Efficient Congestion Management Framework using Compression Techniques

by

KHO Lee Chin

submitted to
Japan Advanced Institute of Science and Technology
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

Supervisor: Associate Professor Yuto LIM

School of Information Science
Japan Advanced Institute of Science and Technology

June, 2015

Abstract

In this dissertation, congestion control management mechanisms are investigated to introduce a new direction for aiding in solving network congestion. Existing mechanisms that directly or indirectly help to eliminate the congestion problem were being investigated.

The well-known empirical model of TCP connection throughput model has been extended in this thesis to include the effect of compression of part of the traffic throughput. The derived model can help to estimate the performance of congested networks when compression is applied.

The proposed generic ECM framework can be applied to minimize the impact of network congestion by orchestrating different existing congestion management mechanism with the newly introduced compressed MPLS mechanism. The ECM framework presented in this research offers an overall idea on how to help in improving congested networks throughput by reducing the impact of network congestion. In other words, ECM framework can indirectly help congestion points or links of the entire network (Internet). ECM framework can be re-engineered to cooperate with other possible mechanisms (i.e., network coding) for further reduction of network congestion.

The ECM/C model is presented in this research which offers an overall idea on how to utilize compression in the networking model to help in improving congested networks throughput for limited resource devices.

Acknowledgments

First of all, I would like to express my sincere gratitude to my research supervisor, Associate Professor Dr. Yuto Lim and Professor Dr. Yasuo Tan for their patient guidance, inspiring instruction and warm encouragement throughout this research endeavour. Their brilliant ideas and suggestions helped me to improve the presentation of this thesis.

I am grateful to Associate Professor Dr. Défago Xavier for his guidance and useful suggestions, which help me in completing the sub-theme research.

My very special thanks go to Professor Dr. Yoichi Shinoda and Professor Dr. Tadashi Matsumoto for his valuable advice and friendly help. Their extensive discussion and constructive comments have been very helpful for this study.

I would like to take this opportunity to express my gratitude to the member of examination committee of my doctoral dissertation, Association Professor Dr. Tan Chong Eng from Universiti Malaysia Sarawak, Professor Dr. Yoichi Shinoda and Professor Dr. Mikifumi Shikida from JAIST, for their valuable comments and suggestions to improve the quality of this thesis.

I also greatly appreciate the help and support of Thao Tran Phuong, Naushin Nower, Amr Ashimawy, Wang Lin, Chen Zuan, Ling Siew Gin, Wu Qiung, and Sarin Achawaranont who have always been there to count on.

In addition, I would like to express my gratitude to the Ministry of Education (MOE) of Malaysia for providing me financial support for this work.

Lastly, I wish to thank my parents and husband Dr. Ngu Sze Song, for their patience and support during my PhD study. I am also thankful to my lovely daughters, Kimberly Ngu Shu Yee and Kathy Ngu Shu Xuen who provided me with an additional joyful dimension to our life mission.

Contents

List of Figures

List of Tables

List of Symbols

Symbols in Chapter 4

α	The number of packets from the beginning of the TD period until and including the first lost packet (unit: packets)
β	The number of sent packets in the last round in that specific TD period (unit: packets)
λ	The average input rate of that TCP connection (unit: packets/second)
μ	The service or processing rate of that network device (unit: packets/second)
b	The slope of additive increase in a TCP congestion control scheme
p	Loss probability of sent packets
r	The duration of the round trip time of this specific round in the TD period considered (unit: seconds)
t_c	The compression time (unit: seconds)
x	The number of rounds in a TD period until and including the round where the TD occurs. It is either the number of rounds in that specific TD period or the same number minus one (unit: packets)
A	The time duration of the TD period (unit: seconds)
B	The number of packets in buffer before compression (unit: packets)
B'	The final number of packets in the buffer (unit: [packets])
R(p)	Relationship between the throughput of TCP connection (R) and loss probability (p)
RTT	Round-trip time of one packet start when the packet of TCP connection is completely transmitted from the sender until its ACK from the receiver is completely received by the sender again
T	Number of packets sent per unit time regardless the eventual fate, either lost or received (unit: packets per second)
W	The maximum window size of that specific TD period, which is the window size of the last or before last round, depending on where did the TD occurred
Y	The number of sent packets in that specific TD period

Symbols in Chapter 6

S_c	The size of encoded data (unit: byte)
S_o	The size of source data (unit: byte)
α	The size of one input symbol (unit: bits)
R	The sum of repetitions of all output symbols and codeword
P_s	The sum of repetitions of all output symbols that were not compressed
P_c	The sum of repetitions of all output codeword that have been compressed
Q	The sum of entries in the dictionary
l_{cmax}	The codeword length that has the maximum number of bits of a codeword
l_{cmin}	The codeword length that has the minimum number of bits of a codeword
$\bar{l}_c$	The codeword length that has the average number of bits of a codeword

List of Abbreviations

ABR	Available Bit Rate
ACK	Acknowledgement
Africa	Adaptive and Fair Rapid Increase Congestion Avoidance
AIMD	Additive Increase Multiplicative Decrease
AQM	Active Queue Management
ATM	Asynchronous Transmission Mode
B	Buffer size
BBN	Bolt, Beranek and Newman
BDP	Bandwidth-Delay Product
BIC	Binary Increase Congestion control
BISDN	Broadband Integrated Services Digital Network
BMP	BitMaP
CC	Congestion Control
CA	Congestion Avoidance
CAD	Connection Admission Control
CBR	Constant Bit Rate
CDV	Cell Delay Variation
CER	Cell Error Ratio
CID	Context IDentifier
CLP	Cell Loss Priority
CLR	Cell Loss Rate
CP	Compression Percentage
CR	Compression Ratio
CRTP	Compression Real Time Protocol
CTCP	Compound TCP
CTD	Cell Transfer Delay
D	Delay-based
DiffServ	Differentiated Services
DNS	Domain Name System
DOOR	Detection of Out-of-Order and Response
DSACK	Duplicate Selective ACKnowledgement
ECM	Efficient Congestion Management
EFCI	Explicit Forward Congestion Indicator
ESRT	Event-to Sink Reliable Transport
FDDI	Fiber Distributed Data Interface
FIFO	First In First Out

FR	Fast Recovery
FWestwood	Fuzzy Westwood
GFC	Generic Flow Control
HEC	Head Error Control
HSTCP	High-Speed TCP
HTTP	Hypertext Transfer Protocol
ICTCP	Incast Congestion control for TCP
IoT	Internet of Things
IP	Internet Protocol
IPHC	Internet Protocol Header Compression
ISDN	Integrated Services Digital Network
ISP	Internet Service Provider
JPEG	Joint Photographic Expert Group
L	Loss-based
LAN	Local Area Network
LBE	Loss Based Estimation
LD	Loss-based and Delay-based
LDBE	Loss-based and Delay-based with Bandwidth Estimation
LDC	Lightweight DeCompression
LP	Low Priority
LZ	Lempel-Ziv
LZAP	Lempel-Ziv-All-Prefixes
LZB	Lempel-Ziv-Bounded
LZC	Lempel-Ziv-Compress
LZH	Lempel-Ziv-Huffman
LZJ	Lempel-Ziv-Jakobson
LZMW	Lempel-Ziv-Miller-Wegman
LZR	Lempel-Ziv-Rodel
LZSS	Lempel-Ziv-Sorter-Szymanski
LZW	Lempel-Ziv-Welch
LZT	Lempel-Ziv-Ticher
MAN	Metropolitan Area Network
MAC	Media Access Control
MANET	Mobile Ad hoc NETwork
MCP	Mobile Centric Protocol
MCR	Minimum Cell Rate
MIMA	Multiplicative Increase Multiplicative Decrease
MPEG	Motion Picture Expert Group
MPLS-TE	MultiProtocol Label Switching Traffic Engineering
MSR	Molecular Sequence Reduction
MTU	Maximum Transmission Unit
NCM	Network Congestion Management

