(Order preference, Publisher filter, Scope,...)
Nonce
Data Packet
Content Name
Signature
(digest algorithms, witness,...)
Signed Info
(publisher ID, key locator, stale time,...)
Data

Figure 4: CCN packet types [1].

In order to have content centric concept realized each node needs to be equipped with three main data structures and generalized concept of interface as follows.

2.3.1 Content Store(CS)

Content Store plays the same role in CCN node as a buffer memory in IP routers, pursuing the goal of avoiding the need to repeatedly fetch popular contents. The main difference between CCN router and IP router is that, the latter aims to discard packets upon forwarding them to the next hop router, but the former keeps the most popular data for the longest possible time. Within a Content Store data chunks are stored and replaced according to a specific policy e.g, Least Recently Used (LRU) or Least Frequently Used (LFU). Since in CCN each packet does not belong to point to point conversation and is potentially useful to many hosts, arriving data need to be conserved in content store as long as possible [1].

2.3.2 Pending Interest Table(PIT)

PIT maintains a track of forwarded Interest packets so that the returned Data packet can be sent to its requester(s). As Figure 4 [12] shows it contains a list of incoming interfaces from which the Interest packets for that name have been received, and a list of outgoing interfaces to which the Interest has been forwarded as well. In contrast to IP network in which packets are routed in both directions toward server(s) and client(s), in CCN only Interest packets need to be routed. When a requester diffuses its Interest packets toward potential data provider(s), a trail of nodes footprint for a desired data will be left in the propagation path in order to define the reverse path to that requester.

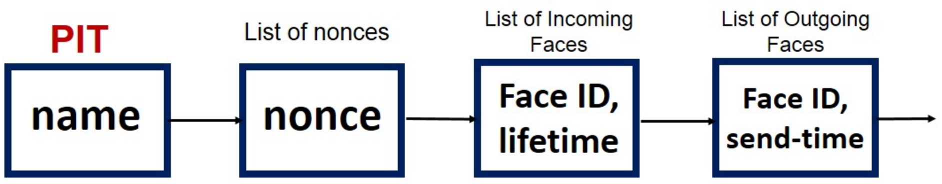


Figure 5: Forwarding State in PIT [12].

It is obvious that by applying ”bread crumb” technique data cannot loop in CCN, however Interest packets are susceptible to loop and duplication. Therefore, in order to prevent this a list of random nonce value has been added to PIT to detect and discard duplicates received over different paths [1].

2.3.3 Forwarding Information Base(FIB)

In IP networks the information necessary to forward IP Datagrams should contain at least the interface identifier and the next hop information for each reachable destination network. This information is gathered in an IP FIB table and differs from routing table which stores all routing information received from routing peers [48]. The CCN FIB is virtually similar to the IP FIB and is a table of outbound faces for Interest packets toward potential data provider(s) of matching data. The only difference between the CCN FIB and the IP FIB is the fact that each prefix entry in the CCN FIB may point to a list of faces rather than only one [38].

2.3.4 Face

A connection to a network or directly to an application party in CCN is called Face. In case of connecting to a network, it is similar to Interface of IP networks . All messages in CCN node are received through a face and will be disseminate through a face. A face can be arranged to send and receive broadcast or multicast packets on a particular network interface, or to send and receive packets using point-to-point addressing in the underlying transport, or even it can be the connection to a single application process running on the same machine [38].

2.4 CCN Transport Model

One of the most important weaknesses points in the IP network is that a host is empowered to send any packet and network is responsible of delivering it to the destination. It simply means that when a node sends out a data packet, network tries its best to deliver that data to its receiver(s) regardless of the fact that receiver(s) might not interested to that particular data. (D)DOS attack and spam are two of the side effects of this issue.
CCN is a receiver-driven data centric network. Its Receiver-driven feature is similar to the Publish/Subscribe model in which every node interested in a specific data needs to subscribe and publisher sends back the requested data to the subscribers. It simply means that CCN Decouples senders form receivers and tackles the aforementioned problem which is an important issue in TCP/IP. Data Centric feature on the other hand, makes CCN interest packets independent of resources addresses due to caching data in every node.

2.4.1 Interest and Data processing

Figure 6 depicts the algorithms used in CCN nodes to process Interest and Data packets. When a CCN router receives an Interest Packet, a longest-match lookup is done on its Content Name in the Content Store. Since there is a chance that the desired Content packet has been already cached in this particular node, checking the CS is the first priority for the arriving Interest Packets.

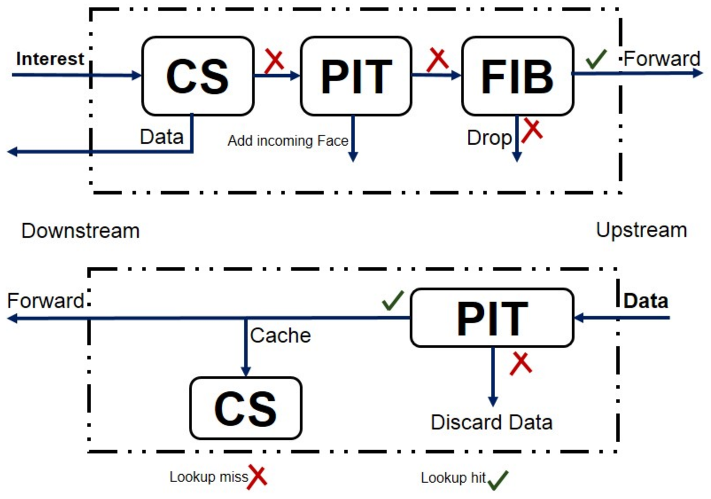


Figure 6: Interest and Data processing in CCN [12].

If there is a Data packet that matches the Interest, it will be sent back on the face that has received interest and finally the satisfied Interest will be discarded.
When the content data cannot be found in the CS, then the second priority is to lookup in the PIT. If the Interest Packet name does exactly match with a PIT entry the receiver face will be added to the PIT Incoming Faces list and the Interest will be discarded.This procedure is for making sure that all the arrival faces of the same Interest will send back the matched Content Data [1].
Beside the Content Name, a random nonce value generated by the consumer is carried in each Interest packet as well. A router keeps record of both the name and nonce of each arrived Interest, so it can recognize whether a newly received Interest is indeed a new one or an old one that looped back [12].
The last step in forwarding Interest Packets is to lookup in the FIB entries. If an Interest Packet matches with an FIB entry then the Interest needs to be sent upstream towards the data. Otherwise, the Interest will be discarded because this node does not have any matching data and even does not have any clue how to find any [1].
As mentioned earlier, Data Packet is not routed but simply follows the trail of PIT entries as footprints to reach the original requester(s). A longest-match lookup of a Data name is done upon arrival of a Data Packet in PIT entries. PIT match proves that the Data was solicited by Interest(s)sent by this router, while mismatched data means Data packet is unsolicited and discarded. When the router finds matched Data name in PIT, sends the Data packet to the faces from which the Interest was received, caches the data, and finally removes the PIT entry [12].

2.5 Summary

In this chapter basic concepts of Internet architecture and its layered protocol suit was provided at the first section. Content Centric Networking system description was introduced in the next part, followed by CCN transport model. Interest and Data processing was the last section of this chapter to cover all the fundamental definitions needed for discussing congestion control algorithms in the next chapter.

3 Congestion Control Algorithms

Congestion in packet switching networks is a state in which the network throughput deteriorates when network resources including communication links, processors cycles and memory buffers are saturated. Generally, congestion happens when there are not sufficient resources to answer the demand, but surprisingly, increasing number of resources, e.g. larger buffer space, high speed links, and shorter packet processing time cannot address this problem [29].
Nagle [31] proves that networks with infinite-buffer routers are as vulnerable to congestion as networks with normal buffer space switches. It can be seen in figure 7.a, that too much traffic load will be lead to the buffer overflow and packet drops. On the other hand, large buffer space as shown in figure 7.b causes long queue and more delay and consequently when packets processing is performed majority of them have already been timed out [30].

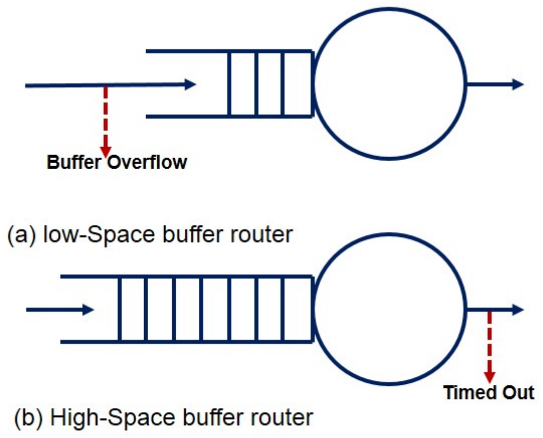


Figure 7: Low-space memory in intermediate nodes is as disadvantageous as High-Space memory [30].

High speed links in the network not only address the problem, but also may exacerbate the situation. Since configuring a network with balanced link speeds and processors is almost impossible, then there is always possibility of congestion due to different link speeds and processing times. Figure 8 shows that if by any chance a network has truly balanced configuration with the same speed for any link and router, it is still susceptible to the congestion due to bandwidth bottleneck problem [30].
A bottleneck generally causes a system performance to be reduced or slowed down due to limited resources or components. Consequently, a bandwidth bottleneck occurs when there is not sufficient bandwidth available to ensure that all data packets in the network reach their destination in a timely fashion. This can be observed in figure 8 that, although all links have the 10Mbps speed, they deliver a higher

volume of data than what is supported by the existing routers. Since the traffic load incoming by links attached to the routers are twice as the bottleneck link capacity then consequently, the congestion occurs and performance of the entire network will be compromised.
Considering all mentioned arguments, it is sensible to conclude that congestion is a dynamic problem and therefore static solutions cannot tackle the problem perfectly. Thus, any new control mechanism need to be designed to deal with the congestion issue in the time of occurrence [30].

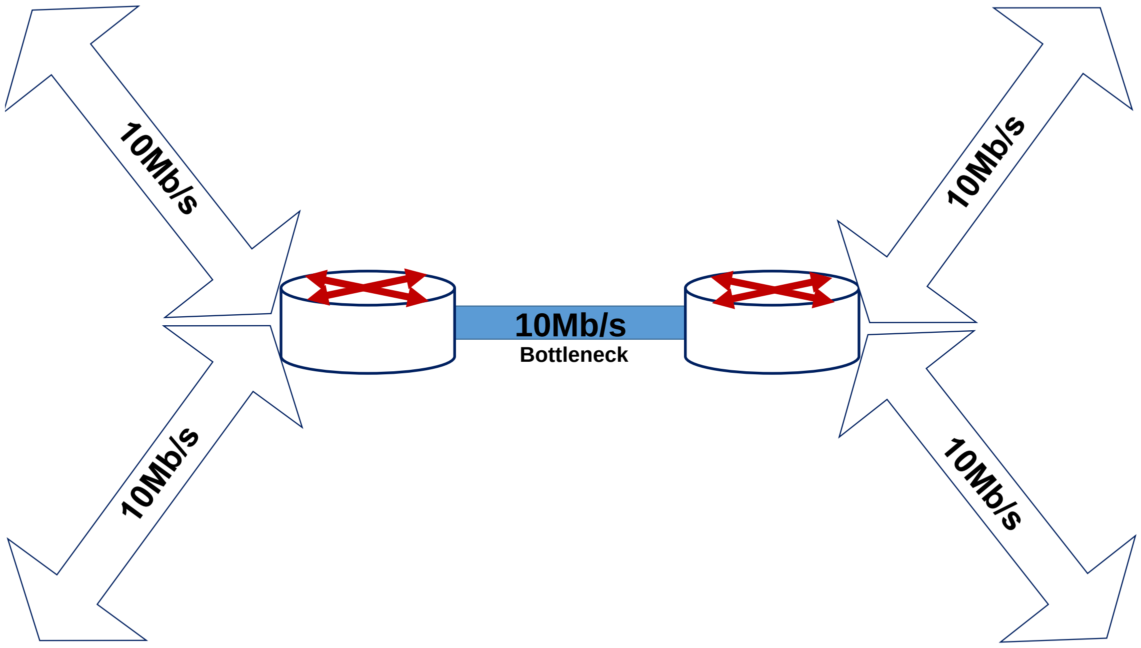


Figure 8: Bottleneck problem in balanced network

3.1 A taxonomy for Congestion Control algorithms

Due to explosive growth of bandwidth and network traffic load in recent years, packet switched network congestion as a resource sharing problem, has been posing a serious threat to these networks and the Internet in particular. Consequently, congestion control became one of the major fields of research for computer networks experts. This section categorizes different congestion control strategies proposed for location based networks and highlights the most admired techniques implemented in the Internet.

3.1.1 Open Loop Control

If a packet arrives at a node with a finite buffer faster than they can be processed, the overflow will occur unavoidably. A sudden increase of traffic load can be one

of the main causes for congestion issue in packet switching networks. Therefore, a congestion control algorithm needs to be designed to constrain bursts of source traffic and to avoid unnecessary packet losses at an intermediate access nodes and within the network.
Open loop congestion control algorithms are mainly designed to tackle this issue by controlling the rate of packet transmission of senders and adopting appropriate regulations to accept or discard packets on receivers. [29]Thus, they do not make decision based on feedback from congested links. These algorithms control flow and congestion with local information about buffer space and bandwidth. This scheme itself is divided into two subcategories of *source control* and *destination control* algorithms as follows:

Source Control

Algorithms in this scheme apply traffic control at the source end and take advantage of local knowledge of the network [29]. The bit round fair queuing method [34], and the input buffer model [35] are two instances of these algorithms. For instance the latter algorithm, controls traffic load by imposing a constraint on the fraction of buffers for input traffic in a node's buffer space.

Destination Control

Algorithms accompanying with this group can be identified by operations on the destinations end with no knowledge of feedback. Algorithms in which packets are simply discarded due to buffer overflow, fall into this category [29]. Since algorithms in this category focus on discarding packets, consequently, they cannot be very effective in decreasing number of packet drops.

3.1.2 Closed Loop Control

Closed loop category is based on monitoring the system to detect congestion, passing this information to where action can be taken, and adjusting system operation to address the problem. The majority of congestion control algorithms are in this class in which decision is based on feedback. This feedback can be sent either globally all the way from source to destination or locally from intermediate nodes [29]. If the feedback is based on end-to-end behaviour analysis without any explicit signal from within the network, it is called implicit feedback. Packet loss due to buffer overflow is the most important instance of this binary feedback. While, explicit feedback is sent in a separate message or piggybacked via a signal bit in packet headers [54]. There are other features to categorize congestion control in closed loop schemes, but these mentioned ones can be sufficed for the scope of this thesis.

Explicit Feedback

Approaches in this category are called network assisted congestion control. Routers as network-layer components provide senders information about congestion state by sending explicit feedback. The feedback can be either as simple as one bit manifesting congested link or messages about capacity of outgoing links for supporting retransmission packets [44]. Explicit feedback congestion control algorithms are mostly designed to avoid congestion and lead the network to the optimal operating point by controlling traffic admission [29]. Adaptive admission control [32], rate-based congestion avoidance [33], and Explicit congestion notification [36] are the most tangible examples of this category.

Implicit Feedback

Congestion control approaches under this category are also known as end-to-end algorithms. The key component of these approaches is that the endpoint nodes are responsible for controlling the data rate. In this model, hosts take advantage of implicit knowledge obtained from the network or globally between source and destination to tune the transmission rate, and time of transmission as well. There is no additional message or explicit notification in these algorithms. Time out alert and missed acknowledgements are mostly used in these algorithms to indicate congested network. Congestion avoidance and control of Jacobson [41], the most appreciated congestion control technique, and timeout-based congestion control scheme [42] are magnificent instances of algorithms in this class.

3.2 Standard TCP Congestion Control Algorithms

Given the taxonomy of congestion control algorithms, this section is in the right position to consider the details of the most acclaimed TCP congestion control algorithms developed by Jacobson which is described in [41], and [43], and standardized in [45]. The algorithm has three main components known as *slow start*, *congestion avoidance*, and *fast recovery*. TCP senders are required to have slow start, and congestion avoidance, while fast recovery is only recommended. The discrepancy between obligatory components is rooted from the way they increase the size of *congestion Window(cwnd)* in response to received ACKs.
The *congestion Window* imposes a constraint on the amount of data that a TCP sender can inject into the network before receiving an acknowledgement. The *slow start threshold (ssthresh)* is another state variable which is used to regulate data transmission by altering between the slow start and congestion avoidance algorithm [45].

3.2.1 Slow Start

This technique is known as sender-based Flow control due to controlling the transmission rate by the sender. The mechanism used by the sender is based on the rate

of acknowledgements returned from the destination node. In other words, the rate at which the sender is allowed to transmit is determined by the rate of acknowledgements returned from the receiver.
When a TCP connection first begins, the slow start algorithm initializes a cwnd to a small value of one MSS. TCP sends the first segment into the network and then waits for the acknowledgement. When Ack is returned by the receiver, the cwnd increases by one segment and sends out two maximum-sized segments. This process results in doubling of the sending rate every Round Trip Time(RTT). Since the transmission rate grows exponentially, slow start is not slow indeed. On the other hand, it is initialized with the least possible value and starts from the one segment size. Thus, the TCP sending rate grows fast, but starts slow during the slow start phase [44].
In accordance with notifications that TCP sender receives, slow start provides several options to stop the exponential growth of cwnd. First, if there is a loss event notified by a timeout, cwnd will be set to 1 by the TCP sender and it begins the whole process again. It also needs to set the value of ssthresh to half of the congestion window value of the time when congestion was last detected. The second way of ending the exponential growth is related to ssthresh. When cwnd equals to ssthresh, slow start ends and TCP transitions into congestion avoidance mode which will be explained in the next section. The last option to cease this process is based on detecting three duplicate acknowledgements. Once ACKs are detected, TCP enters the fast recovery state as discussed below.

3.2.2 Congestion Avoidance

As figure 9 illustrates, congestion avoidance is one of the choices with which slow start phase can be terminated. The slow start algorithm is used when cwnd is less than ssthresh, while the congestion avoidance algorithm is used in reverse order. Sender opts between slow start or congestion avoidance when cwnd and ssthresh are the same [45].
The value of cwnd at the beginning of this state is approximately half its value when congestion was last occurred and during congestion avoidance, cwnd is incremented by roughly one single MSS every RTT. TCP's congestion avoidance algorithms behaves the same until congestion is detected. The end of this state is similar to the case of slow start in which cwnd is set to one segment and the value of ssthresh is updated to half of the value when congestion was last encountered [44].

3.2.3 Fast Recovery

In order to inform the sender about receiving an out-of-order segment, a TCP receiver should send an immediate duplicate ACK when an out-of-order segment arrives. This ACK informs also the sender about the sequence number that is expected [45]. The TCP sender should use the "fast retransmit" algorithm to detect and repair loss, based on incoming duplicate ACKs. The fast retransmit algorithm uses the arrival of 3 duplicate ACKs as an indication that a segment has been lost. After the fast retransmit algorithm sends what appears to be the missing segment,

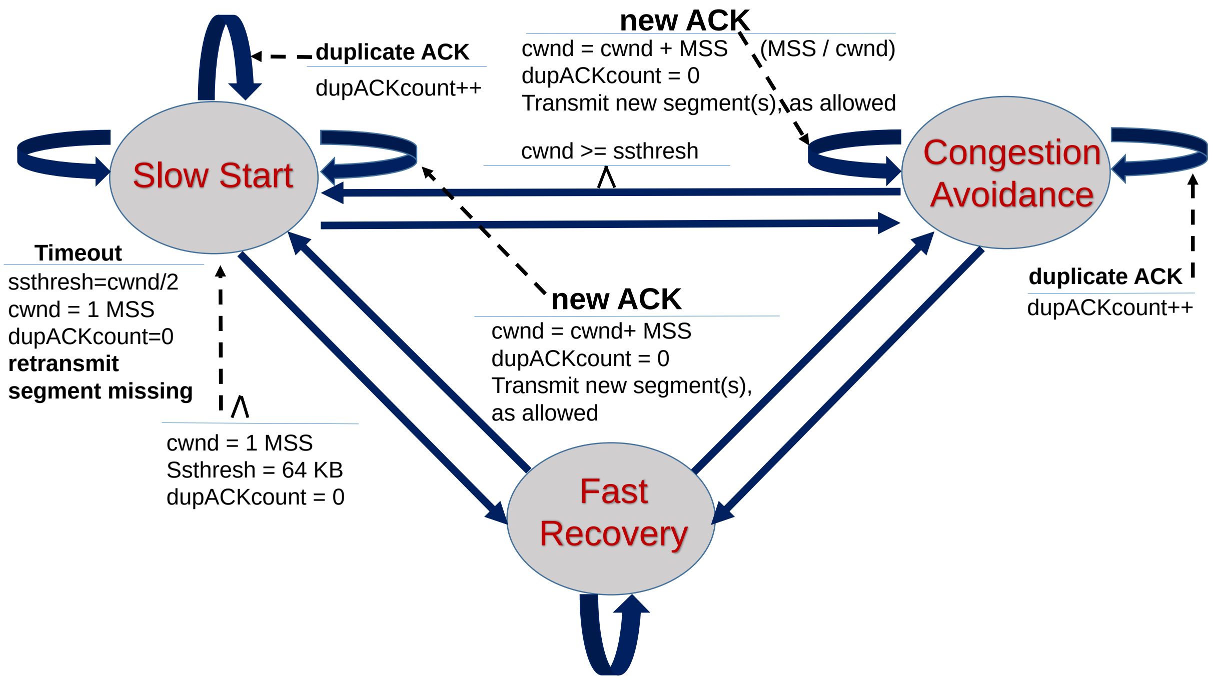


Figure 9: Finite-State machine description of TCP congestion control

the ”fast recovery” algorithm regulates the transmission of new data until a non-duplicate ACK arrives. At the end of this phase the value of cwnd is set to 1 MSS and the value of sstresh is updated to half the value of cwnd when the loss event occurred. It is worth mentioning that fast recovery in TCP congestion control is recommended and at early version of TCP, known as TCP Tahoe, it was not incorporated [44].
Figure 9 projects the complete FSM description of TCP's congestion control algorithms explained above and it indicates where transmission of new segments or retransmission can happen.

3.3 Congestion Control in CCN

In contrast to TCP that applies a flow-based end-to-end communication model between a pair of hosts, CCN content distribution session may involve multiple nodes not identified in advance, making traditional end-to-end approach to congestion control impossible. In IP, the efficiency of TCP relies on the rapid detection of packet loss through out-of-sequence packet delivery. In CCN, where flows can have multiple sources and multiple destinations, the packet sequence is not respected and therefore an analogy of *triple duplicate ACK* for detecting a loss event would not be feasible. On the other hand, delay based congestion notification is not applicable due to the fact that RTT values are variable. Thus, CCN nodes need to apply local measures

to detect congestion in advance that can lead to relieve congestion. In CCN loss of data chunks because of buffer overflow is the main indicator for congestion. Figure 10 outlines the congestion issue in CCNs and concludes to the first step for addressing this problem.
In [1] flow balance, one Interest packets retrieves at most one data packet, has been barely suggested to avoid congestion, while there is no further technique to manage congested traffic flow and saturated transmission buffers.
It is believed that congestion increases packet losses and consequently degrades performance in all packet switched networks. Congestion recovery mechanisms in these networks have been being studied with high priority for decades. Regarding all above, CCN as one of the possible internet infrastructure in the future needs to address this issue in advance.