nrt-VBR	non-real Variable Bit Rate
PCR	Peak Cell Rate
PTI	Payload Type Identifier
QoS	Quality of Service
RCP	Receiver Centric Protocol
RED	Random Early Detection
RLE	Run Length Encoding
ROHC	RObust Header Compression
RSVP	Resource reservation Protocol
RTO	Retransmission timeout
RTP	Real Time Protocol
RTT	Round Trip Time
rt-VBR	real time Variable Bit Rate
SACK	Selective ACKnowledgement
SCP	Sender Centric Protocol
SF	Shannon-Fano
SMTP	Simple Mail Transfer Protocol
SNR	Signal to Noise Ratio
SSL	Secure Socket Layer
STCP	Scalable TCP
TCP/IP	Transmission Control Protocol/ Internet Protocol
TCPW BR	Transmission Control Protocol Westwood with Bulk Repeat
TCPW CRB	Transmission Control Protocol Westwood with Combined Rate and Bandwidth estimation
TD	Triple Duplicate
TD-FR	Time Delayed Fast Recovery
TDP	Triple Duplicate Period
TFRC	TCP Friendly Rate Control
TM	Traffic Management
Tri-S	Slow Start and Search
UBR	Unspecified Bit Rate
UDP	User Datagram Protocol
UNI	User Network Interface
UPC	Usage Parameter Control
VCI	Virtual Channel Identifier
Vegas A	Vegas with Adaptation
VJHC	Van Jacobson's Header Compression
VoIP	Voice over Internet Protocol
Veno	VEgas and ReNo
VPI	Virtual Path Identifier
W	Window Size
WAN	Wide Area Network
WSN	Wireless Sensor Network
XCP	eXplicit Control Protocol

Chapter 1

Introduction

This chapter first briefly introduces the congestion problems in the communication networks. The background of some of the existing congestion control techniques are presented here together with the research motivation. The objective and contribution of this research are also provided, which is followed by the dissertation outline.

1.1 Congestion Problems in Communication Networks

A computer network is collection of autonomous computers interconnected by a single technology. Two computers are said to be interconnected if they are able to exchange information. The data in some communication networks is transmitted in the form of messages, which could be referred to as packets during the transmission process. These packets are sent across the network based on the communication protocols, and reconstructed at the destination [1].

The size and complexity of communication network can be classified into four main types: small network, Local Area Network (LAN), Metropolitan Area Network (MAN), and Wide Area Network (WAN). Small networks commonly connect the sub-assemblies or devices, while LAN connects the distributed terminals and computer equipment in restricted areas, such as University campuses. MAN is a high speed network that is used to interconnect LANs in a geographical region, such as within a city. Meanwhile, WAN is multiple communication connections for a large geographical area, for example, Internet is connected via a large global network of service providers using the servers, modems, routers and switches.

Internet has had a big influence on human daily life for years. It is used at home for personal usage entertainment, communication, banking, shopping, and devices control. It is also used at work as professional tool. Fig. 1.1 illustrates the growth of Internet based on the time-line from year 2004 to 2015 [2]. One important time-stone is in 2010, when mobile broadband connections exceeded fixed broadband connection. The time-line also shows that the number of Internet users increased from one billion in 2005 to almost three billion by 2015.

The rapid growth of Internet users increases traffic loads, at the same time, networks are targeting full utilization to maintain revenue and service pricing. Consequently, the problem of network congestion is likely to occur in certain situations. Network congestion is one of the unending problems that can occur when capacity of an underlying sub-network is insufficient for the demanded amount of data. The growth of demand

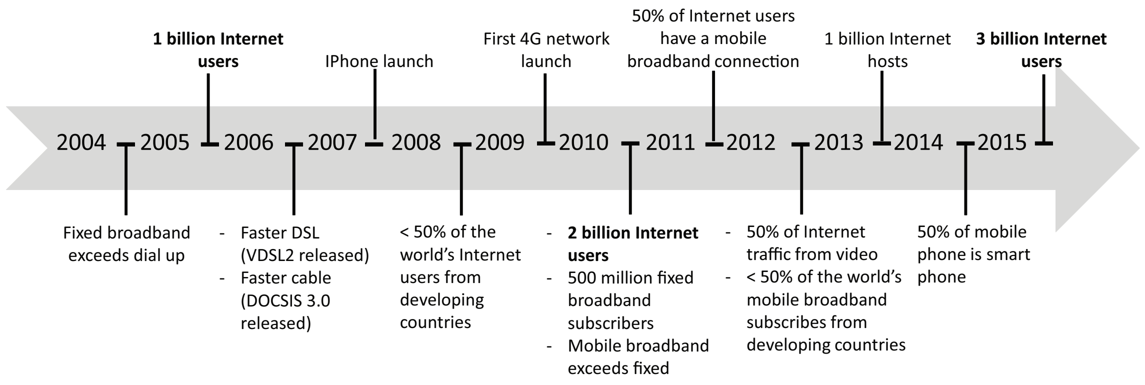

1 billion Internet users
IPhone launch
First 4G network launch
50% of Internet users have a mobile broadband connection
1 billion Internet hosts
3 billion Internet users
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
Fixed broadband exceeds dial up
- Faster DSL (VDSL2 released)
- Faster cable (DOCSIS 3.0 released)
< 50% of the world's Internet users from developing countries
- 2 billion Internet users
- 500 million fixed broadband subscribers
- Mobile broadband exceeds fixed
- 50% of Internet traffic from video
- < 50% of the world's mobile broadband subscribes from developing countries
50% of mobile phone is smart phone

Figure 1.1: The milestone of Internet development

will eventually go beyond the Service Provider's ability to efficiently cope with the huge data traffic. As a result, the network might face tremendous and unpredictable network congestion. When network congestion occurs, the quality of service (QoS) and energy efficiency in the network will degrade. In other words, if traffic load ever goes beyond the Internet's capacity, Internet will face tremendous and unpredictable network congestion. When a resource (buffer or bandwidth) is shared with multiple users who contend to access that resource beyond its availability, network congestion may occur. In other word, the network congestion can be defined as when the incoming traffic demand to some sub-network (node or link) exceeds the capacity of that sub-network (buffers and output capacity), congestion may occur.

Network congestion occurs due to various reasons. It can be generally categorized into predictable events and unpredictable events or alternatively into random congestion and recurrent congestion. Predictable events generate additional Internet demand to the already existing traffic load, like when users accessing global online debut of big branded products. Unexpected life threatening events such as floods, tornadoes, and earthquakes can increase the network traffic erratically and/or suddenly remove vital network resources. This kind of unpredictable events can last for few hours, days or weeks. Random congestion can occur when a number of users are sharing a sub-network and all become highly active within a very short period of time. Random congestion is mainly due to bad design of network resource that is over-utilized above the safe statistical nature of traffic. Finally, recurrent congestion occurs due to the normal pattern of daily activities that create an overall significant traffic load in recurrent time. For instance, higher usage on business hours for commerce and trading servers, similarly higher usage of residential users in the evening to entertainment service providers.

During congestion, the quality of service (QoS) in Internet will be degraded. When a network is congested, the delay time and packet loss increase, and the throughput decreases. The delay time might increase due to packets suffering from long queuing till timing-out. For example, if the worst case for satellite network transmission is 160 ms round-trip, when a voice call experiences 200 ms, the talk will be overlapping. Packet loss in congestion happens when a packet is dropped because of long waiting time due to unavailability of the next buffer. Although the lost packets would be retransmitted with Transmission Control Protocol (TCP), still, the overall network throughput decreases.

1.2 Research Background and Motivation

Network congestion basically can be resolved by three possible methods: increasing physical output bandwidth, increasing physical size of local buffering and implementing better network flow management. When physical output of a network is increased, i.e., the transmission rate is increased, the problem of congestion which mainly caused by an extremely slow link would be solved. However, it is usually not that simple, when some links are upgraded without sufficient studying and planning, worse congestion can reappear in some nearby link due to bad network resources balancing.