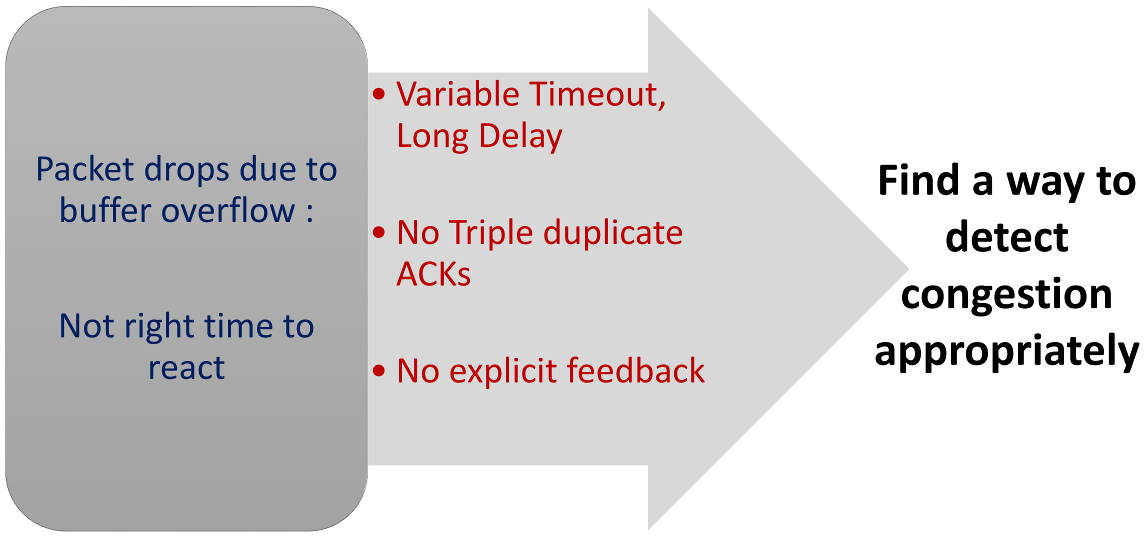

Packet drops due to buffer overflow :
Not right time to react
• Variable Timeout, Long Delay
• No Triple duplicate ACKs
• No explicit feedback
Find a way to detect congestion appropriately

Figure 10: Outline of congestion issue in CCN

3.4 Related work

After delving into the details of CCN architecture in the last chapter, this section is in the right position to introduce some important ongoing CCN research activities studying the congestion control mechanisms. To the best of our knowledge, following algorithms are the main congestion control mechanisms proposed for CCN to date.
Carofiglio et al in [9] designed a window-based Interest flow control protocol driven by an AIMD called Interest Control Protocol. The main idea is to guarantee reliable data transfers by re-expressing Interest packets in case of Data packet losses. ICP

does not rely on losses as congestion notifications, but on delay measurements and timer expirations mechanism to regulate the Interest rate at the receiver. ICP is coupled with a hop-by-hop Interest shaping mechanism in [10] to realize a per-flow Interest rate control at the output interfaces of every CCN router. In every output interface, there is one virtual queue per flow, identified by the name of the content. They have associated each virtual queue with a credit counter initialized to a maximum Data bytes the flow is granted to transmit, with no additional delay. The counter is incremented at the estimated fair rate of the corresponding downlink and decremented by forwarded Interest packets. Since deterministic window decreases are sensitive to delay estimation errors, they have recently introduced an extended initial design of ICP with a probabilistic window decrease mechanism aimed at controlling at the receiver bottleneck queuing delays [11]. The common theme in these evolutionary research is to detect congestion with a simple adaptive threshold set to RTT estimation.
In [7] Rozhnova et al present hop-by-hop Interest shaping mechanism which monitors the transmission buffers of a CCN router to compute the Interest packets rate that have produced the associated Chunks filling these interfaces. The idea is heavily based on the TCP congestion Control algorithm, however the proposed congestion control scheme prefers hop-by-hop Interest shaping instead of end-to-end mechanism. An alternative to relying on time outs would be to explicitly signal packet loss. Oueslati et al in [13] propose an explicit congestion signal by truncating data packet as following. When Deficit Round Robin(DRR) scheduler should discard a packet from the flow with the longest backlog, only the packet payload is rejected rather than refusing the entire packet. The header, including the name, is modified to signal the discard. The end-system identifies the loss immediately and can rapidly retransmit the corresponding Interest and react accordingly to the congestion signal. Interest NACK has been proposed in [12] by Cheng et al. Interest NACK can be categorized under the title of explicit feedback congestion control mechanism. The reason for this classification is the definition of NACK itself as following. When a CCN node can neither satisfy nor forward an Interest due to congested links, it sends an Interest NACK back to the downstream node. A NACK carries the same name as the corresponding Interest packet. If the downstream node has exhausted all its own forwarding options, it will send a NACK further downstream.
To summarize, there are two main parts for any congestion control algorithm suggested for CCN so far. The first part deals with detecting congestion in an effective way, while the second one attempts to address the whole problem by proposing novel solutions. These solutions can be roughly categorized as either adaptive transmission rate or adaptive forwarding. It means that, some algorithms control congestion issue by adjusting the traffic load, while others adopt approaches in which finding the least congested route would be their ultimate goal.

3.5 Summary

In this chapter we explained the concept of packet network congestion, and also summarized the studies of congestion control algorithms both in traditional TCP/IP transport paradigm and in CCN as a candidate of future Internet infrastructure. Since it is a vast topic, it was not feasible and even not sensible to cover the whole subject in this literature. Thus, we attempted to first, arrange a comprehensive taxonomy of proposed algorithms in the research community and then provided some inspiring examples under every single category.

The next topic of the section was devoted to describe elaborately, one of the most notable TCP congestion control algorithms introduced by Jacobson in [41]. The last section, focused on congestion issue in CCNs, and also provided an outline of most noticeable related researches carried out in the community.

4 Neural Networks

In computer science a computational process of discovering correlations and patterns in large amount of data sets is called data mining. It adopts statistical and knowledge-based approaches to analyse data sets. The ultimate goal of the data mining is to transform extracted patterns from a data sets into a comprehensible structure and consequently gain insight into a large collection of data [37]. Artificial Neural Network(ANN) is one of the techniques used in data mining to find non-linear relationships in data without pre-defined model.
This chapter first takes a brief look at the main concept of ANN and its resemblance to a biological Neural network. In order to have a big picture of ANN, neural network architecture and different learning paradigms are outlined in the subsequent sections as well.

4.1 Artificial and Biological Neural Network

The computer science concept of an artificial neural networks is heavily based on a biological neural network specifically the human brain.

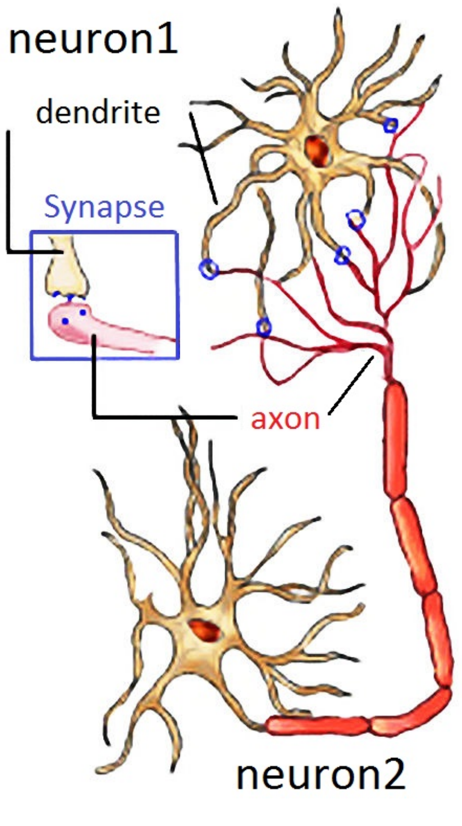


Figure 11: The Connection between Neurons [39].

A biological neural network is a series of interconnected nerve cells called neurons presented in the figure 11. They process and transmit information through electri-

cal and chemical signals. The relevant components of a neuron cell are dendrites, synapses, a cell body and an axon. Neurons interconnection is provided by synapses which are structures that permit neurons pass signals to other cells conducted by the axon. These synapses or connecting links have weights which vary in the degree to their influences on the neurons into which they feed [40].
The main feature of *neural network* is the ability of learning. When a neuron receives enough electric pulses through its dendrites, it activates and fires a pulse through its axon which connects neurons and propagates information. The synaptic connections alter throughout the lifetime of a neuron and the amount of incoming pulses needed to activate a neuron threshold vary as well. This behaviour allows the NN to learn.
Similarly, an artificial neural network depicted in figure 12 consists of highly interconnected processing units to model the human brain in performing a particular task or a process of learning. These processing units are called nodes and their connections have weight similar to synaptic weights to represent how much the output of one node can have effect on the behaviour of the node to which it serves as an input. ANN also has a bias value that allows shifting the neuron threshold, the point of depolarization at which the neuron fires. Since the power of the neural networks comes from the adjustment of the thresholds and weights of each node's input, these values are critical for successful learning[40].

4.2 Models of Neurons

The block diagram of figure 12 graphically illustrates the model of a neuron which performs typically non-linear and analogue computations.

In order to calculate the output value of a neuron, a weighted sum of the node inputs, and a bias term need to be added. This yields the linear output of the node:

$$u_k = \sum_{j=1}^{m} w_{kj}.x_j \tag{1}$$

and

$$y_k = \varphi(u_k + b_k) \tag{2}$$

where u_k is the linear combiner output; x_1,x_2,...,x_m are the input signals to the node which can either be the actual input to the neural network system (hereafter the term *neural network* will refer to as *Artificial Neural Network*), or the output from other neurons in preceding layers, w_{k1},w_{k2},...,w_{km} are the synaptic weights of neuron k; m is the total number of inputs to neuron k; b_k is the *bias*; y_k is the output signal of neuron; and $\varphi(.)$ is a function which defines output of the neuron given sets of inputs, referred to as the activation function [40].

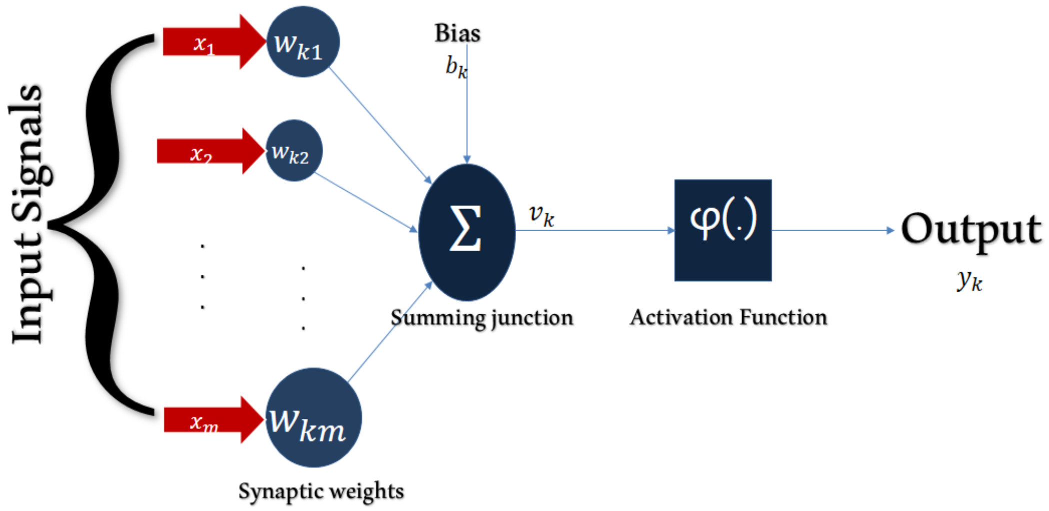


Figure 12: Nonlinear model of a neuron [40]

The behaviour of an ANN (Artificial Neural Network) depends on both the weights and the input-output function (*Activation Function*). The activation function $\varphi(.)$ can have many forms. This function typically falls into one of three categories illustrated in figure 13. In the *linear or ramp* function the output activity is proportional to the total weighted output, while in the *threshold or step-function* the output is set at one of two levels, depending on whether the total input is greater than or less than some threshold value. The *log-sigmoid or logistic function* is the most useful for training data and its output varies continuously but not linearly as the input changes. It bears a great resemblance to real neurones, however all three must be considered rough approximations of transforming the activation level of a unit(neuron) into an output signal.

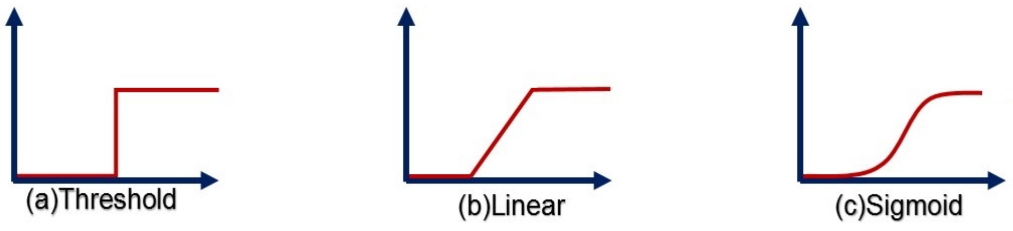


Figure 13: Activation Functions [40]

.

The role of the bias value is to have effect of applying an affine transformation to the output u_k of the linear combiner. As shown in Figure 12, each synaptic connection has a weight and each input is multiplied with its related weight. The weighted inputs are summed together with an externally applied *bias* to give the *induced local field* as formulated in the next equation.

$$v_k = u_k + b_k \tag{3}$$

In particular, depending on whether the bias is positive or negative, the relationship between the induced local field of neuron k and linear combiner's output is modified respectively as depicted in figure 14.

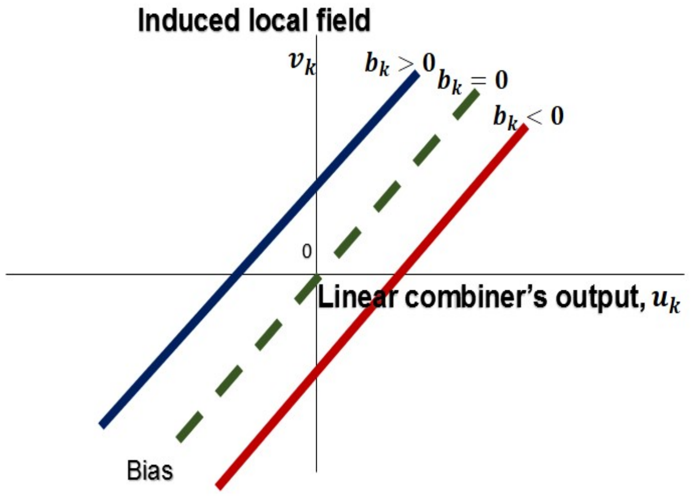


Figure 14: Affine Transformation produced by the presence of a bias [40].

At the very last step, an *activation function* is applied to the *induced local field* of the neuron in order to give the output of y_k of neuron k. This concept can be explained in mathematical term as follows:

$$y_k = \varphi(v_k) \tag{4}$$

4.3 Network Architecture

The neurons can be clustered together in many ways. Basically, a neural network is the grouping of neurons into layers. The way that these layers affect each other and the direction in which signals travel, is referred to *neural network architecture*. In order to train a *neural network*, *learning algorithms* and *network architecture* should be linked together closely.
Since the focus of this chapter is on providing basic knowledge of neural network techniques adopted in this thesis, it would be sensible to describe only the acyclic layered neural network architectures. They have one-way connections from input to output layers and are mostly used for prediction, pattern recognition, and non linear function fitting.

Single-Layer Feedforward Networks

The simplest form of layered neural network is a *single-layer feedforward* network in which there is an *input layer* of source nodes and one *output layer* of neurons. As it can be seen in the figure 15, *single layer* refers to the output layer of computation nodes. In this *neural network* the inputs are fed directly to the outputs via a series of weights. The sum of the products of the weights and the inputs is calculated in each node according to equation 1 described in the previous section, and based on the related *activation function* the neuron fires and takes either the activated value (commonly 1) or deactivated value (commonly -1).
The lines between the nodes indicate the flow of information from one node to the next. Since in this particular type of neural network, the information flows only from the input to the output (from left to right in the picture), it is called an *acyclic feedforward* type [40] .

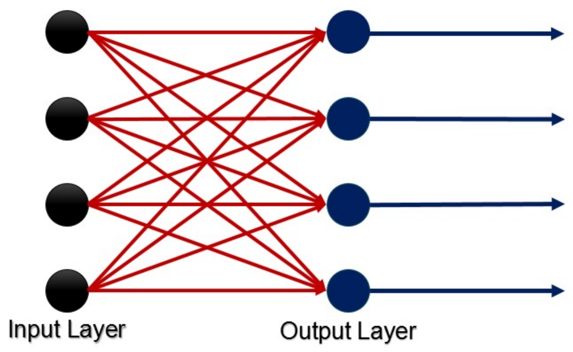


Figure 15: A single layer feedforward network with one-way direction of information flow [40].

Multilayer Feedforward Networks

A typical *multilayer feedforward* neural network depicted in figure 16 is the most popular in the classification [50]. Since every node in each layer of the network is connected to every other node in the forwarding layer, this network architecture is called fully interconnected structure.
As it can be observed in this picture, there is one distinguishable *hidden layer* with corresponding *hidden neurons* as computation nodes. A hidden layer is used to increase the expressiveness of the network.

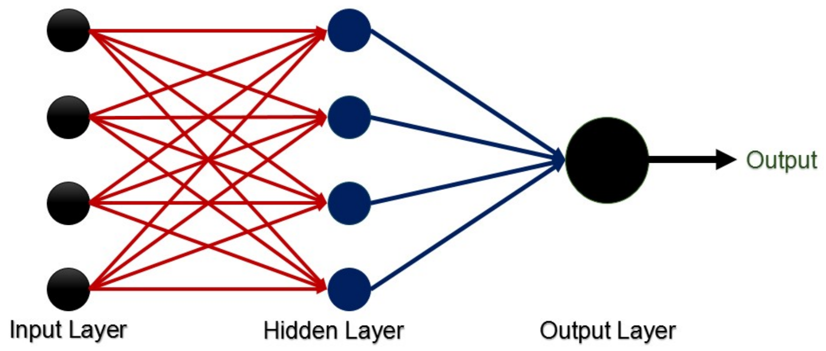


Figure 16: A multilayer feedforward network. This is the most common structure for neural networks [40].

A network with one or more *hidden layers* is enabled to extract higher-order statistics [40]. This ability is particularly valuable when the size of an *input layer* of a network is large. The hidden layer is usually about 10 percent the size of the input layer. For instance, in the case of target detection, the output layer only needs a single node. The output of this node is applied to threshold activation function to provide a positive or negative indication of the target's presence or absence in the input data.
In this particular *neural network* the source nodes in input layer of network supply the input signal to neurons in the next layer which is the first hidden layer. The output signals of this layer are fed to its next layer which is the second hidden layer and so on. The set of output signals of the neurons in the output layer of network are considered as the overall response of a network to the activation function supplied by source nodes of the input(first) layer [51].
The network illustrated in Figure 16 is referred to as a 4-4-1 network because it has 4 source nodes, 4 hidden neurons, and 1 output neuron. Generally, a feedforward network with m source nodes, h_1 neurons in the first hidden layer, h_2 neurons in the second hidden layer, and q neurons in the output layer is referred to as an $m - h_1 - h_2 - q$ network.

4.4 Learning Paradigms

The most significant property of a neural network is the ability of learning and improving its performance through learning. The interactive process of adjustments applied to synaptic weights and bias levels of a neural network leads it to become more knowledgeable about its environments. It can gain more information through increasing the number of learning process iterations [40].
There are two main different learning paradigms that can be used to train a neural network. This section addresses how neural networks learn with or without a teacher as well as simple overview of *reinforcement learning*.

4.4.1 Supervised Learning

Supervised learning networks is a machine learning paradigm for acquiring the relationship between input and output data. It takes a known set of input data and labelled responses to the data called training data, and attempts to design a predictor model that generates accurate predictions for the response to new data. The set of data, which enables the training process, is called labelled data or training data. Since the learning benefits from the assistance of the training data, it is considered as a learning process with a teacher [52].
In this training paradigm, both the inputs and the outputs are provided. The network then processes the inputs and compares its resulting outputs with desired outputs. The difference value between actual *neural network* and the desired outputs are errors of the network. These errors are then propagated back through the system in order to adjust the weights that control the network.

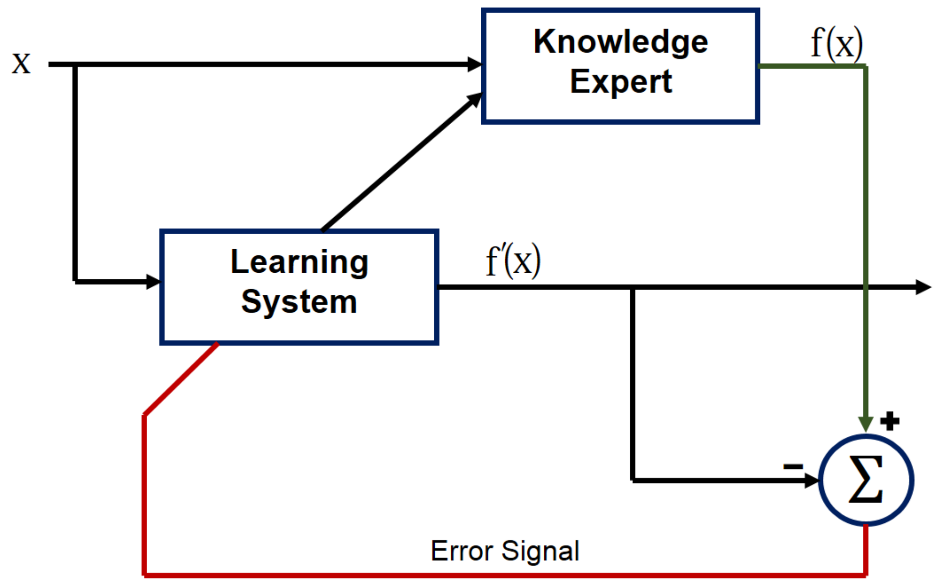


Figure 17: Supervised Learning Model [53]

Figure 17 depicts a simple model for supervised learning. It can be observed that the environment is measured and its measurement vector, x, is given to a knowledge expert who outputs a desired response, f(x). The learning system is also disclosed to the same measured variable and calculates a result, f'(x). The error between the output of the learning system and the desired response from the knowledge expert is computed and used to modify the response of the learning system. In general, improving neural network response means adapting weights, so that its response more closely matches that of the knowledge expert [53].

4.4.2 Unsupervised Learning

The term "unsupervised learning" or "learning without a teacher" is generically associated with the idea of learning to adapt based on the experiences collected through the previous training patterns. Thus, the network is trained to minimize the cost function by finding a suitable input-output relationship without using any teacher [40].

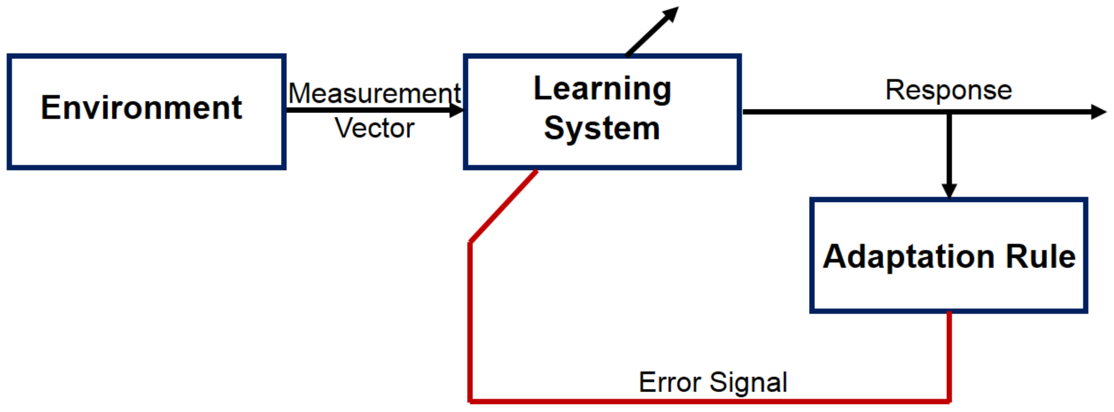


Figure 18: Unsupervised Learning Model [53]

Basically, the unsupervised training model shown in figure 18 is comparable to the supervised method, however differs in that no teacher is employed in the training process. It is analogous to students learning the lesson on their own, while in supervised learning labelled data used as a teacher. The open loop learning process shown in 18 consists of the environment, represented by a measurement vector. This vector is fed to the learning system that provides the response. Based on the system response and the applied adaptation rule, the weights of the learning system will be tuned to acquire the desired performance [53].

4.4.3 Reinforcement learning

Reinforcement learning differs from the learning paradigms mentioned above. It is called "learning with a critic", as altered from *supervised learning* which is based on

”learning with a teacher” idea. The main difference between a critic and a teacher is that, a teacher labels data in advance, while a critic gives a feedback about the learning process in the past. In other words, in this learning method, the learner is an agent that makes decisions and receives reward or penalty accordingly in order to solve a problem.
As it can be observed in figure 19 the agent interacts with an environment. At any state of the environment, the agent makes a decision and takes an action that alters the state and returns a reward respectively [49].

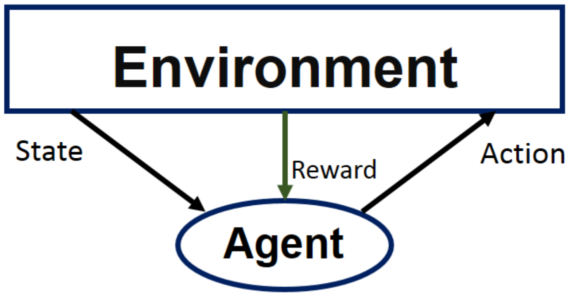


Figure 19: Reinforcement Learning process [49].

4.5 Summary

This chapter provided a general description of *neural network*, and the different types of architectures as well as *learning paradigms*. Three main *learning paradigms* discussed in this chapter, however this thesis employs merely the *supervised learning model*. The more details about the techniques used in this work specifically *backpropagation algorithms*, will be explained in the next chapter.

5 Experimental studies

This chapter describes the design and implementation of neural network applied in this thesis to detect congestion in CCNs. It focuses on presenting an overview of the whole system and also clarifies which components have been used and what roles they played. It describes also the rationale behind opting the specific *Neural Network algorithms* used and considering the particular metrics as well.
This chapter is organized as follows. In the first section, the motivation for applying neural network for our purposes is provided. The next section describes the whole process of designing neural network. In the next section, simulation environment and its specifications are explained, while in the subsequent section the important parameters that have impact on congestion consequences on every computer networking and CCN in particular are discussed. The last section will provide the result of neural network and compares different algorithms applied in this thesis.

5.1 Neural Network motivation

In real world network, packet loss can be caused by a number of factors including bit errors (wireless link), corrupted packets rejection (in NDN, unsolicited data packets will be dropped), and channel congestion. Since we are going to use simulation environment and assume that there is no DOS attack in the scenario, we are almost assured that packet loss is due to congestion and accordingly we can use the packet loss as a decent candidate for congestion signal in our experiment. We use the number of dropped packets due to buffer overflow as a main congestion manifestation. Therefore, the main goal of this thesis work is, to present a new approach of early detection of congestion based on the number of data packet drops in order to mitigate this issue in content-centric networking. In the following sections we will discuss the whole path that we have followed to achieve this goal.
As mentioned earlier in the previous chapter, the basic idea in using a neural network is to learn from its environment and improve its performance over time. It was discussed that a neural network can be varied in an almost infinite number of ways due to its various architectures . However, three main types of learning paradigms in neural networks were highlighted to provide sufficient background from which to interpret new concepts.
Since we benefit from the packet drop rate as a congestion indicator that teaches to the neural network the conditions in which congestion happens, the *supervised learning* method would be the best fit for our training purposes. Number of dropped packets are labelled in order to train the neural network as a teacher. Thus, the main purpose of using neural network is to early detection of congestion based on computed number of dropped packets.
The most important issue concerning supervised learning is the problem of error convergence, i.e. the minimization of error between the desired and computed unit values [59]. It is known that a neural network has synaptic weights (such as discussed parameters in the previous sections) that need to be updated and accordingly improving the performance of an ANN means optimizing these weights. This thesis

considers main important supervised learning algorithms and compares their results to approach the best trained neural network and accordingly the best prediction for network congestion.

5.2 Neural network design and environment

It is implied that we wish to implement neural network technique in our simulation environment to predict the number of dropped data packets at the bottleneck gate. Generally, any predictive data mining technique decomposes data in order to construct one or a set of models and attempts to forecast the behaviour of new data sets. It can be viewed as mapping or function , $y = f(x)$, where x is the input dataset, and y is the predicted value.
In order to have a predictor, we need to consider two issues. The first one is preparing the input datasets which involves a large number of preprocessing steps including data cleaning , relevance analysis, data transformation, and data reduction. Preprocessing the network inputs and targets improves the efficiency of neural network training. The second issue is the stage of comparing the different prediction models. The following sections introduce the methods by which we address these issues. Providing some basic knowledge of neural network, we will explain the data generation steps and at the very last section of this chapter, we will compare neural network models that we examined to conclude to the best predictor for our own purpose.
In this thesis work we have derived substantial benefit from using Matlab Neural Network Toolbox which supports supervised learning with feedforward networks. In addition to training, validating, and testing the neural network, the preprocessing, and postprocessing of datasets experiments have been conducted using this toolbox.

5.2.1 Training and Testing

Two of the most important measures of neural network models are how well the model generalizes to unseen data (generalization), and how well the model scales with problem intricacy (overfitting) [61]. Overfitting occurs when a network has memorized the training set but has not learned to generalize to new inputs. Since the ultimate goal of using neural network is to find the configuration with the best performance on independent data [53], it is of significance to train the neural networks based on an appropriate pieces of training datasets and their number of events. The former is applied to approach the generalization goal and the latter is to address the overfitting problem.
On the one hand, one of the major advantages of neural networks is their ability to generalize i.e., generate the appropriate outputs in response to inputs that are not part of their training experience. This means that a trained set of data can be applied to infer a rule for unseen data from the same class. Generalization is used to determine whether the classifier is memorizing the input data.
On the other hand, when a network is trained with too few events, it will usually memorize the data in classification tasks and will be unable to make predictions

as accurately as possible. In accordance with available neural network literature, overfitting is the most serious issue in neural network training and results in a loss of generalization performance.
Based on these facts, it is vital to use a sufficient number of events with great diversity in training dataset and apply testing data that were not used to train the neural network as well.

5.2.2 Data division

The first step in building the optimized neural network is to generate a set of relevant events which form the input dataset. Having acquired adequate datasets, it is extremely important to divide input data sets into precise pieces of data sets in order to train, generalize, and test the neural network. The input dataset is then divided into training, validation, and test data. Generally, this classification is applied to train an optimized neural network and to verify its accuracy.
A percentage of the records is used to build the model; the remaining records are used to test the model. In order to build model dataset is divided to build neural network with 70 percent for training, 20 percent for validation and 10 percent for testing.
The training set is used during the process of training to train a neural network. The error of this dataset is minimized during training. The validation set is used to fulfil the generalization goal and determine the performance of a neural network on patterns that are not trained during learning. A test set for finally checking the over all performance of a neural net when the training process has been finished.
The test data must be compatible with the data used to build the model and must be prepared in the same way that the build data was prepared. Typically the build data and test data come from the same historical data set. In this specific case, first, dataset collected from the first scenario used to train, validate and test the neural network, and then the other two scenarios are applied to test model further.

5.3 Simulation Environment

In communication and computer network research, network simulation technique has been introduced and developed to model the behaviour of a network and interaction among the different network entities rather than using mathematical formulas, or conducting an experiment with real world systems.
Computer network simulator usually permits system designers to study a problem from many different angles by spending less time in comparison with real system study. Nonetheless, simulations have many constraints. Correctness, speed, accuracy and relevance to the user's simulation needs are all extremely important issues to consider in order to approach the best possible conclusions [56]. The situation is even more difficult when there is a need to simulate a CCN concept which is itself does not exist in the real world.
This work was evaluated in the *ndnSIM*, an open source NS-3 based simulator, which implemented the basic components of a NDN(CCN) network in a modular way using

separate C++ (set of) classes to model behaviour of each network-layer entity in Content-Centric Networking [55].

5.3.1 NS-3 based Named Data Networking (NDN) simulator

The ndnSIM is NS-3 module that is optimized for Named Data Networking(CCN) simulation purposes. Its implementation effort started in fall of 2011 at UCLA, however the first release of the simulator as an open-source package has been made available since June 2012.
ndnSIM provides a set of reference application and helper classes, allowing evaluation of various aspects of NDN protocol under many different scenarios. It is implemented as a new network-layer protocol model, which can run on top of any available link-layer protocol model including point-to-point, wireless, etc., as well as IPV4 and IPV6 of Network-layer and TCP and UDP of Transport-layer. This ability allows ndnSIM to simulate wide range of homogeneous and heterogeneous scenarios. *ndnSIM* models behaviour of CCN network-layer entity including PIT, FIB, content store, network and application interfaces, etc. Moreover, this simulator is able to trace behaviour of every component and CCN traffic flow due to its extensive collection of interfaces and helpers [55].

5.3.2 Simulation settings

Design of a descriptive simulation scenario is a crucial task which needs to be carried out with special attention. The full range of parameters that might have influences on an experiment, must be opted precisely to increase the validity of simulation results. Since our area of particular interest is congestion related mechanisms in content centric networks, our models must include characteristics of congested links, content stores and PITs sizes, and the effect of congestion on these networks.
Table 1 lists the quantities chosen in our simulation scenarios with their design ranges. We agreed on these more influential settings in order to both train the *Neural Network* more appropriate and facilitate comparison of results as well. It is necessary to perfectly analyse the range of parameter settings in order to validate our results in this network model. In the subsequent sections , we will describe the rationale behind choosing these parameters and their possible affects on the network models.

5.4 Simulation Scenarios

In order to have reliable results, we needed to design wide range of possible congested network scenarios. Different scenarios can result in different degrees of congestion, in terms of the discarded packets number in every single router. A wide range of CCN scenarios in which congestion happens with dynamic range of parameters is designed and tested in this thesis to train the *neural network* properly and demonstrate how thoroughly it can perform. However, we evaluate the result based on the dumbbell topology which is typically used to analyse congestion issue in computer networks.

Table 1: Design range for the simulation parameters

Quantity	Design Range	Mode
Number of Senders	1-16	Pseudo-random numbers
Bottleneck Link speed	256Kbps- 1Gbps	Pseudo-random numbers
Link Delay	1ms -100ms	Pseudo-random numbers
Traffic Load	1 - 6000(Interest/second)	Exponential
Content Store Size	10,100,1000 Entries	Exact
PIT size	10,100,1000 Entries	Exact

We designed sixteen main scenarios with different number of senders, a wide range of links delays, and several other parameters that will be discussed in detail later in this chapter. It is worth mentioning that, in order to have more comprehensive results, we adopted an approach to change important parameters values in a random way during the simulation runs. We vary each network parameter in the basic scenario, one at a time, while keeping everything else fixed.

5.5 The dumbbell topology

In this simulation scenario the dumbbell topology illustrated in 20 with variable number of *Interest requesters (Consumers)* and *Data providers (Producers)* are simulated. *Hereafter the terms Interest requesters and Consumers for referring to Interest senders, and also Data providers and Producers for referring to Data senders will be used interchangeably*. These nodes are connected to the bottleneck link whose capacity is variable. Access links have 100Mb/s bandwidth available, their delays are 1ms, and traffic load is variable due to dynamic range of *Interest* rates. In this scenario scenario consumers at the left side of the bottleneck link issues interests for the contents served at the providers. Content payload size is fixed at 1024B and the interest size is 24B (which slowly increases to 28B due to the increasing number of digits in the content name: /datapacket/1, /datapacket/2, ...). This assumption will be maintained throughout the rest of the thesis work.

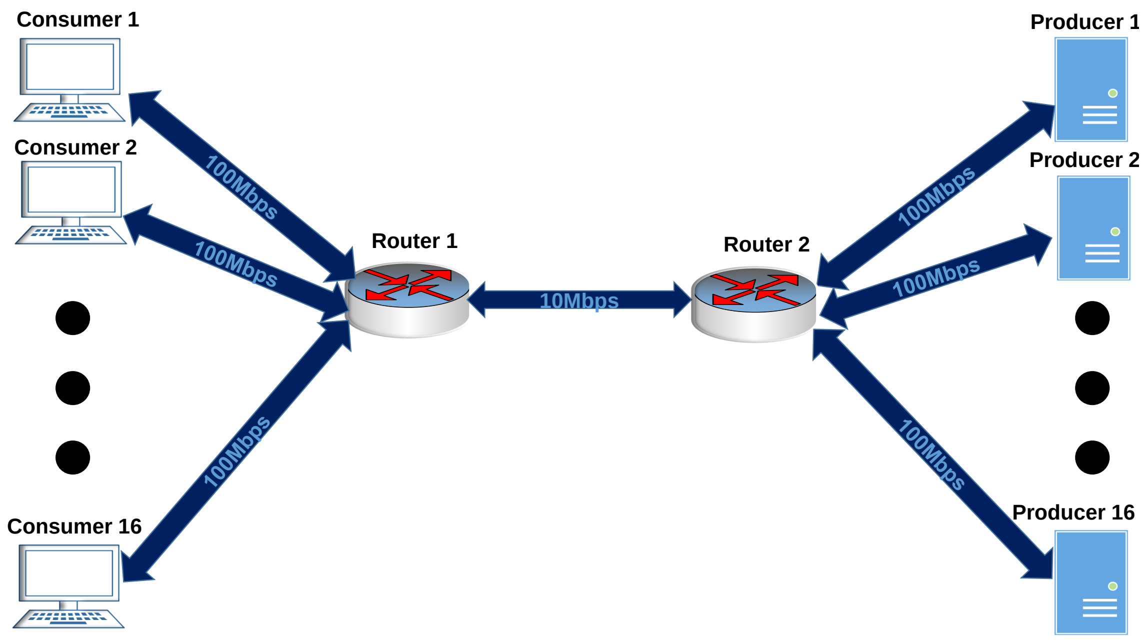


Figure 20: The dumbbell topology

The reverse path is symmetrical, however in order to have more realistic results some randomisation functions have been applied to some settings of the simulation scenario in the script code. These functions produce random numbers within the defined range provided in the Table1. The bottleneck link delay is static during the simulation, however, further scenarios are distinguished, as different number of senders is simulated with default values provided in the table 2. Figure 20 illustrates this network topology.
In this network topology we attempted to test as many as possible combinations of parameters in different simulation runs. This network simulation lasts 30 seconds every time that it runs. In every 5 seconds, simulator experiences different sets of parameters to deliver concrete and informative results. In an environment with large number of parameters, it is not easily feasible to isolate a particular variable and study its relation with a particular parameter due to inter-dependency among variables. However, in the following sections we consider the most important simulation settings and their effects on the whole network model with three nodes for both requesters and providers.

5.6 Input Parameters

In this section, we consider the most important parameters that might impact on congestion on a computer network in general and CCN in particular. In our experiment we run simulation environment in a large number of times, and selected one among them as a default value presented in table 2. In the following sections, the impact of aforementioned metrics will be compared by this default value table.

Table 2: Variable parameters values vs Default values

Time (Second)	Number of Transmitted Interests (Packet)			Bottleneck Bandwidth (Mbps)		Bottleneck Delay (Millisecond)	
Interval	Consumer 1	Consumer 2	Consumer 3	New Value	Default Value	New Value	Default Value
00:05	500	500	500	10	10	1	1
05:10	5700	100	3450	500	10	10	1
10:15	100	4499	1740	50	10	20	1
15:20	1339	878	3590	25	10	50	1
20:25	1990	1178	758	2	10	80	1
25:30	3099	4688	5359	100	10	100	1

5.6.1 Traffic load

One important factor in every packet network simulation is *traffic load*. Every possible computer network might undergo a wide range of *traffic load* from *zero traffic* to *burst traffic*. Since congestion occurs when the load of packets placed onto the network exceeds the capacity of the network to carry the packets, traffic load can have a huge impact on the way that network behaves.
In this particular network model we dynamically alter the *interest transmission rate* during the time that simulator runs in order to consider the crucial role that *traffic load* plays notably in network congestion. As it can be seen in table 1 the design range for the traffic load is from 1 to 6000 Interest packet per second for every *consumer*. In this part of experiment we collected the Interest transmission rate values produced randomly in the simulation script and provided in table 2. We will consider other columns of this table in subsequent sections.
Figure 21 illustrates that every consumer sends its requests with different transmission rate. The transmission rate has been produced in the scenario randomly and changes every time that scenario runs. As the topology of scenario suggests, all the transmitted *interests* will be aggregated in router 1 and then forwarded further through the *bottleneck* link. Figure 22 demonstrates this fact and depicts the *interest* transmission rate in routers as well. It can be simply observed that both routers forward *interest* with remarkably similar patterns except the last 5 seconds.

By observing precisely figure 22, we can underline the subtle difference between the pictures. When a router receives an Interest packet, it first checks whether there is a corresponding data packet in its Content Store. If a match is found, the data packet is sent back to the incoming faces and the Interest will not be forwarded

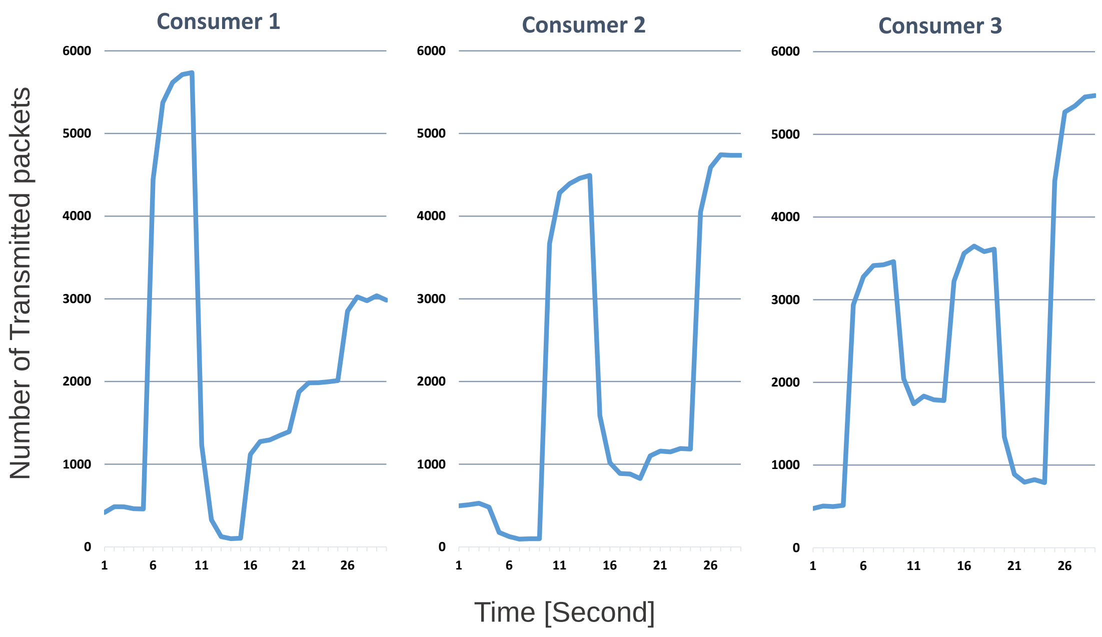


Figure 21: Number of transmitted Interest packets per second in *consumers*

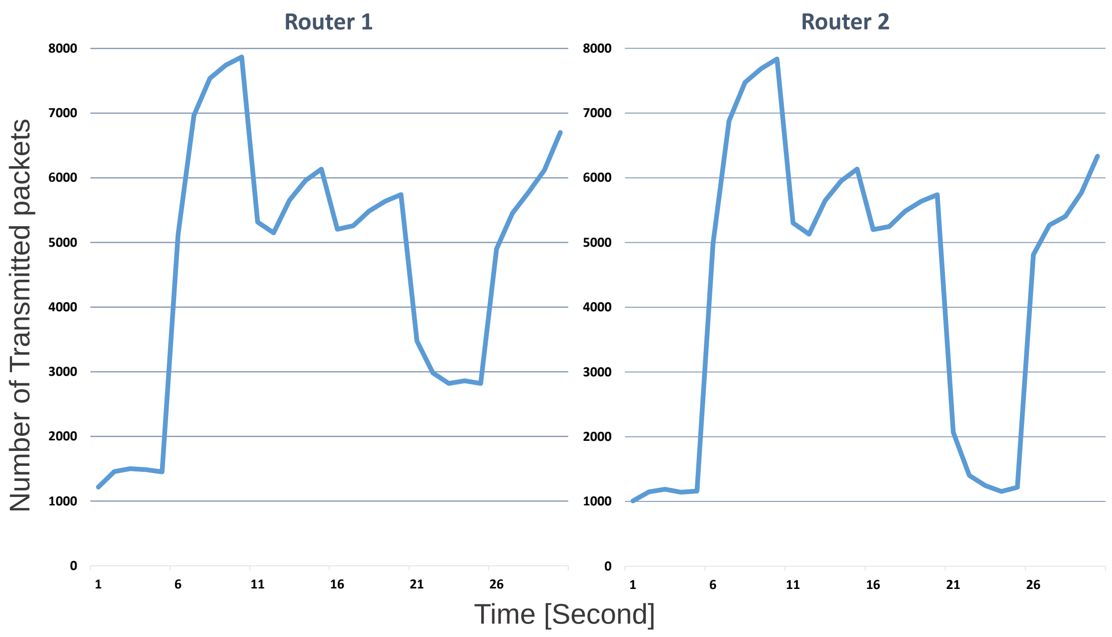


Figure 22: Number of transmitted Interest packets per second in *Routers*

further. The left picture in figure 23 shows the number of incoming Interest packet of router 2, which should be equal to the number of outgoing Interest packets from router 1, and the number of dropped Interest packets on this router. Since there is no discarded Interest packet on the router 2, the only reason that can justify

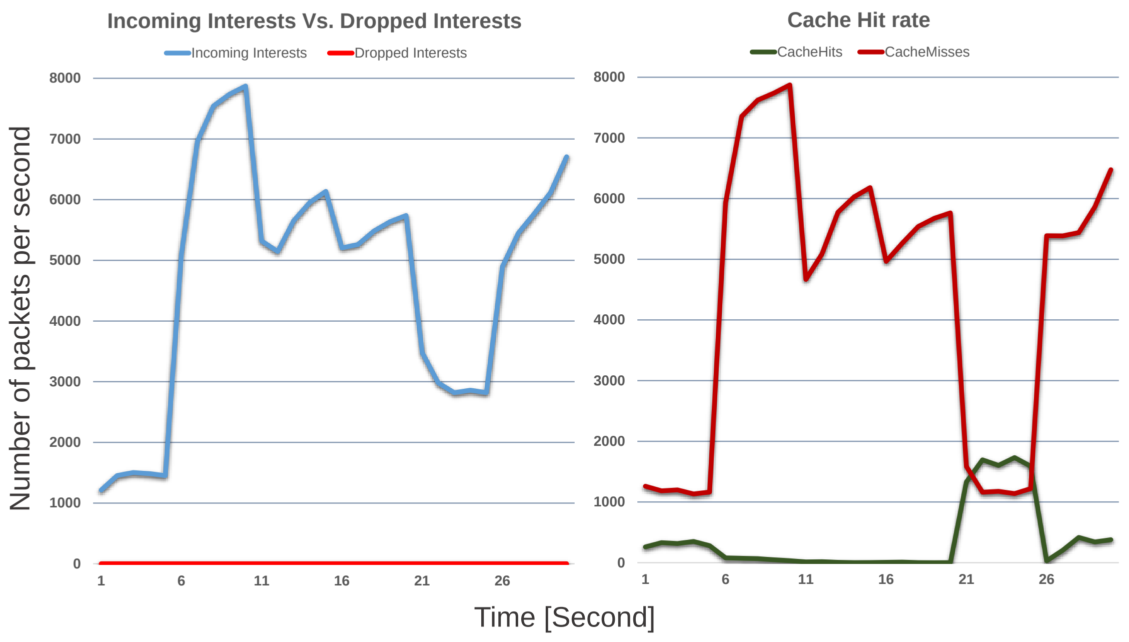


Figure 23: Left: Number of Incoming Interests Vs. the number Dropped Interests of Router 2, Right: Cache Hit ratio of the same router

aforementioned difference would be cache hits. The right picture shows that there are a number of cache hits, specifically in the first round and also the period of time between 21 - 26 seconds, that cause less outgoing Interest packets from Router 2 in this period of time.

5.6.2 Bandwidth

The bandwidth of a communication link is generally characterized by its latency and the number of bits that can be conveyed on the link in a certain period of time [58]. Although in real network, bandwidth of a particular physical link is a constant value, we needed to examine the role of this parameter on the network congestion and particularly on the number of dropped packets. In this network model, the same scenario with the same parameter values, but different bottleneck bandwidth is carried out.
Figure 24 compares the number of transmitted data packets and dropped data in both routers when the simulation run with its default values. As mentioned earlier, the bottleneck bandwidth has the value of 10Mbps for all periods of time in a one round of simulation, and we saw that by changing the Interest transmission rate the number of dropped packets is altered accordingly. In the next simulation round, we consider a hypothetical situation in which during every 5 second interval the bottleneck bandwidth value is changed. As it can be seen in the figure 25, the numbers of dropped data and forwarded data have been altered in accordance with the values provided in the table 2. We applied the gathered data of this test, in order to train our neural network algorithm as precise as possible.