When the capacity of buffering is insufficient, incoming packets which wait too long before being allowed to enter the buffer might experiences times-out and get dropped. If the buffer size is enlarged, more packets will be allowed to enter and stored, this can mitigate the congestion problem. However, larger buffer size can introduce longer delay again causing timing-out and congestion again.

Network congestion is a dynamic problem; the two manual methods mentioned are not always useful and definitely not responsive enough. That leaves only the last method which has been actively pursued by researchers for nearly three decades, which is the network flow management to handle congestion. In short, the congestion management. The goal of any network flow management is *to ensure that the network is utilized as efficiently as possible.* In other word, the highest possible network throughput shall be achieved while trying to avoid over-utilization and its associated problems such as congestion.

Different flow management approaches has been introduced over the years to address the problem of congestion. Congestion control handles the level of traffic entry to the system. The initial model of congestion control is based on microeconomic theory and convex optimization theory, where individuals controlling flow rates can interact to obtain an optimal network-wide rate allocation. Later many distributed network optimization algorithms extended that initial model. However, the initial model has a weakness; it assumes the entire flow is controlled by the same parameter, and different flows will be controlled by different parameters.

Congestion avoidance such as Random Early Detection (RED) and fair queuing are commonly used. In RED mechanism, the average queue size is monitored and the packets are pre-dropped according to statistical probabilities to save space for other packets and explicit messages are sent to inform the sender to decrease the transmission rate. RED is simple to implement without global synchronization and can provide high link utilization. However, the performance is highly sensitive to the controlling parameters; threshold, drop probability and weight in RED algorithm and might cause buffer overflow or slow responsiveness to congestion. Fair queuing is a scheduling algorithm which allows the capacity of the link to be fairly shared by multiple connections; it can prevent high speed data connection from swamping links of insufficient capacity. Priority schemes, is one of the network congestion mending functionalities which allow some packets to be transmitted with higher priority than others. These schemes cannot solve network congestion directly. They only help to relieve congested network through providing some services.

A wide range of network congestion mechanisms have been introduced by researchers around the world, some of current network congestion mechanisms have included fuzzy logic or neural network. Adaptive route changing algorithms have been also presented to alleviate congestion by modifying the routes to steer traffic in the most effective way.

The utilization of network coding in congestion control has also been studied. TCP Vegas with online network coding (TCP VON) uses one of the examples of congestion control mechanisms.

From the point of view of congestion control mechanisms, the term congestion control is wider than just a way to get rid of congestion, but rather to the way for better utilization of the network resources. The goal of researchers is still almost the same, which is to improve the efficiency of network resources usage while providing safety measures or margins to avoid over-utilization.

Here, I emphasize again, congestion control actually refers to the way of effectively use the network resources, not just congestion management itself. Network technologies have reached blazing speeds, Ethernet technology advanced from IEEE802.3a 10 Mbps in 1985 to IEEE802.3ba 100 Gbps in 2010 [3]. Also, wireless LAN technology increased from IEEE802.11a 54 Mbps in 1999 to IEEE802.11n 600 Mbps in 2009 [4]. Still, the congestion sometime occurs due to the unbalance load distribution.

Most of the existing congestion control mechanisms is about 'how to eliminate congestion', focusing on one or more scenarios of special cases. But congestion is a dynamic problem which might occur due to different network scenarios and events. This motivates us to propose an efficient congestion management framework, which is about 'how to better utilize the available mechanisms' by adaptive selection of the suitable congestion mechanism, trying to improve resources utilization. The research introduces a new approach for solving congestion, compression is studied to reduce the effective size of data and possibly to improve performance.

To implement compression in the proposed congestion management framework, the following research problems are carefully considered.

- Lengthy compression time even for the fastest existing hardware compression schemes.
- Highly sensitive compression degree depending on the fragile existing correlation in any data based on bot spatial and temporal locality principals of information theory.
- Enormous temporally memory requirements for compression and sometimes decompression as well.
- The conflicting nature of the mentioned above three problems, meaning improving one will definitely degrade the other two.

1.3 Research Vision and Objectives

Congestion can occur at any time and place in the Internet. Fig. 1.2 shows the possibility of congestion in the Internet. Current congestion management is usually implemented as an integral part of flow management which ensures the network resources are being fully utilized all the time. However, there will never be a perfect flow control mechanism that can make sure that the network resource is fully utilized and at the same time avoid congestion completely. This can easily be realized from the researches' continuous development of new traffic control mechanisms.

The vision of this research is to propose an efficient congestion management framework to minimize the impact of congestion, without wasting much network resources. To accomplish this vision, the research objectives are summarized as follows:

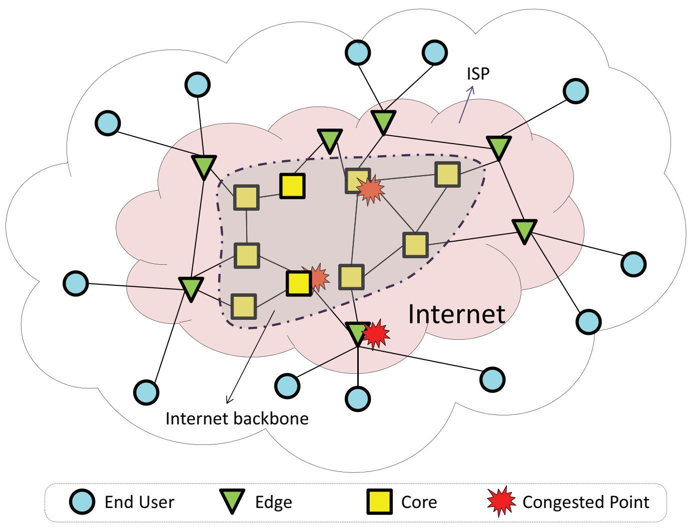


Figure 1.2: The congestion problem in the Internet

- To review the existing network congestion management, especially the ATM and TCP/IP networks.
- To derive an empirical model throughput of congested network with compression capability.
- To design an efficient congestion management framework making use of the compression.
- To propose an efficient congestion management with compression for small devices.

1.4 Summary of Contribution

This research investigates the efficiency of compression techniques in networks from the perspective of throughput. The network congestion management framework (ECM) is proposed for network devices. Although the framework model is quite simple, the work provides valuable insight contribution to the existing research on congestion management. The main contributions of this research can be summarized as follows:

- An empirical model for network throughput with compression capability to study the feasibility of compression in networks.
- A generic congestion management framework is proposed that can be applied to minimize the impact of network congestion by using the existing mechanisms coupled with compression on top of Multi-Protocol Label Switching-Traffic Engineering (MPLS-TE).
- A simple compression scheme with lightweight decompression for devices with limited resources is proposed.

Both ECM framework and ECM/C model proposed show opportunities of using compression in networking such as in the Internet. Both models can co-exist and be applied in different parts of the network simultaneously, or with limited overlapping.

1.5 Structure of the Dissertation

The dissertation is organized as follow:

Chapter 1 (this chapter) introduces the basic concept of congestion and the congestion problems in networks. The motivation and some research backgrounds are described. Also a summary of the contributions of this research are presented.

Chapter 2 reviews some of the existing network congestion management mechanisms used in the Internet. First, the devices in general networks are distinguished, and then traffic management methods are reviewed. Almost 34 of TCP variants are used to explain in-depth end-to-end congestion control.

Chapter 3 reviews some of the common existing compression schemes and provides thorough classification. Example schemes are later explained and discussed to show the suitable chances of application. Finally, the generic compression metrics are presented together with derivation of a relation between overall compression degree and part-wise compressed data compression degrees.

Chapter 4 formulates an empirical model for connection throughput of compression capable networks. This throughput model is the extension of an existing model with finite buffer. The operating curves of the model are also shown.

Chapter 5 proposes an Efficient Congestion Management (ECM) framework for networking devices to minimize the impact of network congestion when it occasionally occurs. The architecture of ECM mainly consists of a congestion classifier. The new congestion control mechanisms introduced together with the compression.