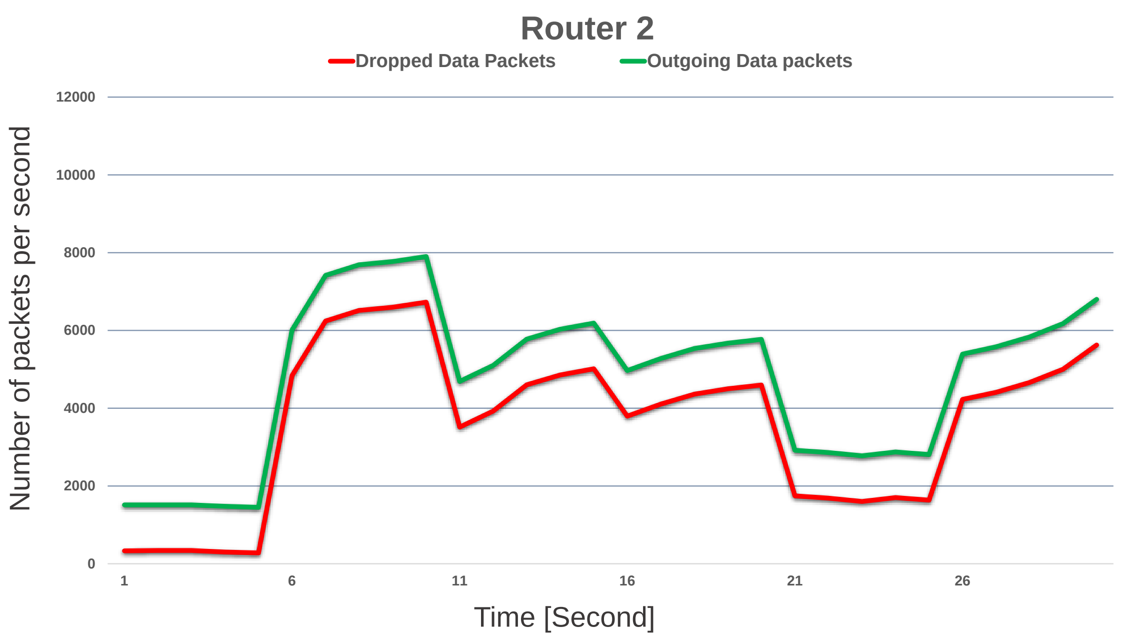


Figure 24: Comparing the number of outgoing data packets with the number of dropped data packets on the bottleneck with *constant* link capacity

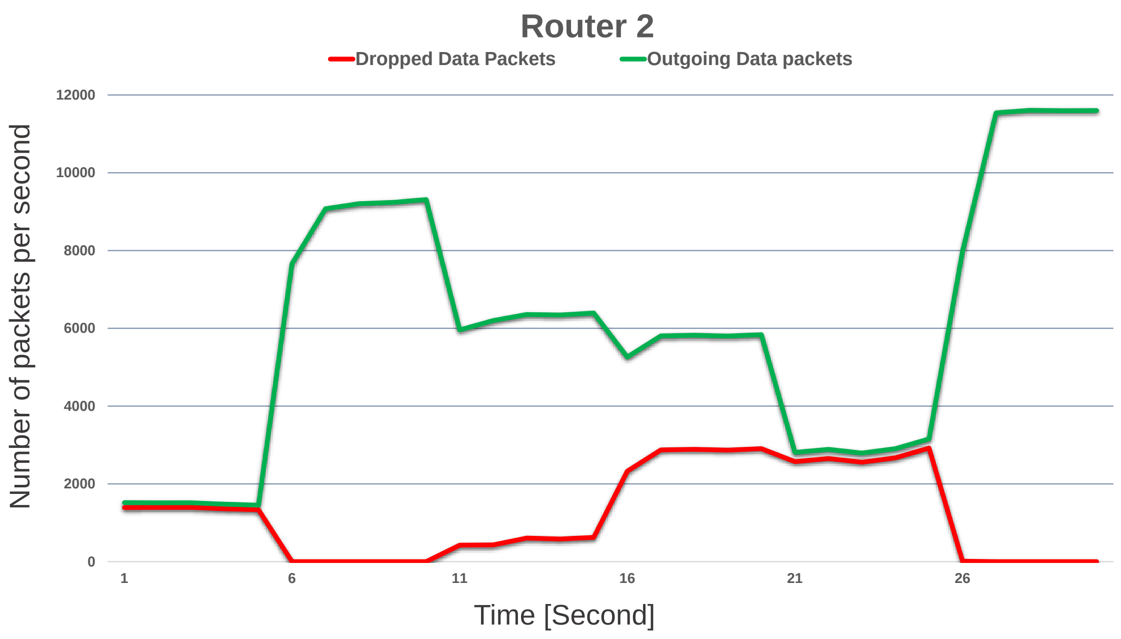


Figure 25: Comparing the number of outgoing data packets with the number of dropped data packets on the bottleneck with *variable* link capacity

5.6.3 Latency

In accordance with a definition of Round Trip delay or Round-Trip Time(RTT) provided in [58], we can apply the similar concept to define RTT for CCN networks.

The time that an Interest packet spends in travelling across the NDN network to retrieve the corresponding data is called Round-Trip Delay. Due to caching characteristic of NDN network, data packet can be retrieved from every node including, a data provider across the globe, or even from the next router. Therefore, in this data centric networking heterogeneous RTTs need to be considered predominantly. Figure 26 projects that when the latency becomes sufficiently high according to the table 2, the number of dropped packets will be increased drastically.

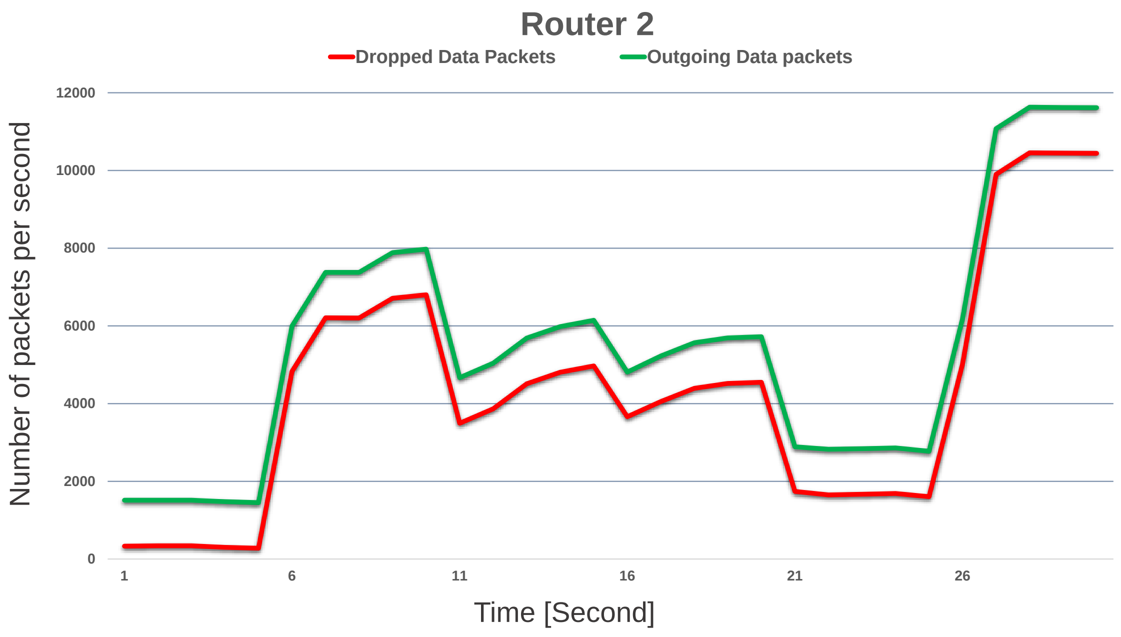


Figure 26: Comparing the number of outgoing data packets with the number of dropped data packets on the bottleneck with *variable* link delay

We used the values provided in table 2 to show that how significant role the link delay might play. By comparing figure 24 which shows the number of dropped packets based on default values and figure 26 which depicts the same parameters with new values, we can understand the impact of link delay or propagation delay on the packet loss rate more intuitively.

5.6.4 Queue length

Each queue has a limit on the number of packets that the router can place onto the queue. This limit is referred to as the queue length. During periods of high traffic, a queue fills with packets waiting for transmission. When a queue reaches its queue limit and becomes full, by default the router drops packets until the queue is no longer full. Accordingly, the long queue length allows the queue to keep many packets during congestion.
The queue length of routers in ndnSIM is user-configurable. In this section, we run

simulator with default parameter values, but different queue length for bottleneck routers to demonstrate the effect of this limit on number of dropped packets. It can be seen in figure 27 that when the queue length of router is larger 100 times, the number of dropped data packets is drastically increased.

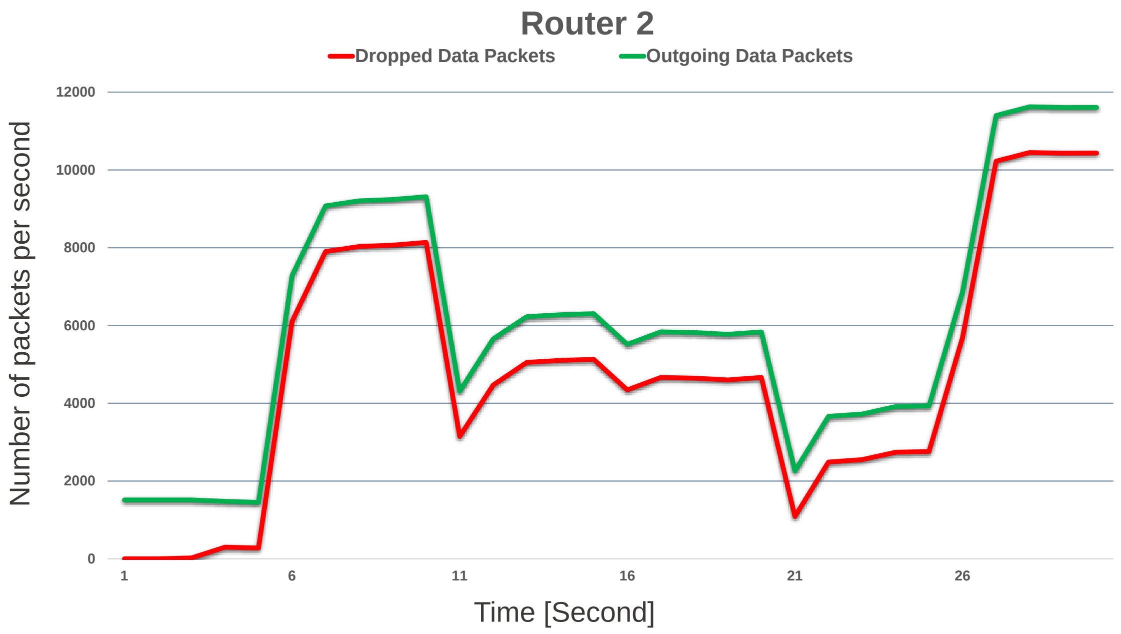


Figure 27: Comparing the number of outgoing data packets with the number of dropped data packets on the bottleneck with *Queue length = 1000 packets*

5.6.5 Content Store size and Caching policies

As mentioned in the section 2.4, each router on the CCN is equipped with a buffer memory called *content store* that has the the ability to cache data. Therefore, content store size can play a crucial role in controlling number of packet drops due to buffer overflow. Figure 28 compares the number of packet drops when the size of CS varies according to the design range value presented in table 1. It can be observed that by increasing the size of content store the number of dropped data packets are decreased accordingly.

On the other hand, sets of rules that manage the order of discarding packets are important features of any buffer memory. Caching strategies of the content stores determine in which order new incoming data packets should replace the already cached data. Caching strategy also, needs to track the content request rate at end nodes to measure the popularity of an object, and adapts the caching decision to better accordingly. Since designing efficient caching schemes for the NDN nodes is still under investigation and is out of the scope of this thesis, we decided to merely consider LRU caching replacement policies explained in section 2.3.1.

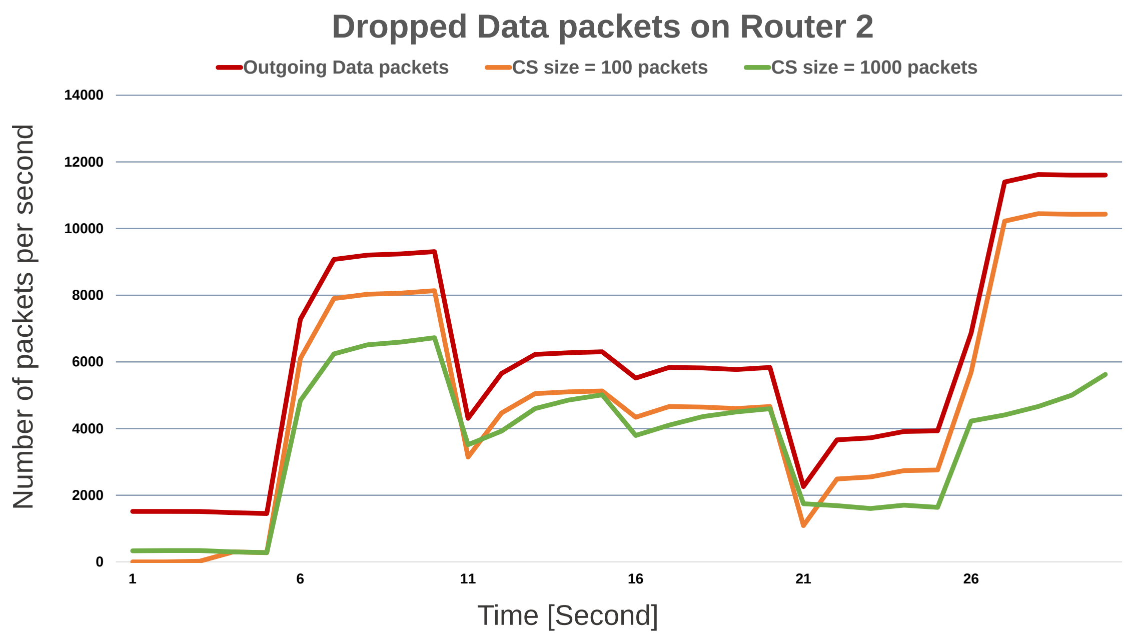


Figure 28: Content Store with 1000, 100 and 10 packets size

5.6.6 Data popularity

Content store replicates passing contents to serve the subsequent requests without the need of forwarding them to their source servers. The popularity of an object and caching policies are complementary parameters for every data centric networking.
One of the other effective parameter that required consideration was the popularity of content and its effects on the network congestion. The noticed network model examined both the situation that all the *data requesters* send *Interest* to request very popular data (like a YouTube video), and a condition of requesting different and unique *Interest packets*. Figure 29 and 30 illustrate caching trace of content stores embedded in the routers and underline the difference between those mentioned situations.
Figure 29 shows that when *requesters* send unique *Interest* in every single time instance, there is almost no *Cache Miss* in both routers. It is worth to remind that a cache miss, generally, is when requested data cannot be satisfied from the cache and the data has to be retransmitted from its original data provider. The *Cache Hit* on the other hand, occurs when the request can be served by simply reading the cache. It can be observed that in the *popular data scenario* shown in the figure 30 both routers hit their caches frequently. In subsequent sections we will describe that how an appropriate learning paradigm of a neural network can benefit from this information and also other data gathered by the routers.

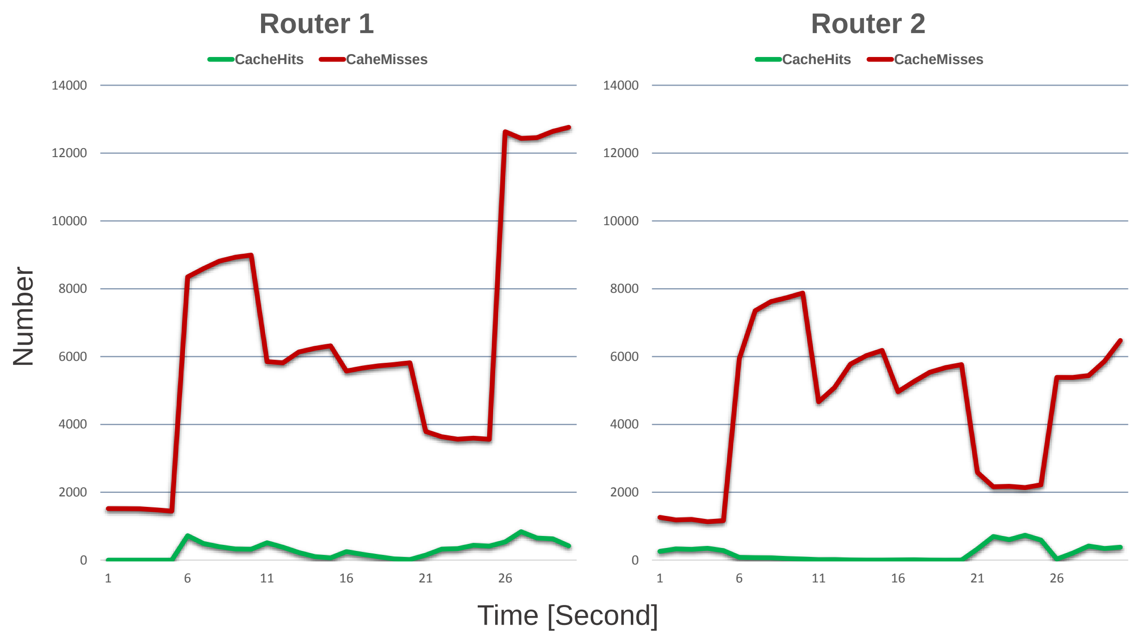


Figure 29: Cache Tracer of the scenario with different Interest packets

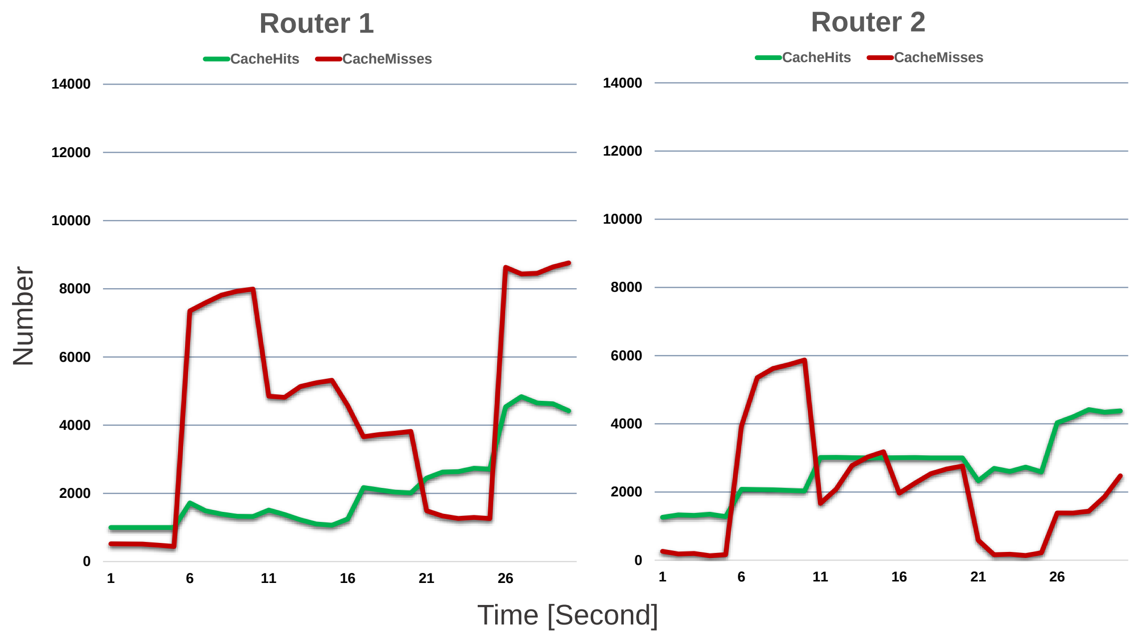


Figure 30: Cache Tracer of the scenario with popular data packet

5.6.7 PIT size PIT entry pruning timeout

PIT brings CCN many significant features [57], including detecting packet loss by keeping track of unsatisfied (*timedout*) Interests. The PIT size(the number of

pending Interests) is a crucial parameter which can have a huge impact on the number of both satisfied and timedout Interests, and consequently on the number of packet losses.
One of the major features of CCN architecture that can contribute to routers to have prior knowledge about incoming data packets is its unique routing and forwarding algorithm. Since a Data packet follows the reverse path of the corresponding Interest packet thanks to the PIT, CCN routers can manage traffic load through managing the PIT size. Moreover, having benefit from this forwarding packet strategy, a router is also guaranteed to observe whether a forwarded Interest is resulted in matching data or triggers PIT timer expiry.
Figure 31 compares the number of dropped data packets while PIT size is larger 10 times. It can be observed that when PIT size grows the second router is overloaded by incoming data traffic from any specific producer.

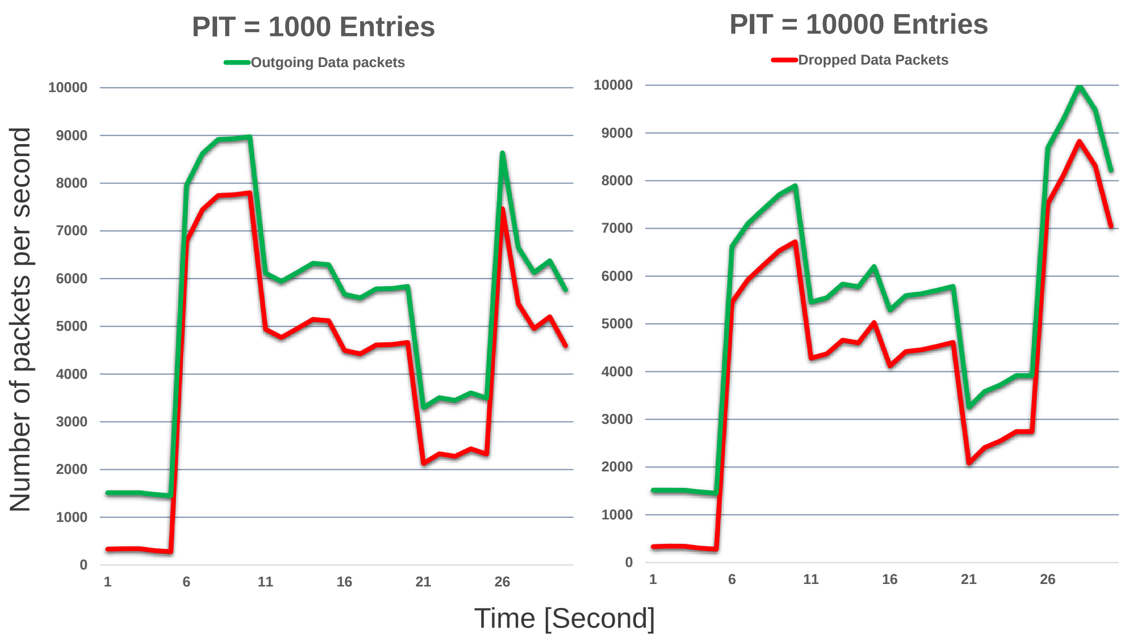


Figure 31: Comparison between the number of dropped data packets while PIT size becomes 10 times larger

As discussed in the section 2.3.2, each PIT entry records the name and incoming interfaces of every forwarded Interest packet, waiting for the matching Data packet to return. A router inserts every incoming Interest into PIT, and for each received corresponding Data packet removes its related PIT entry. PIT entry can be immediately removed (by default), or scheduled for removal within a short time interval. In ndnSIM there is a setting by which it is possible to set a different timer called *PitEntryPruningTimout* for PIT entry to live after being satisfied. The rationale behind this setting is, to assure recently satisfied interest will not be satisfied again.

In order to further clarify the role of this timer, consider the network simulation scenario once again. In this network model when the second router, adjacent to the Producers, receives the first packet, it will forward it to the consumer and remove the PIT entry immediately (PitEntryPruningTimout = 0). In this case, when the second content object arrives to the router from another Producer, it will be discarded due to being deemed as an unsolicited data packet. On the other hand, non-zero pruning time will prevent new Interest packets to be accepted, potentially resulting in more *Timedout* Interest packets. If an Interest is not responded by a matching data packet beyond a time threshold, it will be considered as a Timedout Interest.