Chapter 6 introduces an ECM/C model together with Dictionary-based Lightweight DeCompression (LDC) scheme that is proven to minimize the impact of congestion in small limited resources devices. Those devices which have small bandwidth and small buffering capabilities are the main concerned here. The four stages of LDC; dictionary building, encoding, dictionary loading and decompression are explained. Analysis of the LDC scheme performance due to the affected parameters is also presented.

Chapter 7 summarizes the work in this dissertation and provides insight into the future work.

Chapter 2

Network Congestion Management

In this chapter, the background knowledge of network congestion management is discussed. Some network architectures can suffer from congestion while other designs managed to avoid any possibility of occurrence. An example of each network class is presented in this chapter. Some of the existing congestion management techniques are also introduced while talking about the congestion vulnerable class of networks.

2.1 Introduction

Network congestion can occur when different data streams flowing in from multiple links to a router, require a single resource. Since congestion can only happen in nodes with multiple input links and at least one output resource, it can happen in any device except for end devices. For example, when core Internet devices encounter more traffic than they can handle. The single resource which is competed for here might be memory like a router buffer for traffic, or bandwidth of the output link. In both cases, data experiencing prolonged delays (buffer waiting) would eventually timeout (be dropped). When reliable data is required and the network is congested, the more data being re-sent after dropping, the more time need for successful transfer. This would waste the already consumed energy, bandwidth and memory resources which that data already utilized until the point of dropping. Not to mention the additional overall delay that data would accumulate besides the retransmission delay.

Congestion can occur mainly due to one of two reasons. First, when the amount of data traffic admitted into the whole network doesn't exceed the whole network capacity for data transfer. The second is attributed to poor balance routing of data traffic across the network links. Known routing techniques are still far from perfection, thus frequently fail to balance traffic among the links. Imperfect routing causes congestion by over utilized some parts of the network for some profit (financial, political, energy ...), while other parts are underutilized.

Fig. 2.1 illustrates an example of Internet congestion. Internet is comprised of three elements: end device, edge device, and core device. An end device can be any machine with some application running. An edge device (router) provides an entry point to the lower level devices for the higher level devices. Core devices (extremely fast big routers) form the core Internet and interconnect it to lowest level edge devices (router). In Internet links, data only flows from one end device to another end device. According to what was previously mentioned, congestion in Internet would only occur in edge or core devices but

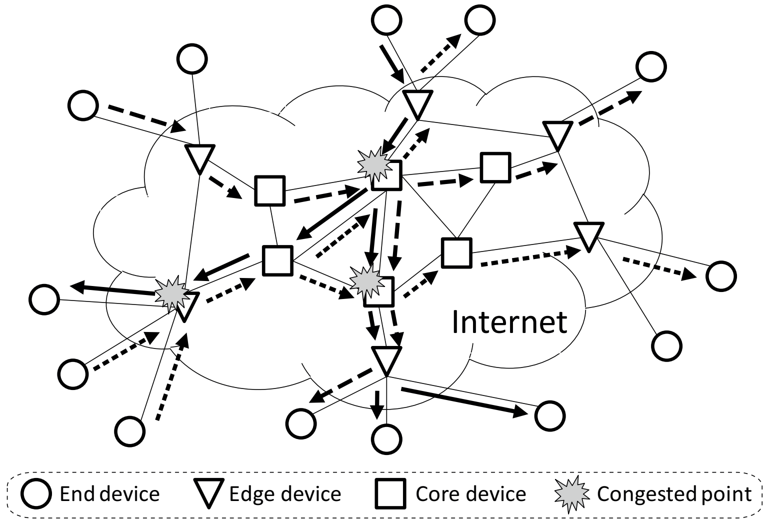


Figure 2.1: Example of network congestion in Internet

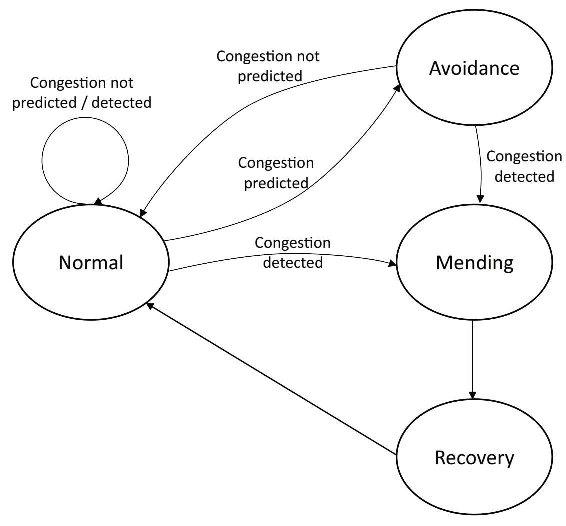
not end devices.

Any network that might experiences congestion will be operating in one of the three main stages; normal traffic flow stage and two congestion stages. When congestion is detected, the network will go through a congestion mending stage, followed by the recovery stage. Some networks will additionally go through congestion avoidance stage after optional congestion prediction. The naming of those stages has varied a lot throughout the literature. Fig. 2.2 (a) shows the network congestion state diagram of the operation of congestion vulnerable networks in the different congestion stages.

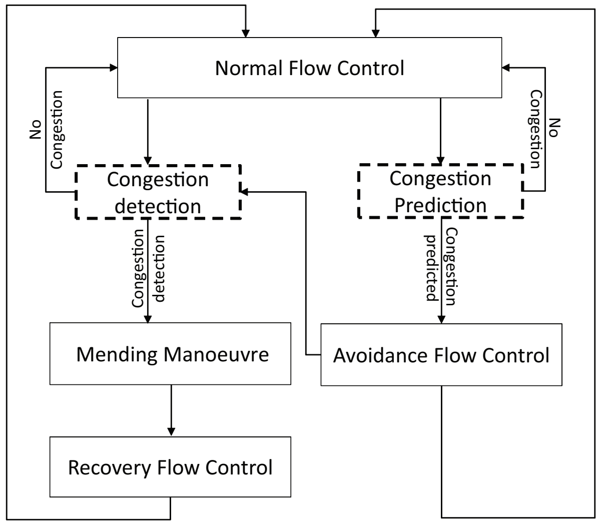
To handle the network operation during those stages, four different basic functions must be provided by the traffic flow management. Those four functions are; normal flow control, congestion detection, congestion mending manoeuvre and congestion recovery flow control. The correspondence between the stages and the functions is clear from the naming. Two additional optional functions are provided in the networks capable of running in optional avoidance stage. Those two functions are congestion prediction and avoidance flow control. Both congestion management and its functions have also been referred to in the literature with different names. The functions responsible for controlling the flow during the four network congestion stages are shown with solid boxes in Fig. 2.2 (b). Those functions are basic normal flow control, congestion mending manoeuvre and congestion recovery flow control, besides the optional avoidance flow control.

Both the congestion detection and congestion prediction functions, which is showed with dotted boxes in Fig. 2.2 (b), are responsible for switching the operation of the network from normal flow stage into either congestion mending stage or congestion avoidance stage, respectively. Switching between either stage is usually straight forward after carrying out the congestion mending manoeuvre or avoidance flow control, respectively. During the avoidance stage, congestion detection can also switch the network operation into the congestion mending stage.

Many techniques and standards have been introduced to resolve network congestion problems or avoid it. Some network standards were designed from the beginning to avoid any possibility of congestion occurrence, ATM and MPLS are among those network standards. In the next subsection, ATM as an example network that totally avoids any congestion is introduced. The following subsection will introduce TCP network as an



(a) Network congestion state diagram



(b) Network congestion control functions

Figure 2.2: Network congestion mechanism

example of network architectures experiencing congestion and how it is managed. The transmission control protocol (TCP) and Internet Protocol (IP) are actually two different network protocols. TCP and IP are normally implemented together. The terms TCP/IP or TCP are going to be used interchangeably throw out this thesis and considered to be equivalent referring to either the whole Internet or only the transport layer.

2.2 Congestion Free Networks

ATM is one of the technologies that are targeted to meet the Broadband Integrated Services Digital Network (BISDN) requirements. The BISDN is an extension of Integrated Service Digital Network (ISDN). In ISDN, the end-to-end digital connectivity for data such as audio, video, and data applications are provided. While in BISND, a wide range of applications that require higher transmission rate are supported. The BISND that uses ATM technique is also called as ATM networks. Almost all other networks mainly targeting voice connection services are also congestion free networks such as Public Switched Telephone Network (PSTN). Some data network as MPLS are also designed to provide congestion free traffic.

2.2.1 Asynchronous Transfer Mode (ATM)

According to the ATM reference model as shown in Fig. 2.3, it consists of three main layers, which are physical layer, ATM layer, and ATM adaption layer [5, 6]. The figure shows the ATM layers side by side with the corresponding OSI model layers. The physical layer is constructed by two sub-layers; physical medium (upper sub-layer) which provides bit transmission capability, and transmission convergence (lower sub-layer) which provides the functions of transmission frame generation and extraction, transmission frame adaption, cell rate decoupling, cell delineation, and Head Error Control (HEC) signal generation and confirmation. The ATM layer is mainly responsible for the cells traffic management and congestion control in the network. In ATM adaption layer, the segmentation is performed to split the higher layer information into a size of cells at the end devices of sender and reassembles back into data units at the end device of receiver devices before being delivered to higher layer.

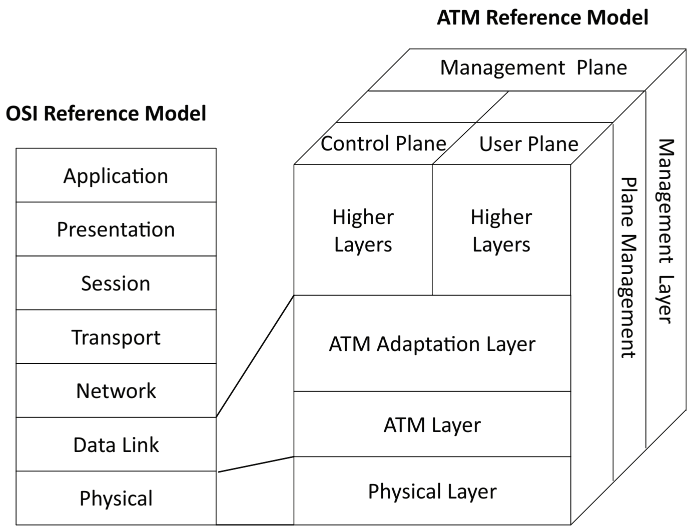
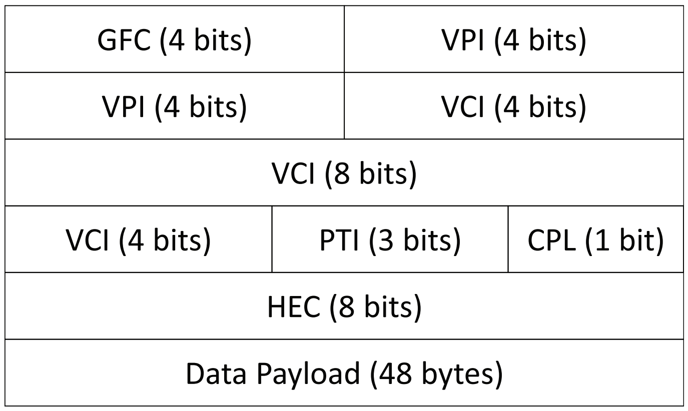


Figure 2.3: ATM Reference Model (courtesy of [7])

The information in the ATM network is transmitted in a short fixed length of 53 byte cells. The 5 bytes are used for ATM header and the remaining 48 bytes are the payload of encapsulation information. The ATM cell structure is shown in Fig. 2.4, where it consists of six header fields, Generic Flow Control (GFC), Virtual Path Identifier (VPI), Virtual Channel Identifier (VCI), Payload Type Identifier (PTI), Cell Loss Priority (CLP), and Header Error Control (HEC). The GFC field is used to control the traffic flow across the User Network Interface (UNI). The VPI defines the virtual paths between sender and receiver for a particular cell. The VCI is used to identify a channel path for a particular cell. By combining both information in the field of VPN and VCI, a virtual circuit for a specified ATM cell is identified. In an idle cell, all the bits of VPI and VCI are set to 0's. The PTI is responsible to identify the type of ATM cell that follows, to indicate whether the cell experienced congestion in its journey, and to determine the last cell in a block for ATM adaptive layer for user ATM cells. The CLP is used as priority indicator, and HEC is used for error detection.



GFC (4 bits)	VPI (4 bits)	
VPI (4 bits)	VCI (4 bits)	
VCI (8 bits)		
VCI (4 bits)	PTI (3 bits)	CPL (1 bit)
HEC (8 bits)		
Data Payload (48 bytes)		

Figure 2.4: The ATM cell structure

The ATM network provides several services to the users, according to the traffic type and the transmission method. The ATM service includes Constant Bit Rate (CBR), real time Variable Bit Rate (rt-VBR), non-real time Variable Bit Rate (nrt-VBR), Unspecified Bit Rate (UBR), and Available Bit Rate (ABR) [5, 8–10]. CBR provides the service of transmitting a constant bit rate of information. This service is mainly for the applications that need a fixed and continuous data during the connection lifetime and have a tight upper bound of transfer delay, such as Skype, YouTube, Live TV and so on. The rt-VBR service is for the real time applications that have limited time delay and variable data rate transmission. On the other hand, the nrt-VBR service is for the applications that consist of busty traffic characteristics and do not require tightly constrained delay variation. The UBR service is for tolerate variable delays and cell losses acceptable applications. During congestion, no feedback mechanism is concerning, the cells are lost and their source will not reduce the transmission rate to overcome the congestion. In ABR, some parameters in the used applications need to be specified, for instances Peak Cell Rate (PCR), Minimum Cell Rate (MCR), Cell Loss Rate (CLR), and so on. An explicit feedback message is used in the ABR to inform the status of congestion and/or the rate of to be transmitted data.

In ATM network, QoS is an important issue [8–10]. When a connection is established in ATM network, a contract about the given services that is related to various QoS parameters must be agreed by both user and the network. These parameters include Peak Cell Rate (PCR), Minimum Cell Rate (MCR), Cell Loss Rate (CLR), Cell Transfer Delay (CTD), Cell Delay Variation (CDV), and Cell Error Ratio (CER). The PCR is the maximum rate of cell that sender plans to send. The MCR is the minimum rate that a user can accept. CLR is the percentage of cells loss in the network. The cells are considered lost even though they did reach the destination, if they were received with an invalid header, or the content of cells has been corrupted by errors. The measured CTD can be defined as the time elapse between the departure time of a cell from sender end devices and arrival time at destination. The CTD includes propagation delays, internal delays (switching, processing, and internal transmission link), external queuing and transmission delays. The CDV indicates the uniformity of cells deliver, and CER is the fraction of cells that is corrupted during delivery.

Until here, the basic structure of ATM network has briefly explained. The following section will focus on the traffic management mechanism on ATM network.

2.2.1 Traffic Management in ATM network

The ATM network is designed to support variety of services and applications. The control of ATM mainly involves providing proper differentiated QoS for the network applications. As the traffic management of ATM, it is used to ensure the network and the end devices are out from congestion problems, so that high network performance is achieved. The traffic management is also used to promote the efficient network resources utilization. It ensures the efficient and fair operation in the networks, meanwhile fulfils the demand and QoS that users desired. In short, traffic management is a set of mechanisms that ensures the network resources are fully utilizes and meet the various QoS as part of traffic contract. The traffic management mechanisms for ATM networks are depicted in Fig. 2.5, which include connection admission control, usage parameter control, traffic shaping, selective cell discard, explicit forward congestion indication, and network resource management [8–10].