5.6.8 Other settings

So far, we have considered almost all the important sets of input parameters that can have effects on network congestion in a NDN simulation scenario. The parameters discussed earlier can lead us to better understanding of NDN congestion and consequently to more precise conclusion. However, there are also other parameters including data payload size, memory access frequency, and number of complexity of both senders and routers which have not been taken into account.
Memory access frequency among them, is a constraint of any abstract network model and obviously cannot be examined. Other parameters will be considered either explicitly or implicitly in subsequent sections, but in order to avoid prolonging the first section of our experimental study it would be wise to postpone them for later consideration.
Traffic load(Interest transmission rate), bandwidth, link delay, Queue Length, number of Cache hits, number of Cache misses, CS size, PIT size, and finally, number of satisfied Interest packets were the parameters in input layer to train neural network as precise as possible.

5.7 Neural Network results

In order to select an optimal neural network architecture, it is necessary to address *which*, and *how* issues. It simply means providing an adequate rationale behind choosing specific training parameters, and training patterns. Having discussed the influence of training parameters, we need to provide an overview of training paradigms examined in this thesis and compare their delivered results.
In this section first, we provide a brief background knowledge about the neural network training algorithms applied in this thesis, and then the comparison among delivered results in predicting the number of packet drops based on datasets produced in default values. It is worth mentioning that, the following sections merely discuss the efficiency of these algorithms and finally choose the best one for further investigation. The neural network results with new and unseen datasets will be provided in the next chapter.

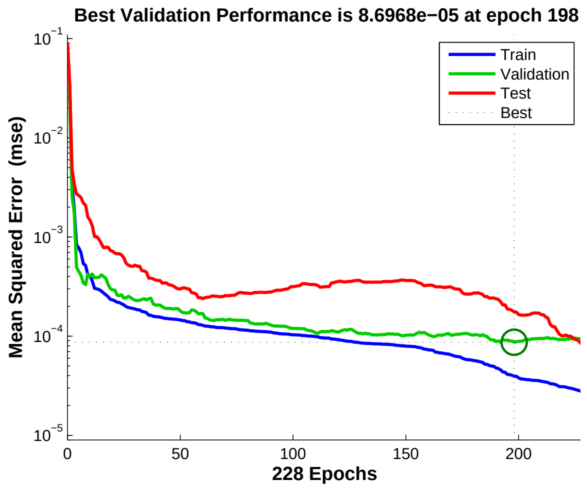
5.7.1 Back Propagation algorithm

A multilayer perceptron (MLP) as a feedforward artificial neural network model that maps sets of input data onto a set of appropriate outputs is used to utilize supervised learning technique called backpropagation for training the network. The Back Propagation algorithm, an abbreviation for "backward propagation of errors", uses the feedforward (see Section 4.3) topology, and is the most widely used algorithm for the supervised learning paradigm.
The neural network is trained with input data for which some desired output is known. The difference between the actual neural network output and the desired output is called an error. The error back-propagation algorithm then uses this error information to find the gradient of the network error function, with respect to the weights for a given input by propagating the error backwards [59]. It simply means that, the algorithm computes the combination of weights (see Section 4.2) which minimizes the error function in order to solve the learning problem.
The backpropagation algorithm looks for the minimum of the error function in weight space using the method of gradient descent. In this thesis work, we applied four different backpropagation algorithms (Gradient descent backpropagation, Gradient descent with adaptive learning rate backpropagation, Levenberg-Marquardt Learning Algorithm, Conjugate gradient backpropagation with Powell-Beale restarts). We then compared their results to define which one will best suit to our desired congestion control algorithm. All these algorithms, try to find the minimum of error function with using their specific method. Since comprehending the whole concept of these algorithms needs a vast background knowledge, we refer the readers to [52], [53], [60], and [61] for detailed explanations of rationales behind them.

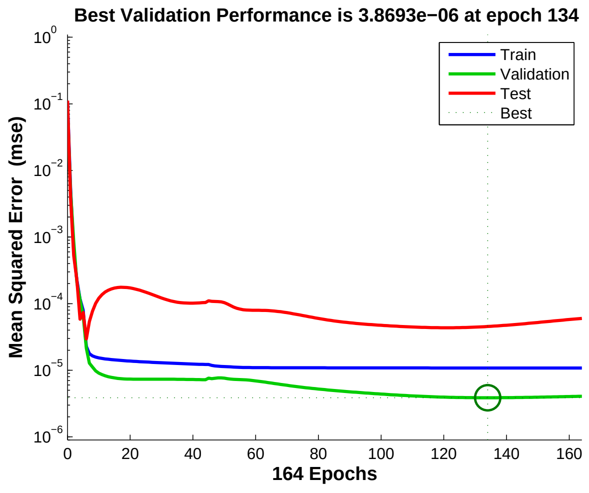
5.7.2 Performance Validation

As it was described earlier, the learning process is carried out through iterative changes in the synaptic weights and biases, in accordance with some learning rules. Each of these iterations is often referred to as an epoch. The important question is that when the learning process should be paused in order to avoid overfitting. Of most interest are the performance, the magnitude of the gradient of performance, the number of validation checks and maximum number of epochs.
However, the magnitude of the gradient and the number of validation checks are used to terminate the training, we used maximum number of epochs for our purpose .It is another way to stop the process of learning in training the neural network by assigning the number of epochs that learning process continues after validation error has been reached to its minimum.
In order to reach to the best generalization point the training process should be stopped in the minimum of the validation set error. When learning is not stopped, overtraining occurs and the performance of the network on the whole data decreases, even if the error on the training and test data still become smaller. Figure 32 plots performance progress of the four training algorithms. It indicates the iteration at which the validation performance reached a minimum error. The training continued

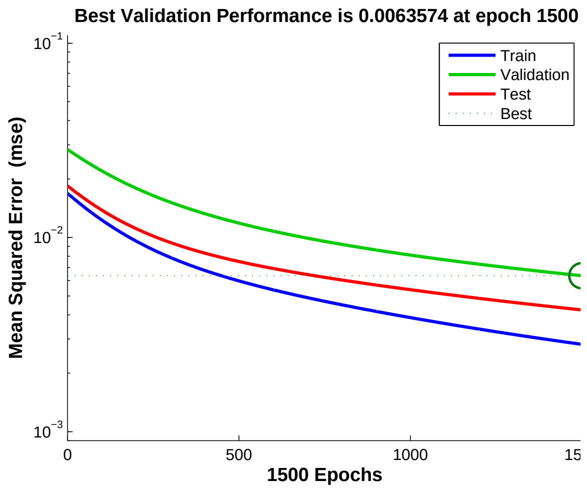
for 30(the limit is adjusted manually) more epochs before the training stopped. It shows also that the validation and test curves for all algorithms are reasonably similar. If the test curve had increased significantly before the validation curve increased, then it is possible that some overfitting might have occurred. It can be seen in the 32(b) that around epoch 134, the network learns the final data point or points in the training set, and its error drops close to its minimum. Although errors for both train and test data decrease further at this point, validation error keeps declining in the next 30 epochs.



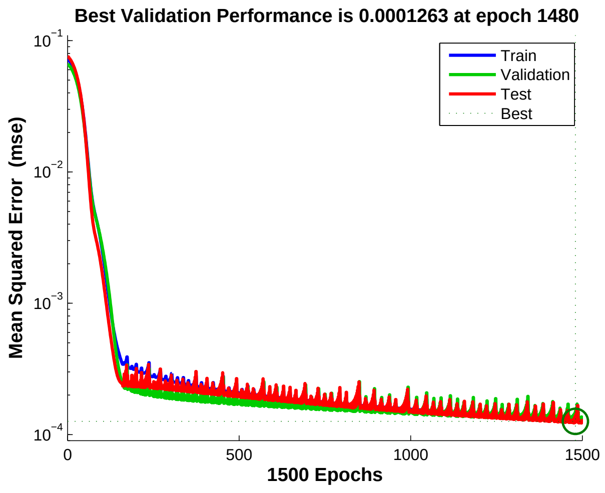
(a) Conjugate Gradient



(b) Levenberg-Marquardt



(c) Gradient descent

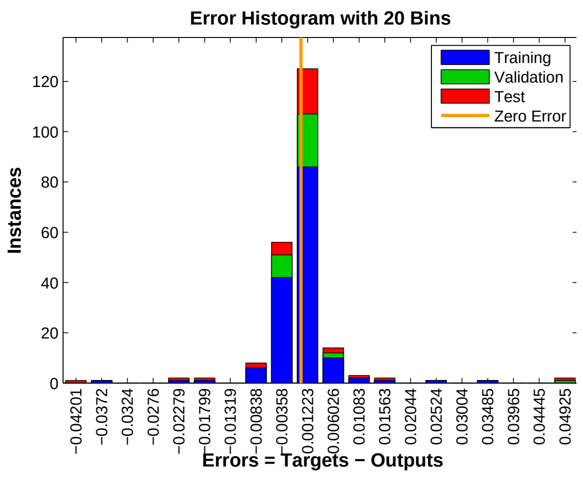


(d) Gradient descent with adaptive learning rate

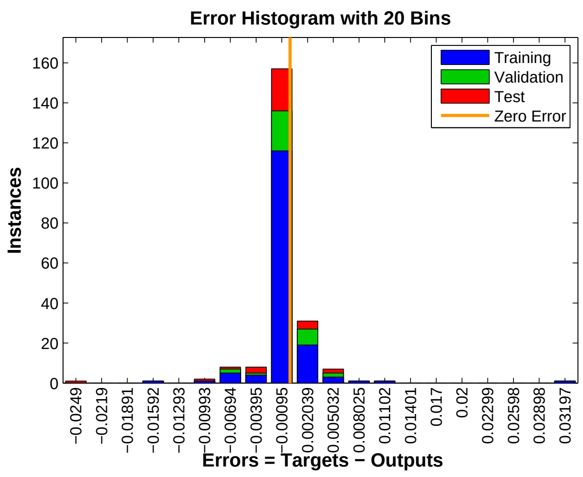
Figure 32: Error development of a training, test, and a validation set. Note that the Y axis ranges are different; This figure compares MSE of train, validation, and test data sets of different algorithms separately, and shows the epoch number at which each particular algorithm has terminated the learning process.

5.7.3 Error histogram

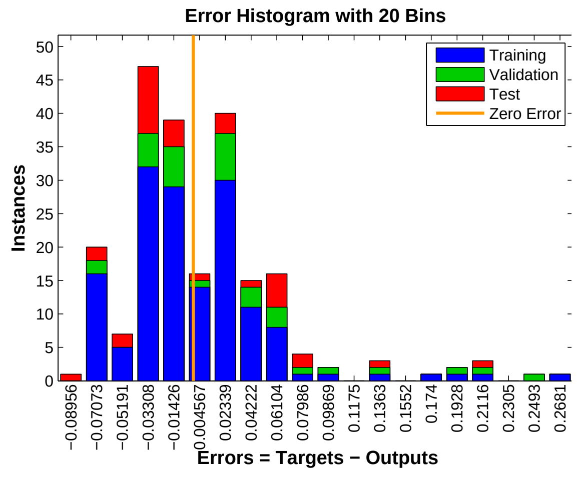
A histogram is ”a representation of a frequency distribution by means of rectangles whose widths represent class intervals and whose areas are proportional to the corresponding frequencies” [62]. The error histogram is used to interpret how reasonable is to assume that the random errors inherent in the process have been collected from a normal distribution. The normality assumption is needed for the error rates we are willing to accept when making decisions about the process to indicate the reliability of an estimate.



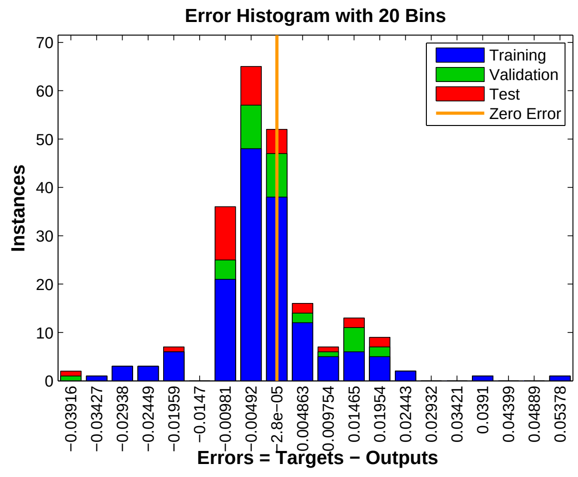
(a) Conjugate Gradient



(b) Levenberg-Marquardt



(c) Gradient descent



(d) Gradient descent with adaptive learning rate

Figure 33: Error Histogram

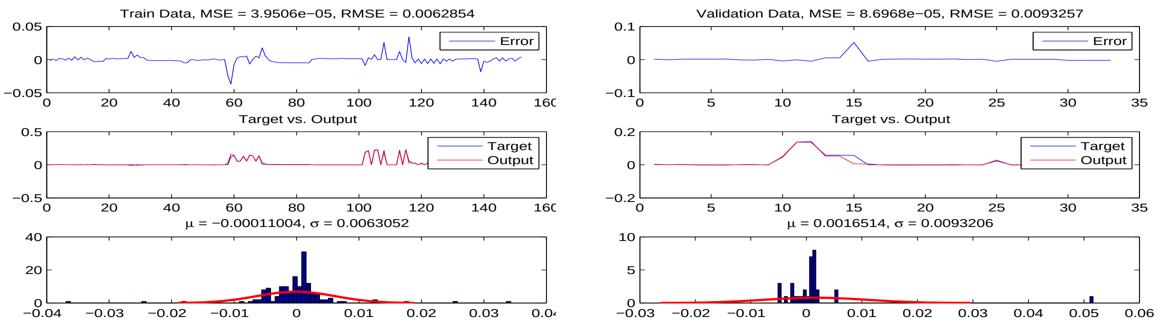
The most common form of the histogram is obtained by splitting the range of the data (error in our particular purpose) into equal-sized bins for which the number of points from the data set that fall into each class (bin) are counted. Figure 33 graphically summarizes the distribution of errors in four mentioned training algorithms. Conjugate Gradient and Levenberg-Marquardt seem to have more reasonable error histograms. They do not appear to be significant outliers in the tails, and they imply sensibility of supposing that the data are from approximately a normal distribution.

5.7.4 Predictor accuracy

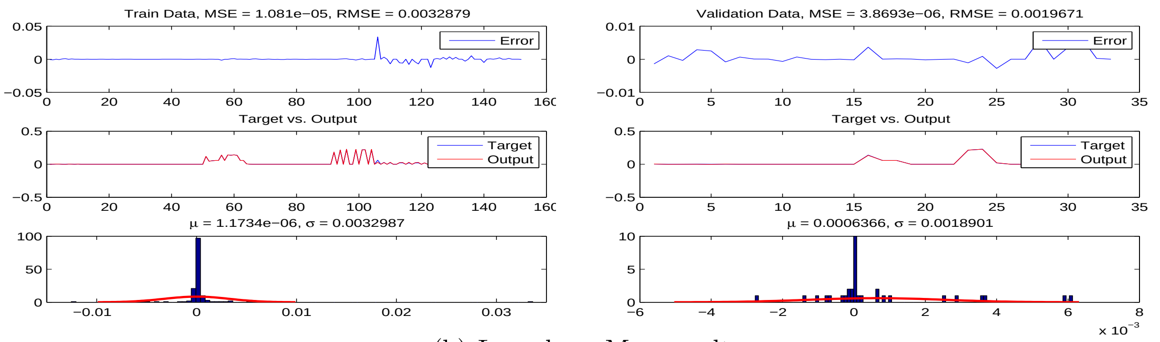
The Mean Squared Error (MSE) and Root Mean Square Error (RMSE) are arguably the most important criterion used to assess the goodness of a predictor or an estimator in terms of the relative magnitude of its bias and variance [62]. An MSE of zero, meaning that the estimator predicts observations of the parameter with perfect accuracy. Generally, this measure aggregates bias and precision to evaluate how close an estimator is to the parameter it is predicting. The RMSE is just the square root of the mean square error and is directly interpretable in terms of measurement units to measure the goodness of prediction.
Results shown in figures 34 and 35 are in order MSE and RMSE at the top, the normalized target and output values at the middle, computed standard deviation and mean error at the bottom for train, validation, test datasets separately and the whole data sets totally. By considering all plots, it can be concluded that Levenberg-Marquardt algorithm delivers more reliable results in terms of error measurement, and fits target and output data more accurately. These four plots prove that this specific algorithm is capable of predicting the number of packet drops in similar scenarios.

5.7.5 Correlation between Target and Output data

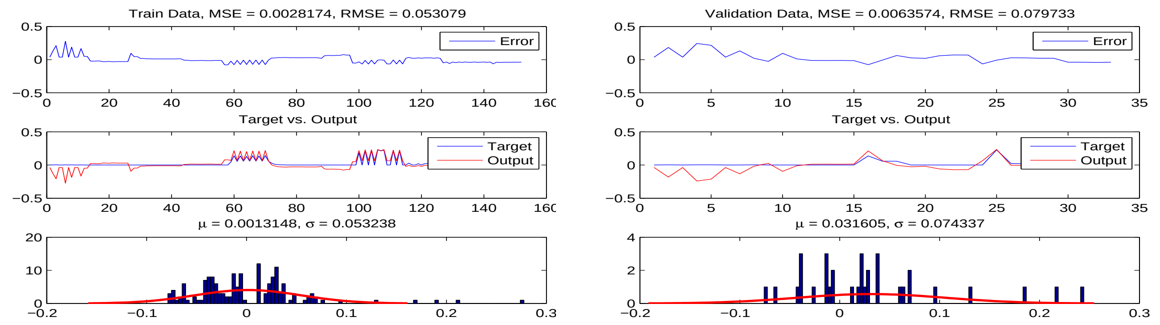
Regression is a data mining function that predicts a number. Regression models are tested by computing various statistics that measure the difference between the predicted values and the expected values. This metric is used to estimate how accurately the model predicts these known values. The regression has been projected in figure 36 for train, validation and test data both individually and totally. Based on all statistical analysis, performance of the Levenberg-Marquardt technique is the most suitable for our purpose and meets the router requirements to be able to predict the congestion situation of every link in the future time instance.