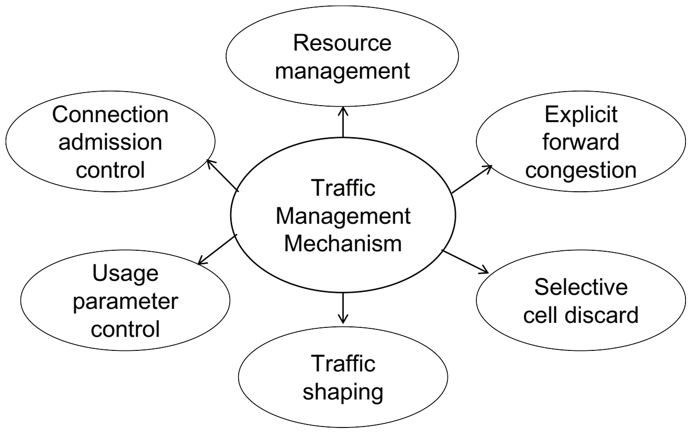

Resource management
Connection admission control
Explicit forward congestion
Traffic Management Mechanism
Usage parameter control
Selective cell discard
Traffic shaping

Figure 2.5: Type of traffic management mechanism

The connection admission control (CAD) is used to determine if the specific service required by the user can be accepted or should be rejected during the connection establishment. If a connection is accepted, the user and the network agree on a certain traffic pattern of that connection (traffic contract). As long as the user complies with the traffic contract, the network continues to deliver the cells in a timely fashion.

The Usage Parameter Control (UPC) is a mechanism that is used to monitor the traffic and enforce the traffic contract. The aim of UPC is to protect the network resource from mischievous or unintentional misbehaviour that would adversely affect the QoS of other established connections, by detecting violations of assigned parameter and taking appropriate actions such as cell discard or cell tagging.

The traffic shaping is attained to modify the characteristic of a connection into a desired pattern. The aim of this mechanism is to achieve higher network efficiency, at the same time meet the QoS objective. Accordingly, the traffic shaping can smooth out the flow of traffic and reduce cell clumping. The network resources can be fairly allocated and the average delay is also reduced in the traffic shaping mechanism. In other word, traffic shaping reduces congestion. One of the approaches to traffic shaping is the leaky bucket concept. The principle of leaky bucket is very simple. A bucket with a small hole in the bottom, water can enters the bucket at any rate; the outflow is at a constant rate if there is water in the bucket, else the outflow is zero. Still, once the bucket is full, water that enters the bucket is spilled over the sides. In this case, water is considered lost because it does not appear in the output stream under the hole. The same idea is applied to cells in the ATM network as shown in Fig. 2.6. Each end devices is connected to the network by an interface that contains a leaky bucket. The leaky bucket is the buffer which has a finite queue. When a cell arrives, if there is available queue, the cell is appended to the queue; otherwise, it is discarded. At every clock tick, one cell is transmitted.

The selective cell discard operates only when some point in the network behaves beyond the UPC function. The aim of this mechanism is to discard lower priority cells and protect the performance of higher priority cells.

The explicit forward congestion indicator is a mechanism that used a bit in the cell header, called EFCI. The EFCI bit is set to 1 when the congestion occurs in the network. The aim of this indicator is to notify the congestion status in the network, so that user can initiate the use of congestion control mechanism.

The ATM stack layer can be split, so that the lower layers can be utilized as one

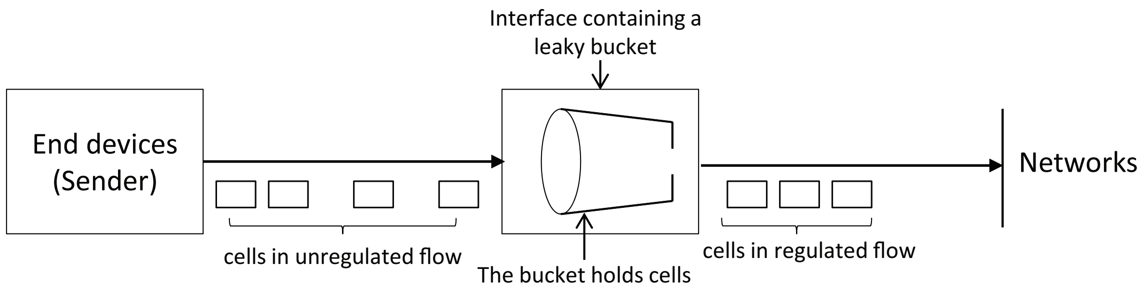


Figure 2.6: The leaky bucket concept in ATM network

of possible network interface layers of the TCP architecture stack as shown in Fig. 2.3. The collaboration of both stacks can be done in more than one configuration to service different goals. Accordingly, some of the setting does not manage to keep the congestion free property of the basic ATM stack. Those configurations would still have to delegate a congestion handling to the upper TCP/IP layers or introduce new mechanism to handle the possible congestion scenarios. Next sub section will start by introducing some of those mechanisms.

2.3 Congestion Vulnerable Networks

Most connection oriented networks that do not provide strict enough resource reservation per connection would surely suffer from congestion at some point of time. This sub section introduces an example of collaboration between congestion free networks and congestion vulnerable networks, followed by a pure congestion vulnerable networks example. The mechanism utilized by both network examples to handle congestion is also presented in this section.

2.3.1 Congestion in TCP on top of ATM networks

The main concern of traffic management is to ensure the users get their desired QoS, as well as the network resources are fully utilized. This concern is very difficult to achieve during the period of heavy load, especially the demands cannot be predicted in advance. This is the reason for congestion control schemes to take part in traffic management of network with TCP on top of ATM. The congestion control scheme plays the role of ensuring the connection has better throughput, low delay performance, and network resources are fairly allocated.

Various congestion control scheme have been proposed for TCP on top of ATM networks. Based on [11], the congestion control schemes can be classified into two main categories, open loop and close loop as shown in Fig. 2.7. Open loop congestion control scheme refers to the control decision that does not depend on any feedback information in the network. The controller is completely according to its own knowledge such as bandwidth capacity of the local links and the available buffers in the system. The open loop congestion control schemes usually consist of the admission handling mechanism and continuous activation features to help in stabilizing the traffic arrival rate. These schemes can be further classified into sender and receiver control. In the sender control, the transmission rate is control by the sender, whereas in the receiver control, the network traffic is control either at the receiver or intermediate nodes along the path. The congestion control schemes that are categorized under sender control are bit round fair queuing [12], schedule based approach [13], VirtualClock [14], input buffer limit [15], stop and go scheme [16]. Meanwhile, the congestion control schemes which are categorized under receiver control such as isarithmic method [17], packet discarding [18], and selective packet discarding [19]. In general, the open loop congestion control scheme is not powerful enough to handle the entire traffic pattern that might cause congestion in the network.

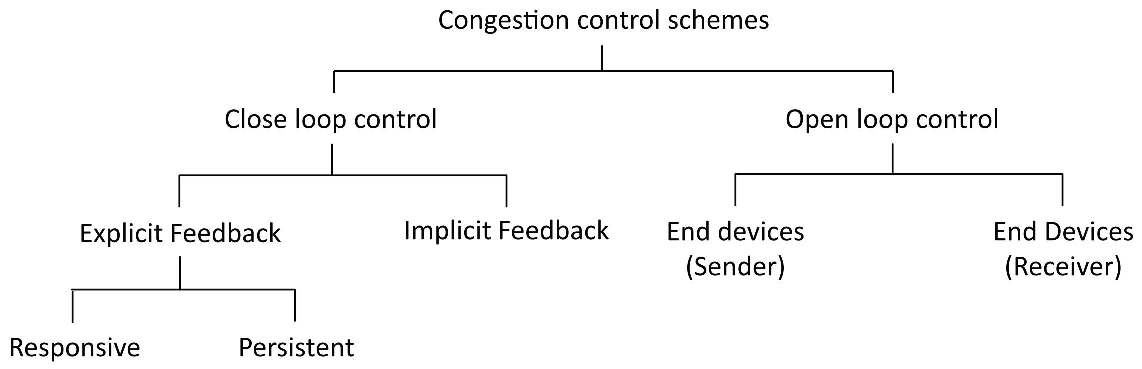


Figure 2.7: Taxonomy of congestion control schemes

In the closed loop congestion control scheme, their control decisions are based on the feedback information that might be from receiver and immediate neighbours. With this feedback information, the network performance can be monitored. Two types of feedback information are used in the close loop congestion control scheme, which are explicit feedback and implicit feedback. The explicit feedback involves a feedback that is sent explicitly as separate message. It can be further classified into persistent and responsive feedback. The feedback that is always available is called persistent, whereas the feedback that is only generated under specific condition is named as responsive. Examples of close loop based explicit feedback with persistent approach schemes are binary feedback scheme [20], selective binary feedback scheme [21], BBN scheme [22], adaptive admission control [23], Q-bit scheme [24], loss load curves [25], and hop by hop control [26]. For the existing of close loop based explicit feedback with responsive approach schemes are choke packet [27], rate-based control [28], dynamic time window [29], and source quench [18].