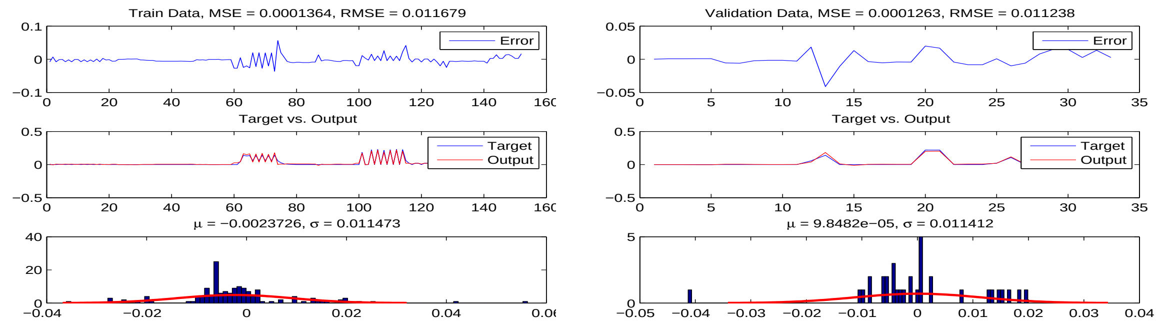
(a) Conjugate Gradient



(b) Levenberg-Marquardt

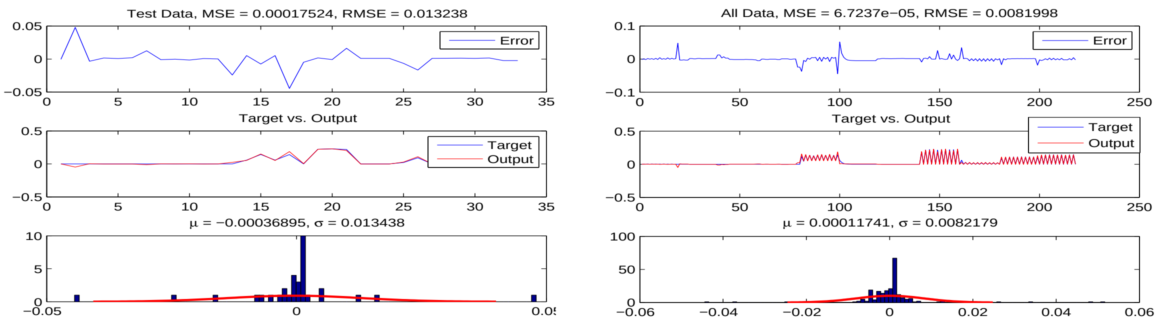


(c) Gradient descent

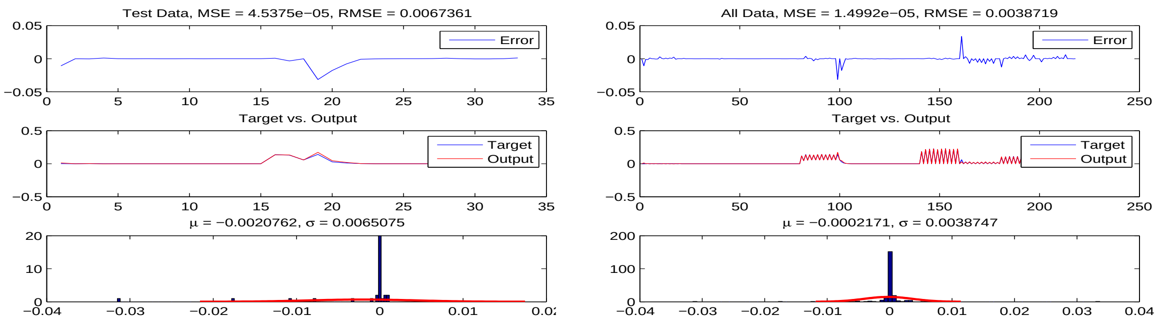


(d) Gradient descent with adaptive learning rate

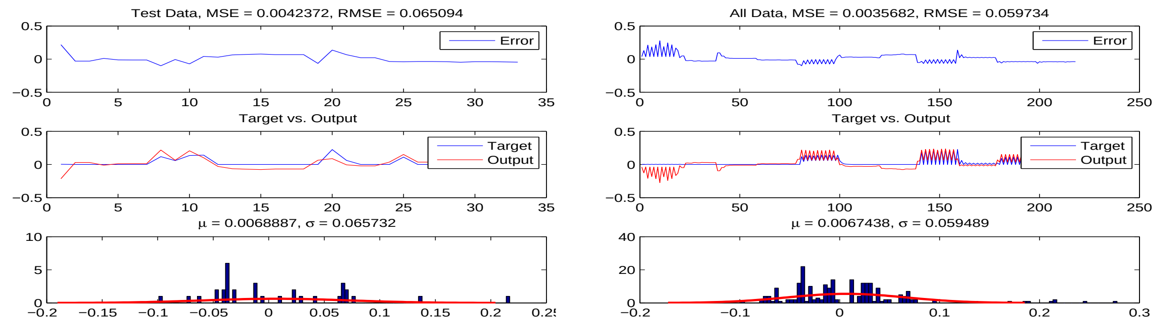
Figure 34: Train and Validation



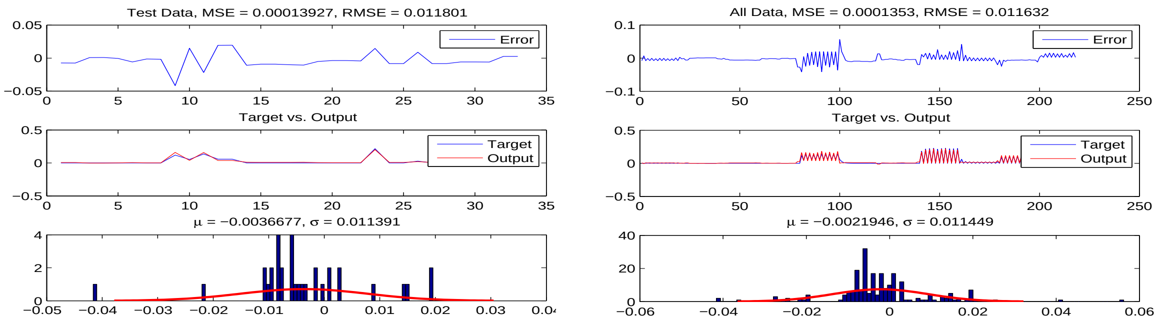
(a) Conjugate Gradient



(b) Levenberg-Marquardt

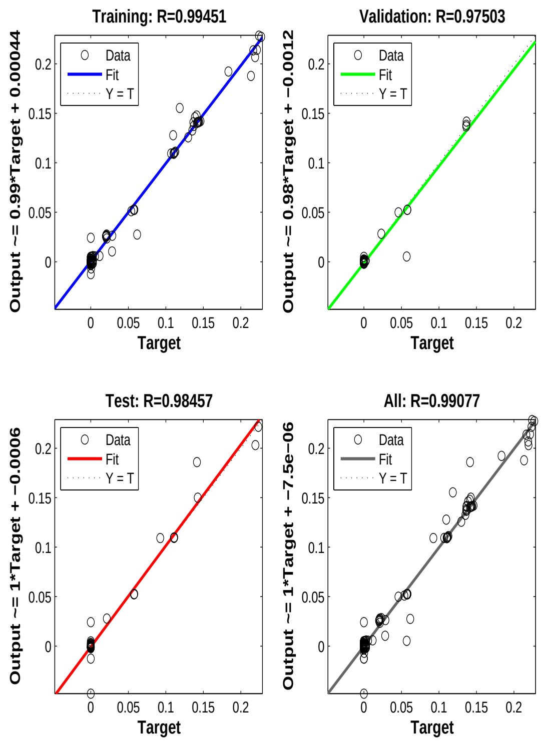


(c) Gradient descent

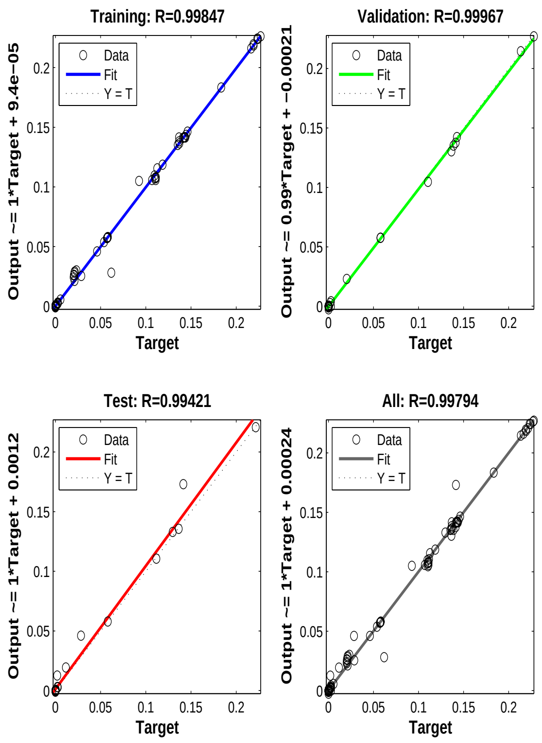


(d) Gradient descent with adaptive learning rate

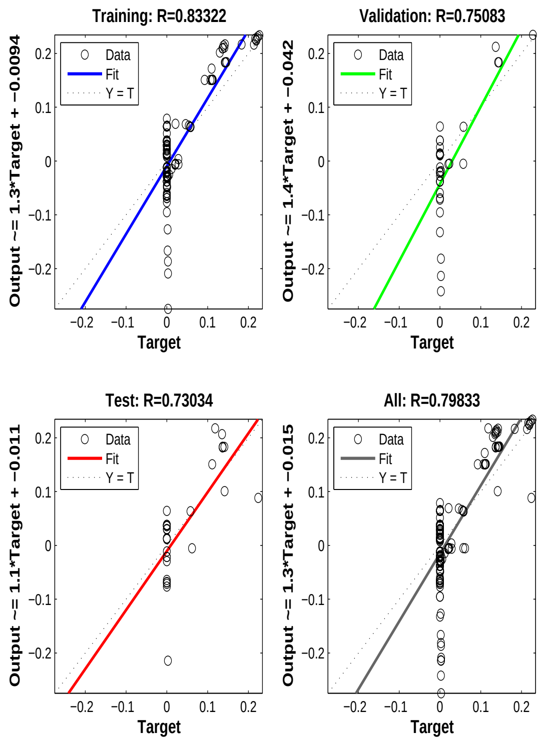
Figure 35: Test and All data



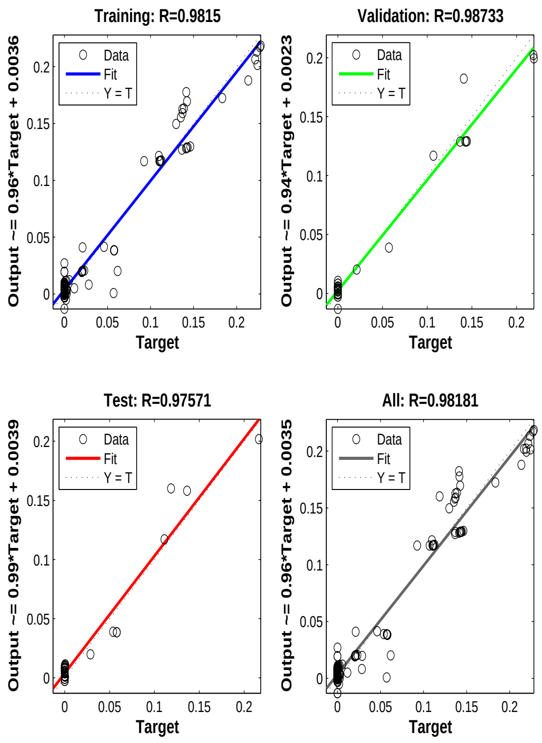
(a) Conjugate Gradient



(b) Levenberg-Marquardt



(c) Gradient descent



(d) Gradient descent with adaptive learning rate

Figure 36: Regression

5.8 Summary

In this chapter, first, we discussed the important parameters that can affect the congestion of NDN network. Then, we explained their possible impacts on the network and collected them to build input dataset for desired neural network. In the next section, the neural network design and four different algorithms applied in this thesis described and their results compared.
Having considered the results of neural networks in the last section of this chapter we came to conclusion that Levenberg-Marquardt is the best suitable algorithm to be applied for our purposes. In the next chapter we will describe how we have applied our neural network model to the different NDN scenarios and prove that the proposed congestion control algorithm for these networks.

6 Evaluation

Chapter 5 described the design and implementation of our neural network scheme for predicting the number of dropped data packets in every attached face of NDN routers. In this chapter, we evaluate the behaviour of our already trained neural network through different network scenarios as well as performance measurements in congested bottleneck. This chapter illustrates the effectiveness of the proposed technique to control congestion in content centric networking and in a broad sense in Information centric networking.
In the first section we describe the proposed congestion algorithm producing some results of the simulation run. We, then turn to solidify the accuracy of trained neural network by providing different network topologies and comparing theirs behaviours with and without neural network in routers. In the last part of this chapter we implement the neural network algorithm in the first scenario and measures its throughput and packet drop rate which are the most important metrics for assessing the efficiency of any congestion control algorithm.

6.1 Congestion control algorithm

In chapter 3 we provided the background about congestion in computer networking, and described the most important congestion control algorithms both in traditional IP networks and in the Information Centric Networking focusing on CCN. We came to the conclusion that, since CCN communication model does inherently differ from flow-based end-to-end model of the current Internet, we need to have an algorithm that is capable of early detection of congested links based on collected data in routers.
In this thesis, a neural network algorithm is implemented in each CCN router to decompose data gathered from its local environment (PIT, CS and Faces) in order to early detect of congestion. Based on the output of a trained neural network, router is able to predict the probability of congestion based on queue overflow in the next time instance.
Queue overflow occurs when an output link from a node has a load factor that exceeds 1.0. That is, data is arriving at the queue for the link faster than it can be transmitted. As a result, queue length grows until there is no more space in the queue. At that time, a node has no choice but to discard a packet. As mentioned earlier, an equipped neural network router can predict the number of data packets that are going to be discarded due to overflow in each face. The main idea is rooted from the fact that every data packet in CCN network follows the reverse route along which corresponding Interest packet has already travelled.
Neural network will deliver the predicted number of dropped packets for every time instance. This number can provide the probability of link congestion by dividing it to the whole number of outgoing data packet in every interface as shown in the equation 5. It is worth highlighting that, we apply the packet loss as a manifestation for congestion in content centric networking.

$$P = D/(I + D) \tag{5}$$

where P is the probability of discarding data packets; D is the number of dropped packets, and I is the number of successfully received data packets. At this stage, router sends a notification to downstream nodes to adapt their Interest transmission rate in accordance with the value of P.
Figure 37 depicts signalling between two routers attached to a bottleneck. Once NN calculates packet drop probability in Router2, it sends back a packet transmission rate packet to its downstream node shown as Router1. Downstream node then, decreases its Interest transmission rate according to the probability of congestion predicted by neural network. This process continues until NN output reaches zero which means any Interest will be sent successfully in the next time instance. From this point, congestion control transits to a new phase which is similar to TCP congestion control slow start. In this stage, every time that Router1 receives zero percent shaping rate, it will increase transmission rate exponentially as it it is shown in the figure.

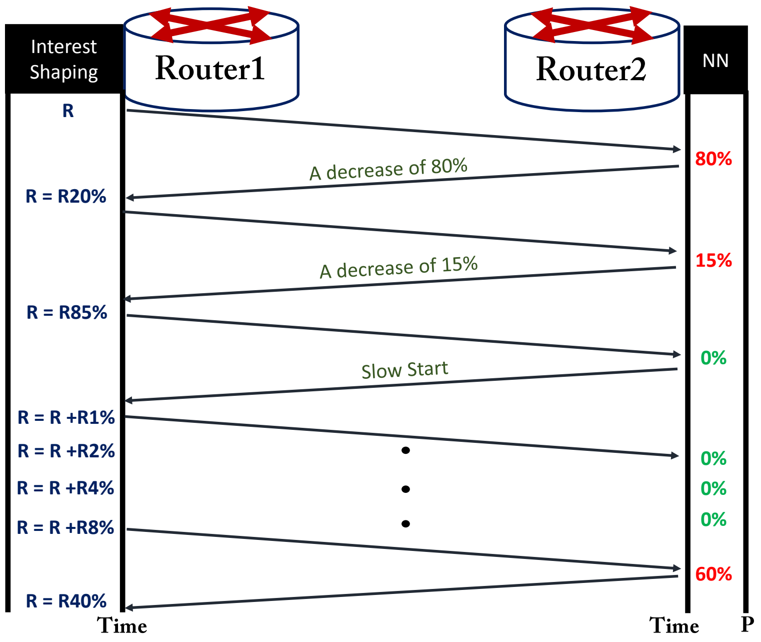


Figure 37: Shaping Interest rate signalling

The shaping rate notification implies that the transmission rate of each node is based on the value that is delivered from the upstream nodes. For instance in the mentioned scenario the Interest transmission rate of the first router is adapted at any time instance by the received shaping rate sent back from the second router. The table 3 applies the equation 5 to modify the Interest transmission rate of the first

Table 3: Transmission rate for Router 1

Time Instance	Transmission rate(Packet/second)	Probability of dropped data packets	Shaping rate	Modified rate for the next time instance
3	2078	43%	57%	1184
8	2986	61%	39%	1164
13	3093	62%	38%	1175
18	2324	49%	51%	1185
23	1282	01%	99%	1269
28	2155	83%	17%	1185

router based on the probability of discarded data packets assessed by the trained neural network. It can be seen that the Interest transmission rate of the first router is being modified by the shaping rate notifications sent by the second router. It is worth to highlight that the traffic load is assumed to be constant for the time intervals of five seconds. This is due to the fact that the simulation environment is not capable of responding to the sudden traffic load fluctuations immediately.
It is important to clarify that, the thesis does not take into account the influence of sending shaping rate notification to downstream nodes on congestion of the network. Therefore, the question of how fast the rate notifications need to be sent back remains as a future area of study.

6.2 Neural Network validation

Having established training method and concluded to the sensible result in the previous chapter, it is necessary to determine how well our trained neural network performs in detecting packet loss rate with unseen datasets. Multi layered perception with Levenberg-Marquardt backpropagation algorithm leaded to the best results among the other neural network techniques. In this step we run the three different dumbbell topology scenarios within design range, but with absolutely new combinations of discussed metrics. The new test datasets are provided to the neural network to determine how accurately it can predict the number of dropped data packets on the bottleneck of the dumbbell topology applying new simulation settings.

6.2.1 Validation scenarios

In this section we introduce three different networking models used to test the accuracy of the neural network in predicting the number of data packets. The very first topology that we consider in this section is, the baseline topology which is the simplest form of the mentioned dumbbell topology.

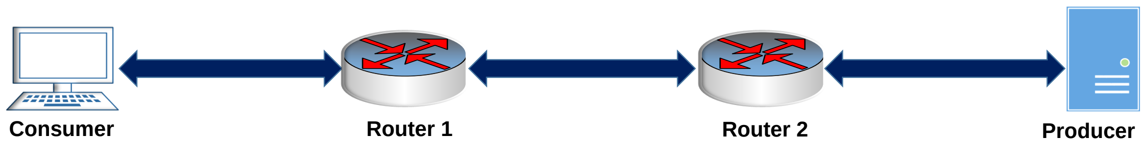


Figure 38: The Baseline topology

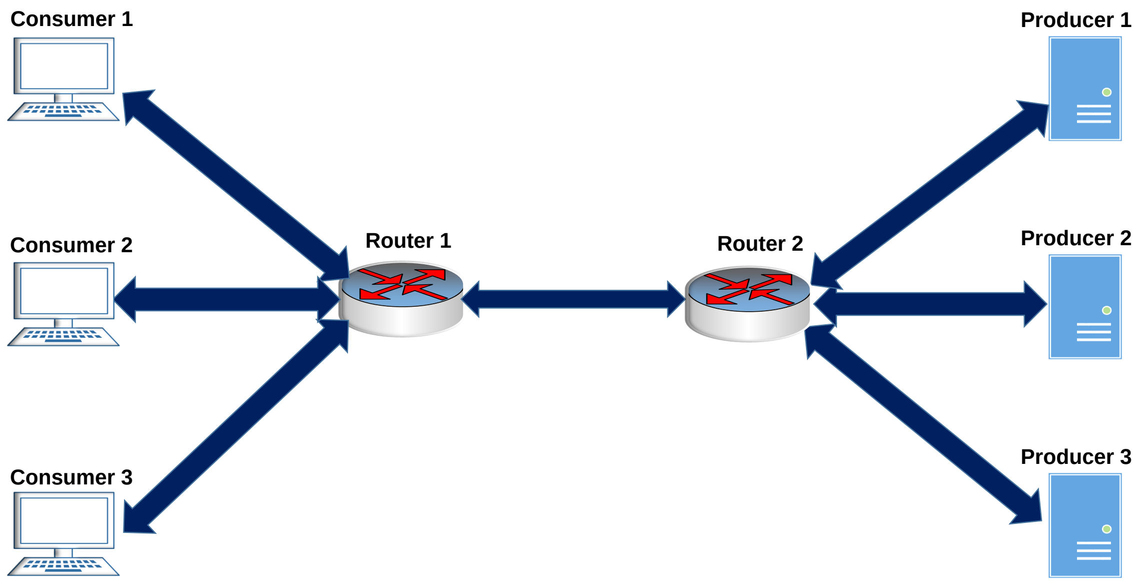


Figure 39: The dumbbell topology with three flows on the two ends of bottleneck link

In the baseline scenario shown in figure 38, consumer(Src1) at the left side of the network transmits interests through the routers(Router1 and Router2) for the contents served at the provider(Dst1). Content payload size is fixed at 1024B and the interest size is 24B (which gradually increases to 28B due to the increasing number of digits in the content name: /datapacket/1, /datapacket/2, ...). It is worth

mentioning that the payload size of both Interest and Data packets will be kept constant for all the other networking models.
The next scenario illustrated in figure 39 is the dumbbell topology with three consumers and three providers. This networking model studies the new settings of the exact topology that we presented earlier in the chapter 5 to analyse the roles that different metrics play to alter the number of dropped data packets on the bottleneck. As it can be observed in this figure, packet drop event occurs on the right side of the bottleneck link where the second router is struggling to put the data packets to the bottleneck link. Therefore, the term packet drop in this thesis work refers to the number of discarded data packets on the router 2.
The last networking scenario that we consider in this thesis is the scenario depicted in figure 40 with the same dumbbell topology, but with nine flows at both ends of bottleneck link. This scenario is designed to solidify that the neural network can perform its task regardless of number of data flows attached to the bottleneck.

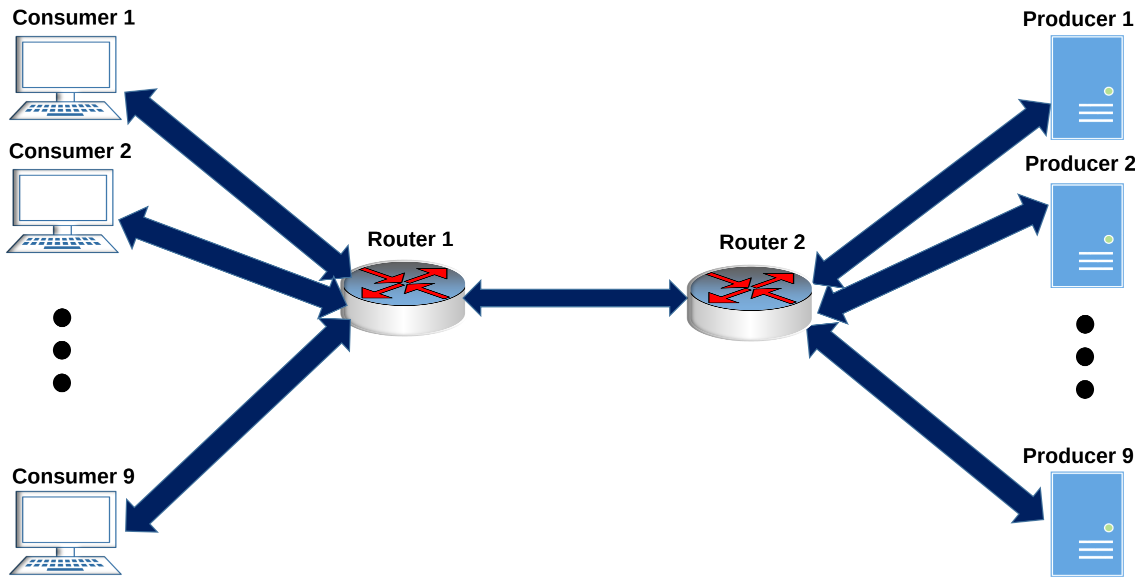


Figure 40: The dumbbell topology with nine flows on the two ends of bottleneck links

6.2.2 Validation results

The effectiveness of neural network can be measured using the accuracy of a predictor refers to how well a given predictor can guess the value of the predicted attribute for new or previously unseen data. In this section we provide some graphs to prove that trained neural network can predict the number of dropped data packets with high accuracy.

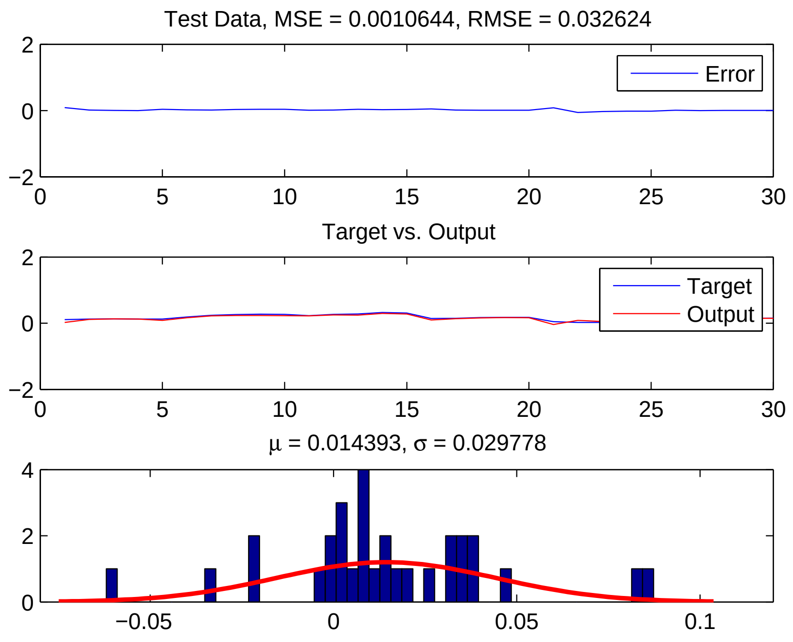


Figure 41: measured Error at top, normalized target and output between -1 and 1 at the middle , and error histogram at the bottom when the test data is gathered from the baseline topology

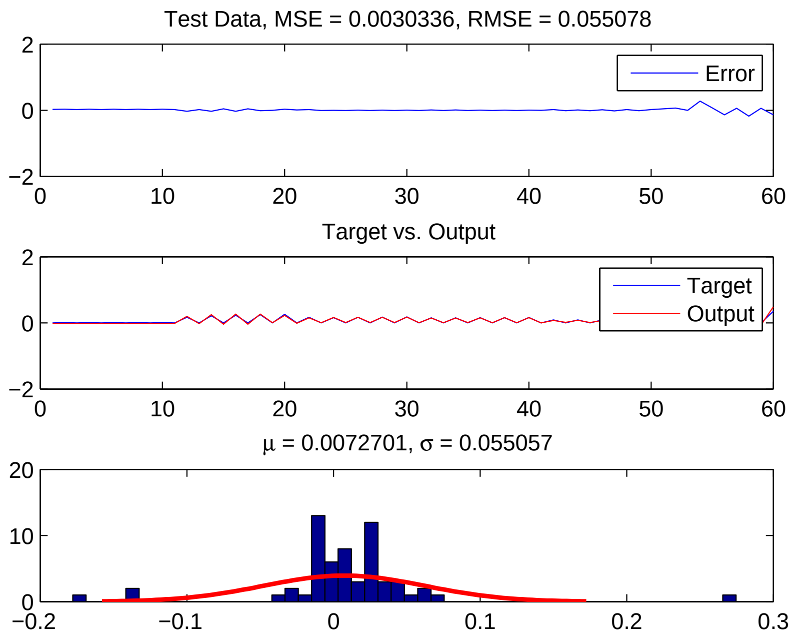


Figure 42: measured Error at top, normalized target and output between -1 and 1 at the middle , and error histogram at the bottom when the test data is gathered from the dumbbell topology with three flows

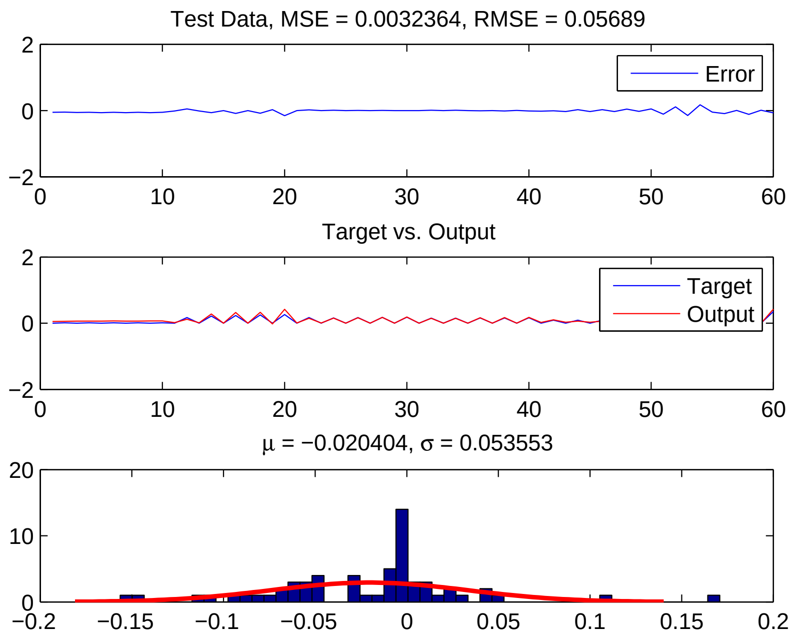


Figure 43: measured Error at top, normalized target and output between -1 and 1 at the middle , and error histogram at the bottom when the test data is gathered from the dumbbell topology with nine flows

Figures 41, 42, and 43 depict the correlation between the actual number of dropped packets(target) and the predicted number of dropped packets(output) considering the new networking scenarios as described in the last section. The pictures show in order, the MSE and RMSE values at the top, the normalized target and output values at the middle, and the error histogram at the bottom when the test datasets alter with the gathered data from the mentioned scenarios.
Although increasing the number of flows attached to the bottleneck has reduced the performance of the neural network and accordingly increased the MSE and RMSE values, the error is still negligible. On the other hand, the large center peak of error histogram graphs also indicates very small errors and implies that the output value is adequately close to the targeted values.
Figures 44, 45, and 46 are regression models of the neural network when test datasets differs in order from the baseline topology, the dumbbell topology with 3 flows and with 9 flows. These graphs visualize that how well the predicted value fits the actual output data and outlines the influence of increasing number of flows in predicting number of dropped data packets. It can be observed that by increasing the intricacy of the network topologies the accuracy of the predictor has been declined. However, the results are still valuable for our purpose and the neural network congestion control measures will justify implementing this data mining predictor technique. In the next section we will explain how we have gained benefit from the neural network and provide the implemented congestion control measures.

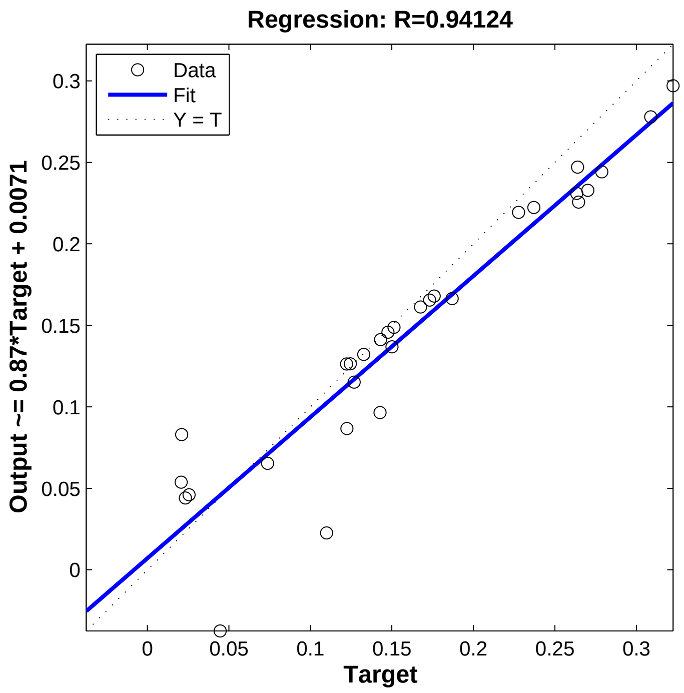


Figure 44: Correlation between target and output values when the test data is gathered from the baseline topology

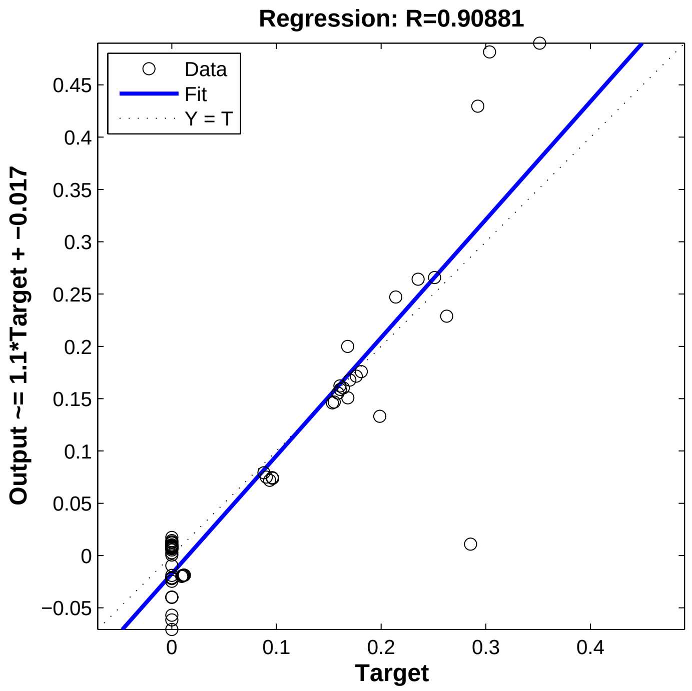


Figure 45: Correlation between target and output values when the test data is gathered from the dumbbell topology with three flows

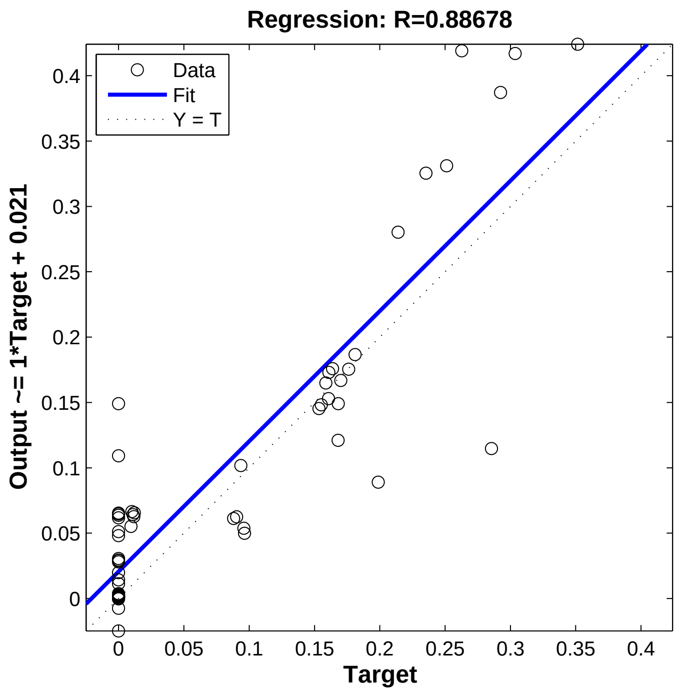


Figure 46: Correlation between target and output values when the test data is gathered from the dumbbell topology with nine flows

6.3 Simulation results

This part is dedicated to presentation of simulation results. We start by analysing performance over the simple, classic single-bottleneck dumbbell topology described in the previous chapter. Although it is simple and it does not model the richness of real-world network paths, it gives a decent indication of behaviour of congestion control algorithms. It is a valuable topology to investigate because in practice there are many single-bottleneck paths experienced by any computer networking flows.
In this section we implemented the trained neural network in the bottleneck routers and at every time instance modified the outgoing Interest rate according to the predicted dropped data packets. It should be considered that due to using simulation environment we were able to alter the transmission rate without sending a notification to the downstream nodes. It means that in these experiments we examine the use of early detection in its ideal form where congestion notification is free and causes no packet losses. However, In case of applying this technique in the real world networking, we have to consider the influence of sending notification to downstream nodes on congestion issue.

6.3.1 Interest transmission rate

This section depicts the Interest transmission rate of the first router when the proposed congestion control algorithm has been implemented in the networking models.

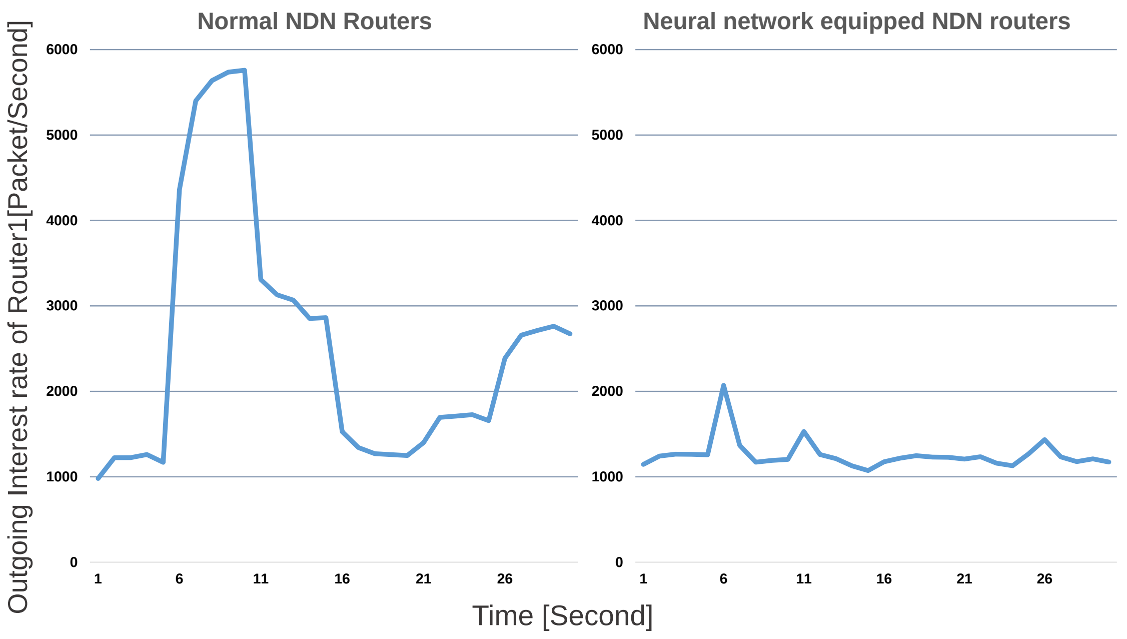


Figure 47: Outgoing Interest transmission rate comparison of the baseline topology with and without congestion control algorithm

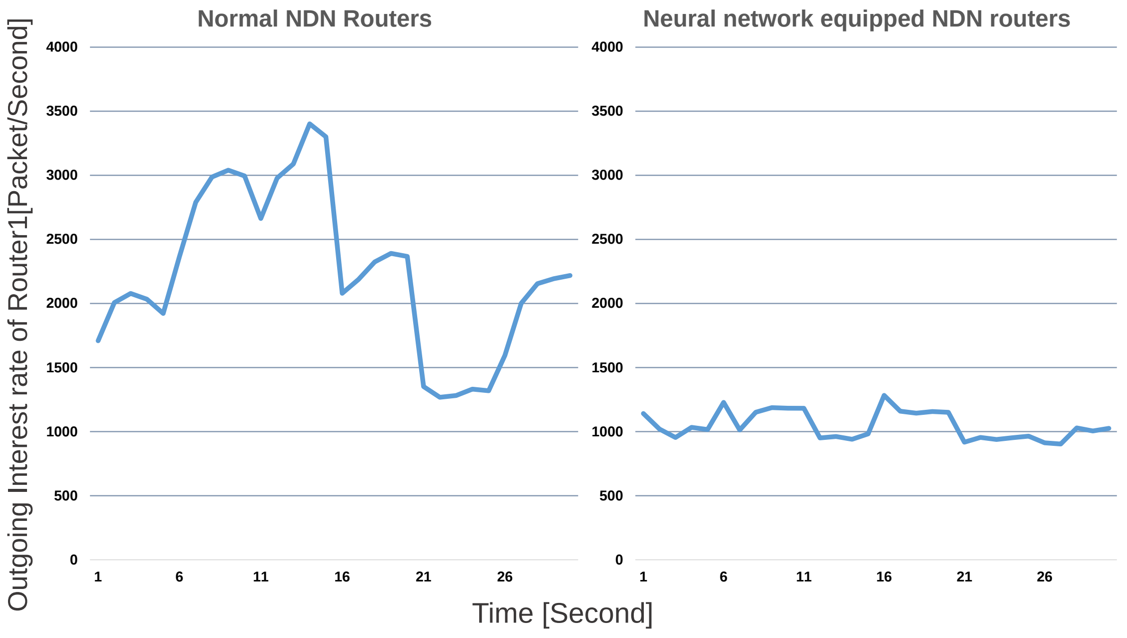


Figure 48: Outgoing Interest transmission rate of the dumbbell topology with three flows with and without congestion control algorithm

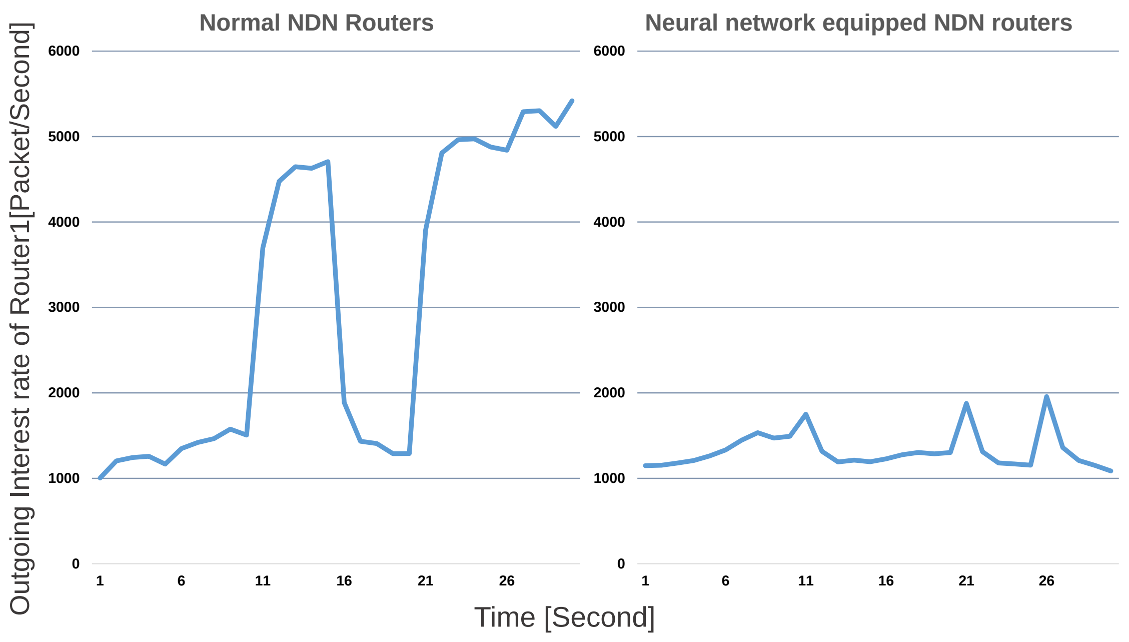


Figure 49: Outgoing Interest transmission rate of the dumbbell topology with nine flows with and without congestion control algorithm

Figures 47, 48, and 49 illustrate the discrepancy between the number of outgoing Interest from the first router both in absence and presence of the neural network rate adapter. It can be observed that regardless of the structure of the networking scenarios, the neural network Interest adapter attempts to keep the transmission rate in proximate of the full bottleneck utilization. On the other hand, by comparing the graphs it is deducible that whenever there is a sudden change in the transmission rate, the Interest rate adapter has some difficulties to adapt the rate close to its optimum values. However, the adapter approaches to the desired value very soon. It is worth to highlight again that due to having better understanding of the neural network performance, we adopted an approach in which the Interest transmission rate of the networking models alters in every five seconds.

6.3.2 Packet loss rate

As discussed earlier, one of the ultimate goals of every congestion control algorithm is, to decrease the packet loss rate. However, this doesn't mean that no packet loss cannot guarantee the high performance of a congestion control scheme. We must make sure that bottleneck link utilization also reaches a certain value. Figures 50, 51, and 52 compares the number of dropped data packets on the bottleneck for the three mentioned scenarios when the neural network rate adapter has been implemented with the condition that there is no congestion control algorithm adopted.

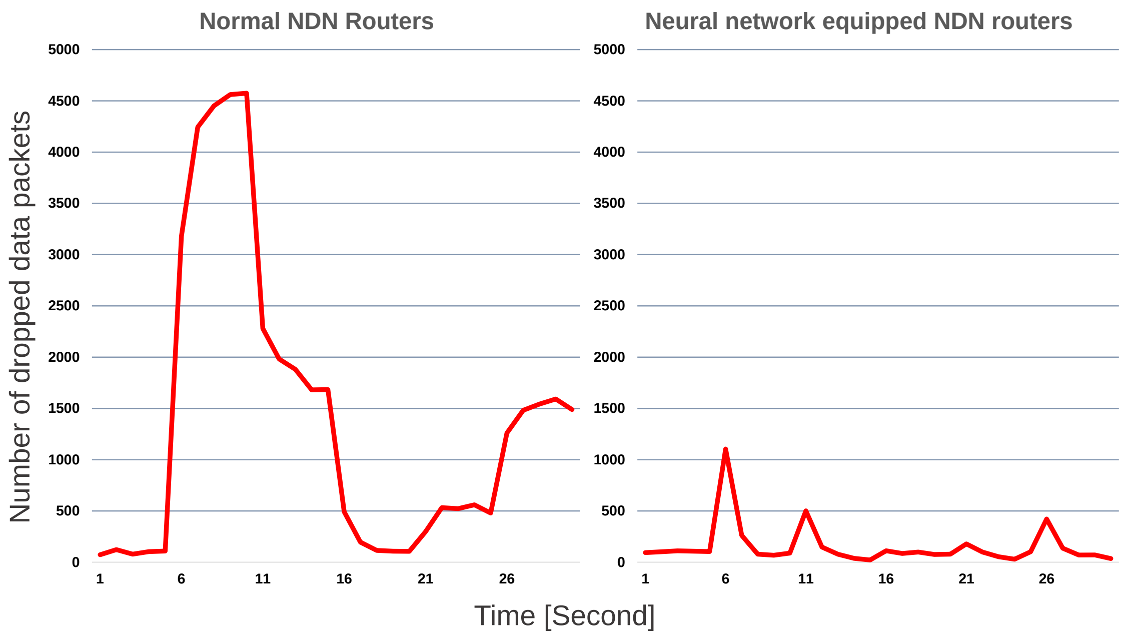


Figure 50: Drop rate Bottleneck comparison of the baseline topology with and without congestion control algorithm

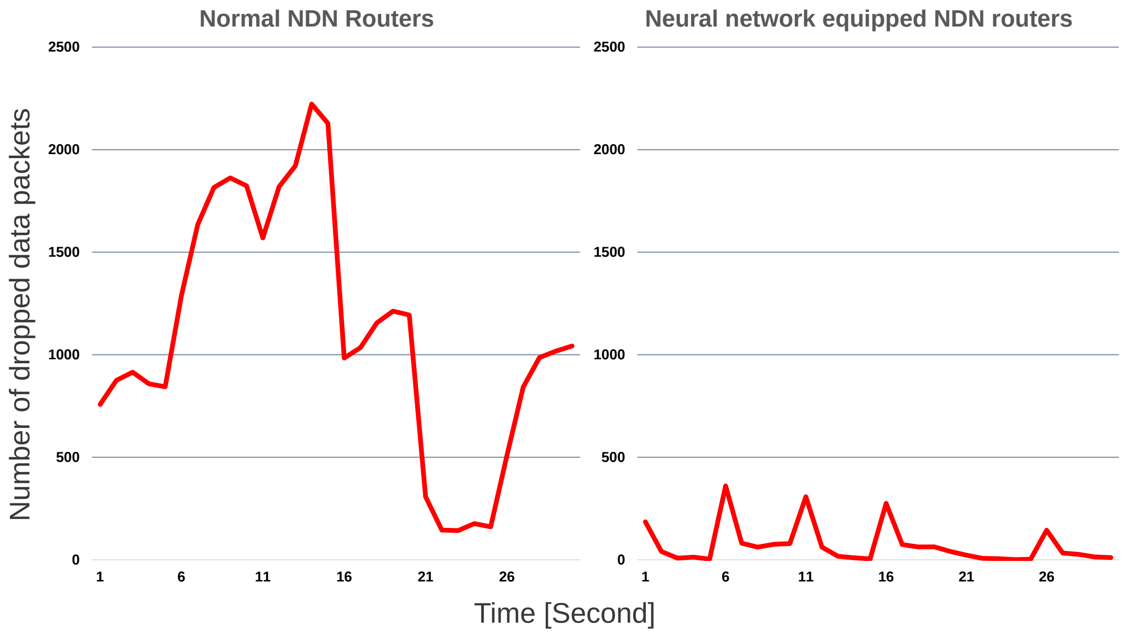


Figure 51: Drop rate Bottleneck comparison of the dumbbell topology with three with and without congestion control algorithm

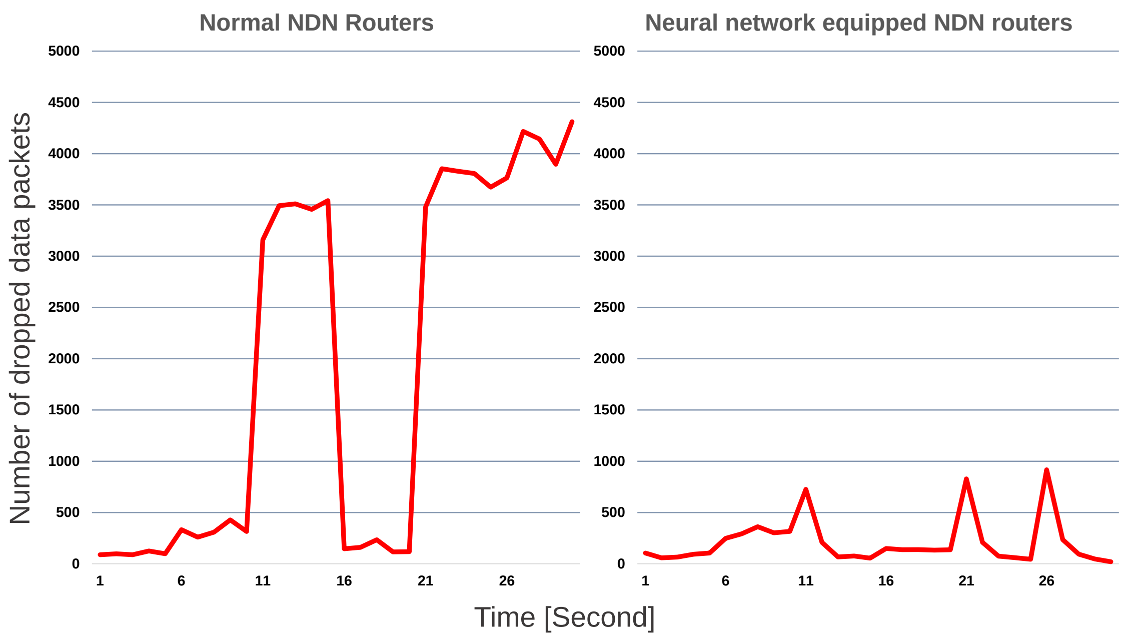


Figure 52: Drop rate Bottleneck comparison of the dumbbell topology with nine with and without congestion control algorithm

It can be seen that, by implementing the neural network rate adapter the number of dropped data packets has been drastically decreased and approached to zero. However, it is observable that when there is more connections attached to the bottleneck the number of dropped data packets are not as satisfactory as we have tested the simpler topologies. The reason for which this incident happens refers to the fact that with nine flows attached to the bottleneck the neural network accuracy in prediction is lower than other cases.

6.3.3 Throughput

The throughput is shown to verify the ability that the neural network can effectively utilize the network capacity. Figure 53, 54, and 55 depict that in case of using no data mining algorithm the bottleneck throughput is close to the theoretical value which is 1250 Kilobyte per second for the bandwidth capacity of 10 Megabits per second. It can be observed that when the number of active connection grows, the throughput undergoes more fluctuations and it will be maintained under the desired value which is 1250 Kilobytes per second.
By considering the packet drop rate graphs on one hand, and throughput on the other hand it can be concluded that our proposed neural network based algorithm has effectively controlled data congestion, and achieved near-optimal data throughput with near-zero packet loss across all the test cases we have simulated considering mentioned presumptions.