A scheme is considered as implicit feedback when there is no necessity to send feedback explicitly, such as time delays of acknowledgement. The existing congestion control schemes that is categorized under the closed loop based implicit feedback are slow start scheme [30], timeout-based scheme [31], Tri-S scheme [32], and warp control [33].

2.3.2 TCP/IP network

TCP/IP provides end-to-end data transfer that requires segmentation, addressing and routing among a lot more functions. Those functions are organized into the well-known abstraction layers of the TCP/IP architecture model as in Fig. 2.8, which is similar to the OSI counterpart [34].

The application layer is a place where the network applications and their applications protocol is located. The application protocol includes Hypertext Transfer Protocol (HTTP), Simple Mail Transfer Protocol (SMTP), Domain Name System (DNS), and so

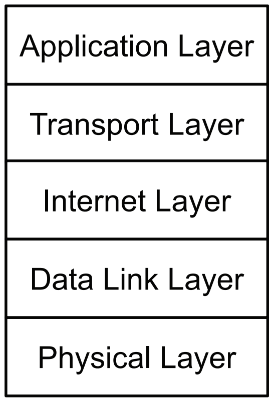


Figure 2.8: TCP/IP reference model

on. The application layer protocol is distributed over multiple end devices. The application is implemented by different protocols to exchange information in the form of messages between end devices.

The transport layer transmits the messages from the application layer from one end device to another. The main two transport protocols in the Internet are Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). TCP provides a reliable connection-oriented service to applications. TCP also have to handle flow control including congestion control for the connection. On the other hand, UDP provides a connectionless service to applications, thus flow control is inapplicable. UDP does not readily provide reliability or connection, still most applications make used of UDP header fields to implement some kind of reliability or connection.

The network layer takes the responsibility of moving the information in the form of datagrams from one host to another. The network layer in Internet is also known as the IP layer. There is only one IP protocol, and the entire Internet devices with network layer must be able to run the IP protocol beside other supplementary network layer protocols mainly for control.

The network interface layer is in charge of moving data from one node to the next node in route as long as there is a direct point to point physical connection. To perform this function, the link layer needs a specific link layer protocol that is used over the link. The link layer protocols commonly used through the Internet includes the Ethernet standard family as well as wireless and mobile standards.

The physical layer is responsible for moving the bits within the frame from one node to the next. The protocol used in this layer depends on the actual transmission medium such as single-mode fibre optic, twisted pair copper wire, coaxial cable and so on.

Among these layers, the existing congestion control techniques are mainly located in the transport layer of TCP with some supporting techniques in other layers. Some of the congestion control techniques depend on the collaboration between more than one layer, for example the cross-layer control techniques [35]. Surely transport layer is always one of the collaborating layer to manage congestion. It has been generally accepted that any congestion control in both TCP and network layers can only solve a congestion situation after occurring by decreasing the transmission rate of the end device sender. Still, by the time the rate starts to decrease, a number of packets may have been dropped or have been retransmitted without any real need.

2.3.2.1 Types of Congestion Control Techniques

This section will focus only on the techniques implemented in both transport and network layers. Those techniques are classified as end-to-end and hop-by-hop congestion control techniques, respectively shown in Fig. 2.9. End-to-end congestion control techniques are particularly associated with transmission control protocol (TCP) [36]. In end-to-end congestion control, the sender (sometimes the receiver) receives an acknowledgement from the receiver or a network signal periodically. The sender continuously adjusts its transmission rate based on the acknowledgements received. When acknowledgements are over waited (timed out) the sender would assume congestion and start transmission rate reduction with the retransmission. End-to-end congestion control will be thoroughly discussed in sub-section 2.3.2.1.2.

Hop-by-hop congestion detection initiates a signal of congestion from the affected intermediate nodes to travel towards the receiver and then back to the sender. If the sender receives this signal without the packet being dropped due to congestion, then the sender will start transmission rate reduction. The hop-by-hop congestion control is used in the network layer based on the intermediate node buffer space availability to avoid buffer overflow [37–39]. Hop-to-hop congestion control for the transport layer requires per-flow state information in intermediate nodes, which limits its scalability. Thus, hop-to-hop congestion control almost restricted to helping end-to-end congestion control techniques by providing detection or prediction and such functions. The congestion control techniques that were presented in [35,38] have combined hop-by-hop and end-to-end control. Hop-by-hop is also quite popular in wireless networks specially those having rudimentary transport layers. Example of hop-by-hop congestion control techniques are given in the next sub-section.

Hop-by-hop has been restricted in use to only complement the end-to-end congestion control to prevent congestion. In some cases, it can help detect the congestion earlier and provide effective feedback to the end device to respond. Hop-by-hop congestion control techniques are not widely used because of its additional memory, long processing time and higher complexity. According to the conventional wisdom, the core network should be flexible and simple, thus most of the congestion control is implemented at the end device. However, the current advanced and powerful hardware technology allures researchers to start considering more complicated tasks in the intermediate the edge/core device for limited size networks.

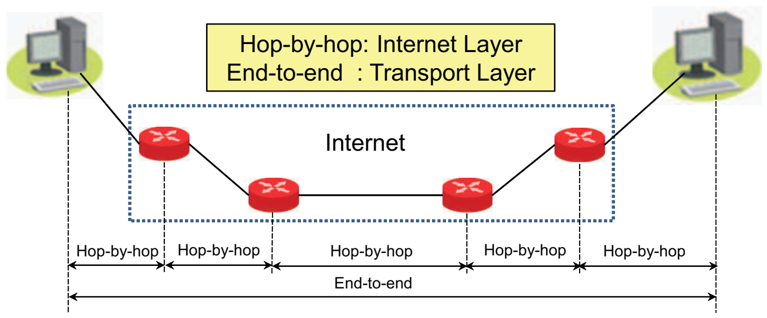


Figure 2.9: The hop-by-hop and end-to-end transport control

2.3.2.1.1 Hop-by-hop Congestion Control

Hop-by-hop network control with Active Queue Management (AQM) is one of the well-known complementary approach in helping the transport congestion control to avoid intermediate node buffer overflow. Fig. 2.10 shows the classification of AQM mechanisms. Hop-by-hop congestion control is mostly implemented in the wireless networks. In wireless system, congestion that happens in the transmission medium is referred to as link level congestion. If the congestion occurs in queue (buffer), it is called node level congestion. To solve congestion, both MAC layer functions and transmission rate reduction at upper layers are utilized. Generally, flow control in wireless networks performs three basic functions: traffic control, resource control and reliable data transport. Traffic control is basically an end-to-end control function. Resource control attempts to utilize alternative path to the sinks when the main path is congested. Table 2.1 shows an example of the congestion control techniques in wireless sensor networks (WSN). The H-by-H and E-to-E in the Table 2.1 refer to hop-by-bop and end-to-end respectively.