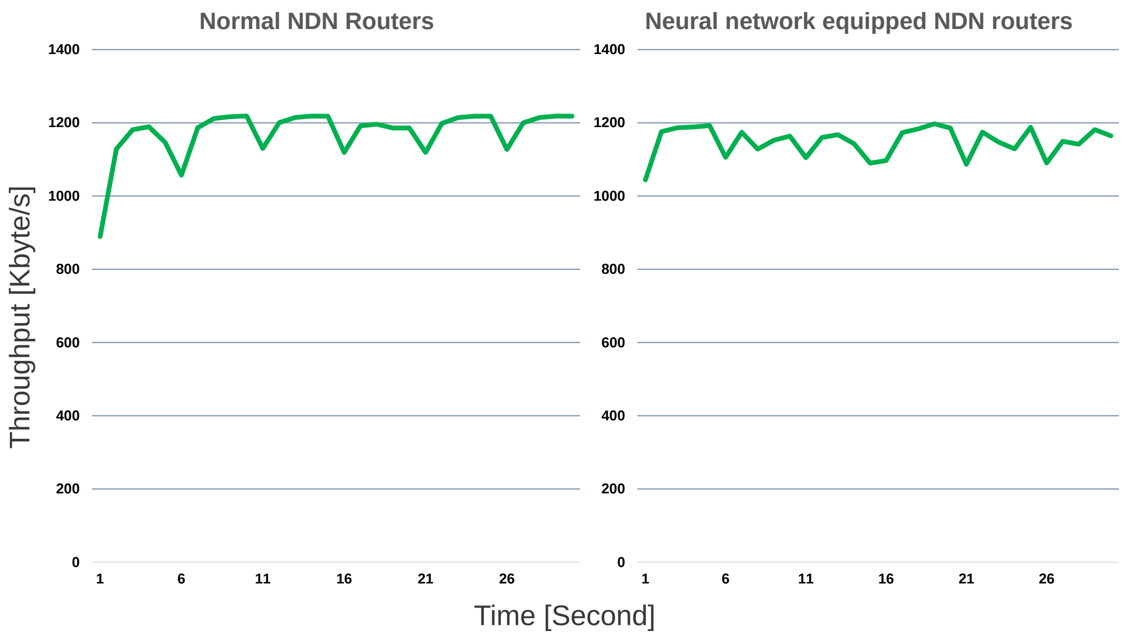


Figure 53: Bottleneck throughput comparison of the baseline topology with and without congestion control algorithm

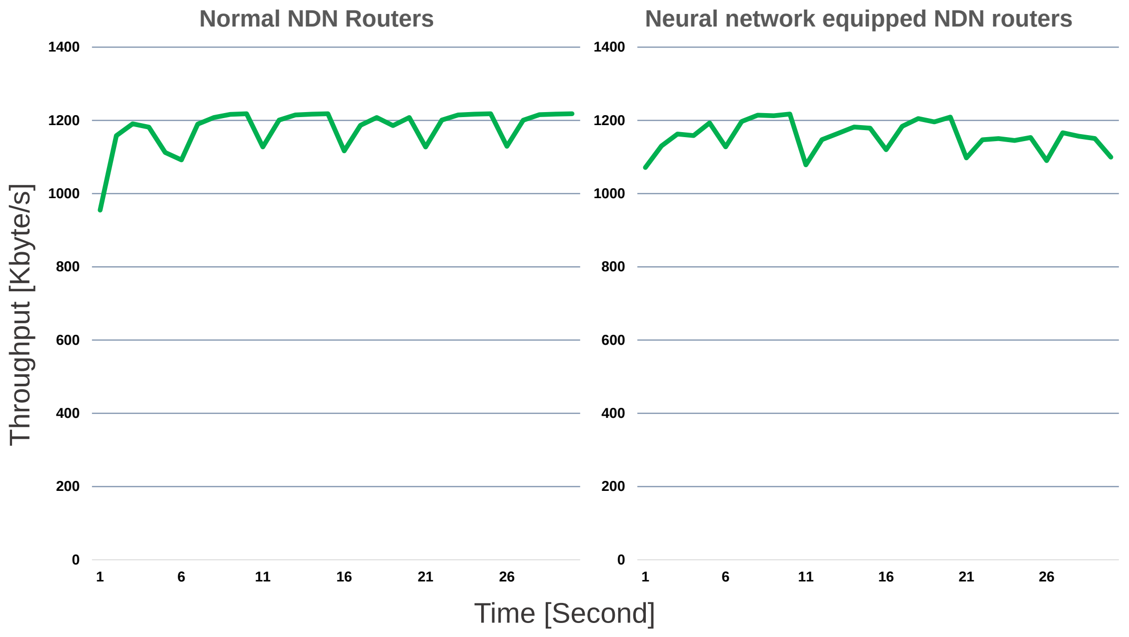


Figure 54: Bottleneck throughput comparison of the dumbbell topology with three flows with and without congestion control algorithm

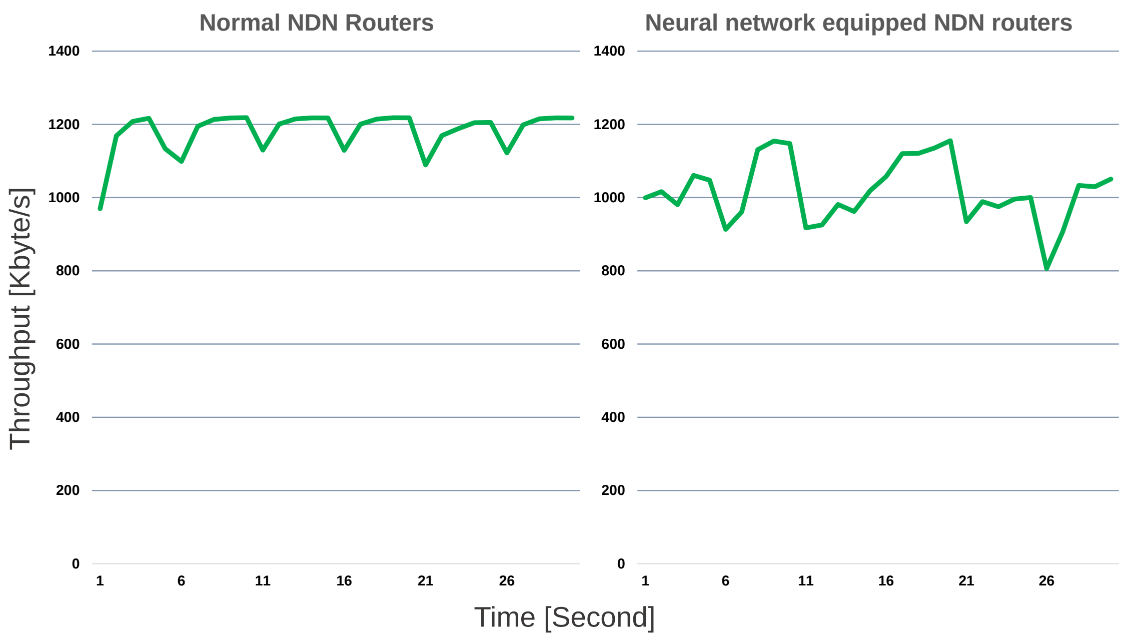


Figure 55: Bottleneck throughput comparison of the dumbbell topology with nine flows with and without congestion control algorithm

6.4 Summary

This chapter presented the proposed congestion control algorithm and provided some measures of to demonstrate its proficiency in control of congestion in content centric networkings. The main idea of the congestion control scheme accompanied with its assumptions were described in the first section. The next section provided three new networking models to validate the accuracy of the trained neural network in predicting the number of dropped data packets at the bottleneck. The last section assessed the performance of the implemented algorithm in the routers by comparing the throughput and packet drop rates of the bottleneck in the absence and presence of the suggested congestion control mechanism.

7 Conclusions and Future work

This chapter describes several avenues of future work providing suggesting solutions. Moreover, a conclusion of the research conducted and some of the difficulties that have been run into during this project are presented. Finally, major contributions of the thesis are outlined, followed by feasibility of implementation of the presented work in the real world networking.

7.1 Concluding remarks

In this thesis work the design and implementation of a novel congestion control algorithm for content centric networking were presented. The main idea is to early detection of congestion based on predicting the number of discarded data packets using neural network algorithms. Moreover, the effectiveness of the proactive congestion detection and notification in decreasing the packet loss rate and keeping the link throughput close to its optimal value simultaneously was presented.
Our results demonstrate the great potential of the proposed algorithm to control congestion in any information centric networking and even in the current Internet transport paradigm. Since every router gathers data from its local resources, training a neural network based on the collected datasets would not be a difficult task. However, the feasibility of implementing neural network techniques in any particular networking architecture needs to be assessed according to its specific attributes. We hope that our proposed technique will encourage further exploration of interacting data mining schemes in control and avoidance of congestion on various computer networking transport models.
There are many aspects of early detection congestion control on CCNs that still need to be addressed. In this thesis, only very simple networks were analysed. The analysis of this algorithm in large networks need to be carried out. Since CCN itself is in the early stage of research and it has not been widely examined in the real world networks, it is not an easy task to measure the feasibility of implementing the proposed congestion control algorithm. Therefore, the question of using neural network for Interest transmission rate adaptation remains unanswered.
In summary, the proposed algorithm for controlling congestion in content centric networking is based on the idea of early detection of congestion using neural network technique in every router. Therefore, it can be categorized as a hop by hop proactive congestion control algorithm that regulates the Interest transmission rate of every node in accordance with the number of discarded data packets predicted by the supervised learning algorithm of the neural networks.

7.2 Future work

It was observed that when the neural network collects necessary information, it could predict with high precision the probability of congestion based on the number of dropped packets. This technique will provide multiple options for routers to

decide to respond to the impending congestion in advance. An important area to explore is to consider more sophisticated data mining algorithms and to examine more mature networking topologies to enrich the detection algorithm efficiency.
The results presented in this thesis point to a number of possible areas for future work. One of the main promising future work avenue would be to consider the option of adaptive forwarding algorithm based on the predicted number of packet discards and moreover its combination with proposed Interest shaping algorithm based on neural network.

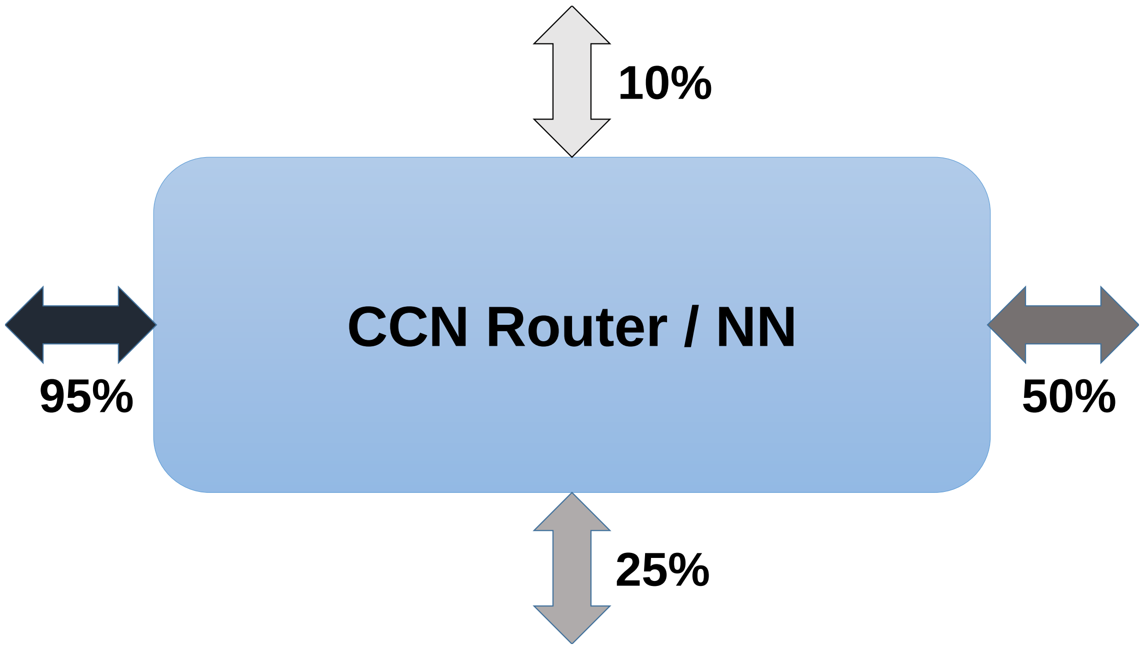


Figure 56: Data packet drop rate on each face of NDN router predicted by the neural network. The face with 95 percent and the one with 10 percent probability of data packet drops show in order the worst and the best possible route for forwarding Interest packets.

The figure 56 depicts a hypothetical CCN router with four attached faces. It shows that the neural network has already predicted the number of dropped data packets for the next time instance. It illustrates that a neural network equipped router can have a clearer understanding of impending congestion in their attached interfaces. In this thesis we merely considered an Interest shaping algorithm which is being regulated by the prediction notification. For instance in this figure the router sends a notification to each face and asks them to shape their Interest transmission rate by the prediction number accordingly. A very considerable future work would be to adopt an approach to benefit from the predicted packet discard rates by coupling them with FIB databases of each router. It would simply imply the adaptive forwarding technique based on the predicted number of dropped data packets and choosing possible routes adaptively.

When a trained neural network implemented in a router detects that a link is going to reach to its load limit, it will automatically try other available links to forward the Interests. In other words, the neural network can provide sophisticated alternatives paths for FIB, and consequently the router would be able to opt the best possible route to forward interests. PIT makes sure that the reverse path for content data is the same with the route of interests towards producers. Thus, it is a crucial decision for routers to select the least probable congested link to forward interests. This technique realizes this goal with an intelligent and precise decision algorithm. In case of high probability of congestion in all the available links, the router will send feedback to downstream routers, which will try their alternative paths.
The more mature approach to control congestion in CCNs would be the combination of the adaptive forwarding technique outlined above and the neural network Interest shaping algorithm presented in this thesis work. The approach can make a huge impact on the link utilization especially in extreme cases of high probability of packet discards. It was discussed that the neural network regulates the Interest transmission rate of downstream nodes based on the number of predicted discarded data packets. The interaction of these algorithms can be realized by defining a threshold point for the routers. It means that, we need to define that when would be the right time to alter between adaptive Interest rate algorithm and the adaptive forwarding scheme.

In order to have a better imagination of suggested algorithms, figure 57 has been provided. This figure is a snapshot picture of running CCN scenario and illustrates a networking topology that has four bottlenecks in the center by which three consumers in the left can send their Interests to the data providers in the right.

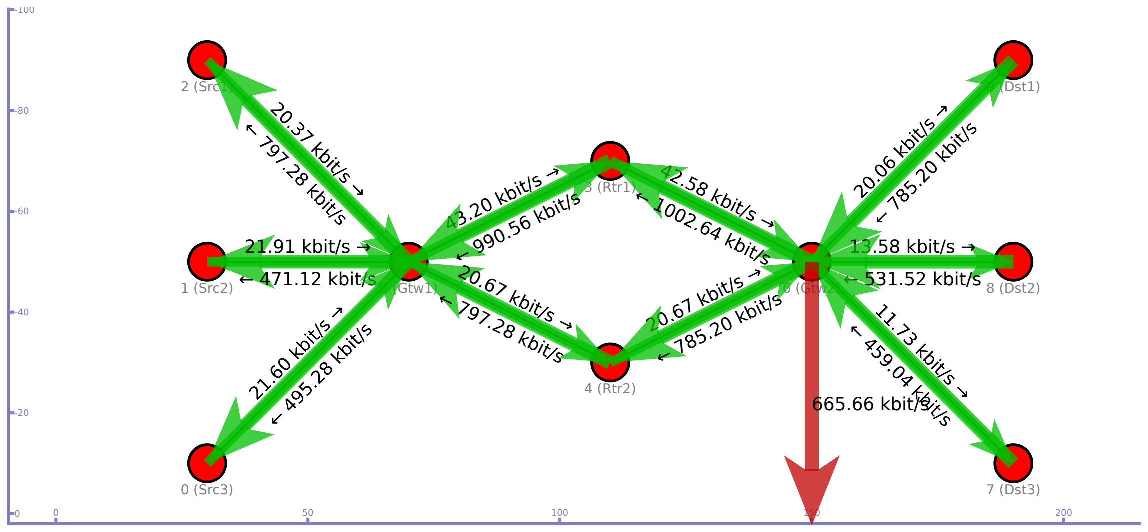


Figure 57: Variable link capacity scenario with four bottlenecks

The adaptive forwarding algorithm alone suggests that the router(Gtw1) close to the consumer sides can make decisions to forward Interests to each of its upstream nodes based on the notification packets sent by routers(Router1 and Router2) in the center. The difference between this algorithm and its interaction with shaping Interest algorithm lies in the forwarding stage. The combined algorithms can benefit from altering between adapting Interest rate and choosing the least congested path based on some assumptions, while the adaptive forwarding algorithm or even the proposed shaping Interest algorithm have less options to consider.

References

[1] V. Jacobson, D. Smetters, J. Thornton, M. Plass,N. Briggs, and R. Braynard. "Networking named content". In Proc. of ACM CoNEXT 09.

[2] M. Handley,"Why The Internet Only Just Works" BT Technology Journal, Vol 24, No 3, July 2006.

[3] J. Rexford and C. Dovrolis, "Future Internet architecture: clean-slate versus evolutionary research," Communications of the ACM, vol. 53, no. 9, pp. 36:40, 2010.

[4] Carofiglio, G.; Gehlen, V.; Perino, D.; , "Experimental Evaluation of Memory Management in Content-Centric Networking," Communications (ICC), 2011 IEEE International Conference on , vol., no., pp.1-6, 5-9 June 2011 doi: 10.1109/icc.2011.5962739.

[5] Koponen T., Chawla M., Chun B.-G., Ermolinskiy A., Kim K. H., Shenker S., and Stoica I. "A data-oriented (and beyond) network architecture". In SIGCOMM, 2007

[6] B.Ahlgren,Ch.Dannewitz,C.Imbrenda,D.Kutscher,and B.Ohlman.A Survey of Information-Centric Networking (Draft). In Bengt Ahlgren, Holger Karl, Dirk Kutscher, Brje Ohlman, Sara Oueslati, and Ignacio Solis, editors, Information-Centric Networking, number 10492 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2011. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany

[7] Rozhnova, N.; Fdida, S., "An effective hop-by-hop Interest shaping mechanism for CCN communications," Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on , vol., no., pp.322,327, 25-30 March 2012

[8] S.Tarkoma,M.Ain,K.Visala "The Publish/Subscribe Internet Routing Paradigm (PSIRP):Designing the Future Internet Architecture", 2009

[9] G.Carofiglio, M.Gallo, and L.Muscariello. "Icp: Design and evaluation of an interest control protocol for content-centric networking". In Proc. of IEEE INFOCOM NOMEN Workshop, 2012. Technical Report available at http://perso.rd.francetelecom.fr/muscariello

[10] G.Carofiglio, M.Gallo, and L.Muscariello. "Joint hop-by-hop and receiver-driven interest control protocol for content-centric networks".ICN12, August 17, 2012, Helsinki, Finland.

[11] Giovanna Carofiglio, Massimo Gallo, Luca Muscariello, Michele Papalini, "Multipath Congestion Control in Content-Centric Networks" in IEEE NOMEN Workshop, co-located with INFOCOM 2013, Turin, Italy.

[12] Cheng Yi,Alexander Afanasyev ,Lan Wang ,Beichuan Zhang ,Lixia Zhang,"Adaptive forwarding in named data networking", ACM SIGCOMM Computer Communication Review,Vol.42,N.3,July 2012

[13] Oueslati, S.; Roberts, J.; Sbihi, N., "Flow-aware traffic control for a content-centric network," INFOCOM, 2012 Proceedings IEEE , vol., no., pp.2417,2425, 25-30 March 2012

[14] "APNIC IPv4 Address Pool Reaches Final /8". APNIC. 15 April 2011.

[15] Xylomenos, G.; Ververidis, C.; Siris, V.; Fotiou, N.; Tsilopoulos, C.; Vasilakos, X.; Katsaros, K.; Polyzos, G., "A Survey of Information-Centric Networking Research," Communications Surveys and Tutorials, IEEE , vol.PP, no.99, pp.1,26,

[16] M. Gritter and D. R. Cheriton, "An architecture for content routing support in the Internet," in USENIX Symposium on Internet Technologies and Systems (USITS), 2001.

[17] T. Koponen, M. Chawla, B. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, "A data-oriented (and beyond) network architecture," in ACM SIGCOMM, 2007, pp. 181?192.

[18] FP7 PSIRP project.[Online]. Available: http://www.psirp.org/

[19] FP7 PURSUIT project.[Online].Available: http://www.fp7pursuit.eu/PursuitWeb/

[20] FP7 SAIL project.[Online]. Available: http://www.sail-project.eu/

[21] FP7 4WARD project.[Online]. Available: http://www.4ward-project.eu/

[22] FP7 COMET project.[Online]. Available: http://www.comet-project.org/

[23] FP7 CONVERGENCE.[Online]. Available: http://www.ictconvergence.eu/

[24] ANR Connect project.[Online]. Available: http://anr-connect.org/

[25] Named Data Networking project.[Online]. Available: http://www.named-data.net/

[26] Content Centric Networking project.[Online]. Available: http://www.ccnx.org/

[27] Mobility First project.[Online]. Available: http://mobility?rst.winlab.rutgers.edu/

[28] Delay-Tolerant Networking. Available: http://tools.ietf.org/html/rfc4838

[29] Yang, C.-Q.; Reddy, A.V.S., "A taxonomy for congestion control algorithms in packet switching networks," Network, IEEE , vol.9, no.4, pp.34,45, Jul/Aug 1995

[30] Jain, R., "Congestion control in computer networks: issues and trends," Network, IEEE , vol.4, no.3, pp.24,30, May 1990

[31] Nagle, J., "On Packet Switches with Infinite Storage," Communications, IEEE Transactions on , vol.35, no.4, pp.435,438, Apr 1987

[32] Z.Haas, "Adaptive Admission Congestion Control," ACM, SIGCOMM'88, 1991

[33] Comer, D.E.; Yavatkar, R.S., "A rate-based congestion avoidance and control scheme for packet switched networks," Distributed Computing Systems, 1990. Proceedings., 10th International Conference on , vol., no., pp.390,397, 28 May-1 Jun 1990

[34] A. Demers , S. Keshav , S. Shenker, "Analysis and simulation of a fair queueing algorithm", Symposium proceedings on Communications architectures and protocols, p.1-12, September 25-27, 1989, Austin, Texas, United States

[35] Lam, S.; Reiser, M., "Congestion Control of Store-and-Forward Networks by Input Buffer Limits–An Analysis," Communications, IEEE Transactions on , vol.27, no.1, pp.127,134, Jan 1979

[36] Floyd, S., "TCP and Explicit Congestion Notification," ACM Computer Communication Review, V. 24 N. 5, October 1994, p. 10-23.

[37] M.Craven,J.Shavlik,"Using Neural Networks for Data Mining," Future Generation Computer Systems, 13, pp. 211-229 ,1997

[38] [Online]. Available : http://www.ccnx.org/doc/technical/CCNxProtocol.html

[39] [Online]. Available : http://www.utexas.edu/research/asrec/neuron.html

[40] Simon Haykin,"Neural Networks a comprehensive foundation", Second Edition ,1999

[41] Jacobson, Van, Karels, MJ (1988). "Congestion avoidance and control". ACM SIGCOMM Computer Communication

[42] R. Jain, "A Timeout Based Congestion Control Scheme for Window Flow-Controlled Networks," IEEE Journal of Selected Areas in Communications, Vol. SAC-4, No. 7, October 1986, pp. 1162-1167.

[43] Jacobson, Van., "Modified TCP Congestion Avoidance Algorithm," end2end-interest mailing list, April 30, 1990.

[44] J.Kurose, K.Ross.,"Computer Networking,A Top-Down approach," Fifth Edition

[45] M. Allman, V. Paxson, and W. Stevens. "TCP Congestion Control," April 1999, RFC 2581.

[46] Requirements for Internet Hosts – Communication Layers.[Online]. Available: http://tools.ietf.org/html/rfc1122

[47] Craig Hunt.,"TCP/IP Network Administration,"ISBN 1-56592-322-7, 630 pages. Second Edition, December 1997.

[48] Terminology for FIB based Router Performance.[Online]. Available: http://tools.ietf.org/html/rfc3222

[49] Ethem Alpaydin.,"Introduction to Machine Learning," Second Edition, The MIT press Cambridge, Massachusetts , London , England.

[50] Lipo Wang, Xiuju Fu "Data Mining with Computational Intelligence," 2005

[51] Steven W. Smith., "The Scientist and Engineer's Guide to Digital Signal Processing," 1997

[52] D. Kriesel., "A Brief Introduction to Neural Networks," 2005

[53] Kevin L. Priddy, Paul E. Keller "Artificial Neural Networks: An Introduction,"

[54] Michael Welzi, "Network Congestion Control: Managing Internet Traffic"

[55] A. Afanasyev, I. Moiseenko, and L. Zhang, "ndnSIM: NDN simulator for NS-3", Technical Report NDN-0005, 2012

[56] Tsai, Pearl T,."Parallel network simulation techniques," Massachusetts Institute of Technology. Dept. of Electrical Engineering, 1995

[57] H. Dai, B. Liu, Y. Chen, and Y. Wang, "On pending interest table in named data networking," in Proceedings of ACM/IEEE ANCS, Austin, Texas, USA, Oct 2012, pp.211,222.

[58] L.Peterson, D.Davie, "Computer Networks", A systems approach," Third Edition

[59] S. Rajasekaran and P.Vijayalakshmi, "Neural networks,Fuzzy Logic and Genetic Algorithms",New Delhi, Prentice Hall of India, 2004.

[60] Raul Rojas, "Neural Networks - A Systematic Introduction" , Springer-Verlag, Berlin, New-York, 1996 (502 p.,350 illustrations).

[61] Menlo Park, "Lessons in Neural Network Training: Overfitting Maybe Harder than Expected," Proceedings of the Fourteenth National Conference on Artificial Intelligence, AAAI-97, AAAI Press, California,pp.540,545,1997.

[62] Michael J.Pan, "Advanced Statistics from an Elementary Point of View", 2005

[63] [Online]. Available : http://www.merriam-webster.com/