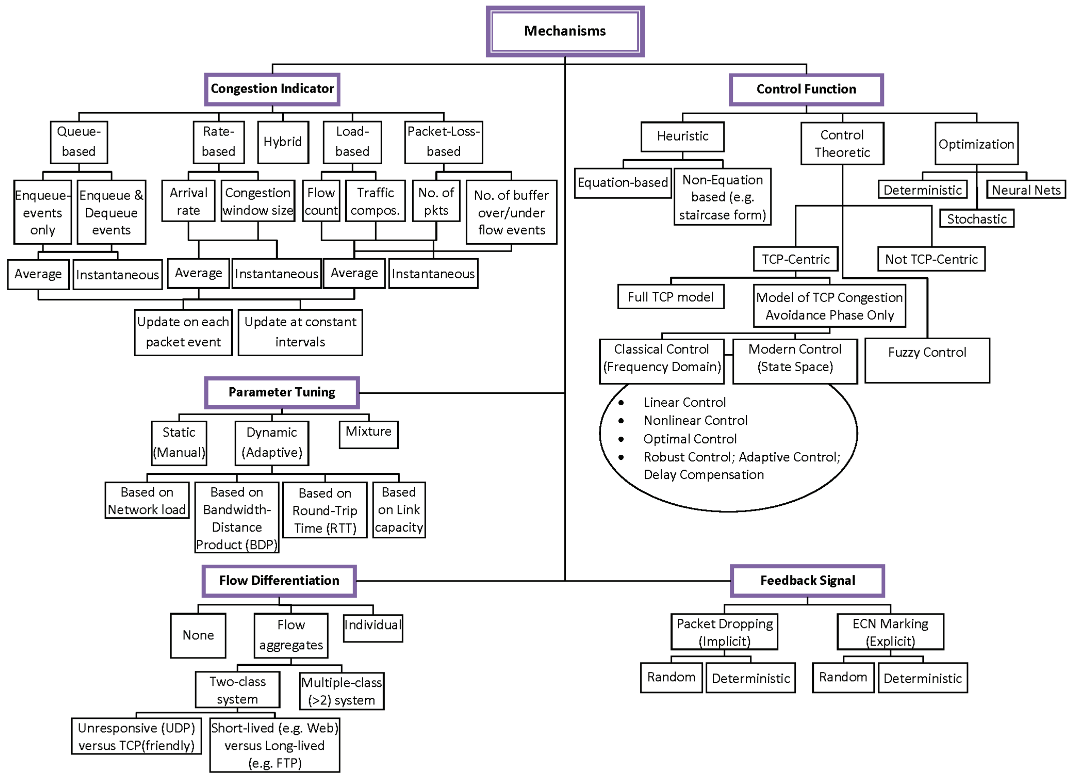


Figure 2.10: The classification of AQM mechanism (courtesy of [40])

Table 2.1: Example of congestion control mechanisms in WSN

Protocol	Technique	H-by-H/ E-to-E	Description
CODA [41]	Traffic Control	Both	Two main techniques: 1) open loop hop-by-hop backpressure: A node broadcasts backpressure mechanism when congestion detected. The node that receives a backpressure acknowledgement will adjust its sending rate by AIMD approach. The upstream node (towards the source) receives the backpressure acknowledgement will decide to further propagate the backpressure upstream based on the local network conditions. 2) closed loop multisource regulation: Operates on a slower time scale and control the congestion on multiple sources from a single sink in an event of persistent congestion [42, 43].
SenTCP [44]		H-by-H	A transport protocol that uses open loop hop-by-hop congestion control. It detects congestion using local congestion degree and uses hop-by-hop for control.
HTAP [45]	Resource Control	H-by-H	A protocol that based on the creation of alternative paths from the source to sink. When the candidate congested receiver sends a backpressure packet to the sender, the sender stops the transmission of packets to the candidate congested receiver and searches in its neighbour table to find the least congested receiver in order to continue the transmission of data.
ESRT [46]	Reliable Data Transport	E-to-E	A protocol that based on two parameters: Event reliability and reporting frequency. Event reliability is the number of data packets received at the decision interval at the sink. The end-to-end data delivery services are regulated by adjusting the sensor report frequency. If the reporting frequency is too low, the sink will not be able to collect enough information to detect the events. If the reporting frequency is too high, it endangers the event transport reliability. ESRT adjusts the reporting frequency such that the observed event reliability is higher than the desired value to avoid congestion.

2.3.2.1.2 End-to-End Congestion Control

The end-to-end transport is the classical approach to the TCP congestion control techniques. Fig. 2.11 shows the example of end-to-end congestion control transport. A plethora of research has been proposed in relation to the TCP congestion control techniques, the first congestion control variant was introduced by Van Jacobson in 1988 which used the end-to-end mechanisms [47]. In his design, there are four congestion control stages: slow start, congestion avoidance, fast retransmit, and fast recovery. Together with these four stages, they form the basic TCP flow and congestion control [47, 48]. Later, other TCP variants were designed to increase the network throughput not just for congestion but also for other network conditions. These TCP variants include features like early congestion detection, available bandwidth detection, and loss type estimation. With these features, a sender can detect congestion state, buffer utilization, loss events and route condition changes to ensure the network resources are fully utilized.

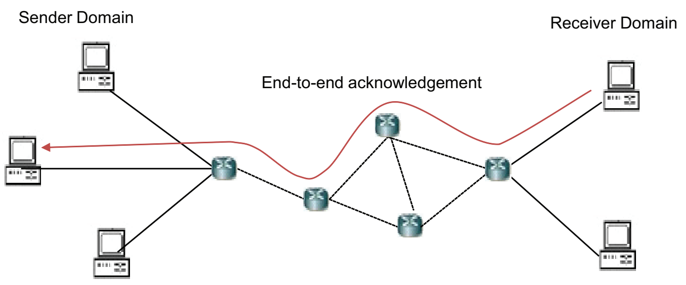


Figure 2.11: Example of end-to-end congestion control transport

Recent research includes TCP/Network Coding [49], history-based TCP throughput prediction [50], and neural networks model in TCP throughput estimation [51]. Such research investigates communication networks, such as fifth generation mobile networks, machine-to-machine (M2M) networks, and Internet of Things (IoT) networks, network coding, prediction, machine learning, neural networks, optimization theory and the others.

The vast development of TCP variants lead to many surveys that have been conducted by researchers. Those surveys classified the variants performance from 3 different perspectives; generally, in certain network environments, and with specific parameters. The first perspective, TCP general performance compare between the different TCP variants. Westwood, NewReno, and Vegas were surveyed by L. A. Grieco et al. [52]. Similarly, H. Jamal et al. [53] compared a standard TCP Reno to various variants and also classified the variants into loss-based, delay-based, and mixed loss-delay based. The second perspective, many surveys studied how different TCP variants perform in different network environments. A. A. Hanbali et al. [54] have discussed wireless issues and the major factors involved in TCP congestion control over mobile ad hoc networks (MANET). Similarly, H. Balakrishnan et al. [55] compared protocol categories of end-to-end, link-layer, and split-connection of TCP congestion control over the wireless networks. K. S. Reddy et al. [56] focused the congestion control in high-speed network, by considering TCP variants such as BIC, CUBIC, FAST TCP, HSTCP, Layered TCP, STCP, and XCP. The last perspective, A. Afanasyev et al. [57] collected and described a comprehensive set of TCP variants and mechanisms that optimize various parameters in different network environments. The survey also explained each TCP variant, which includes its strengths and weaknesses.

In this sub section, existing TCP variants are described according to its controlling device entity and characteristics, followed by their association. This can give a quick view of the types of TCP variants and their relationship. Association of the variants is important here because it allows the researches to understand the evolution and the importance of congestion control in current and future networks. In addition, TCP congestion detection is discussed in terms of its evolutionary chain. The variants are also compared according to the device of detection. Two parameters: duplicated acknowledgement (ACK) and round trip time (RTT) are two of main factors effecting the TCP variants mechanism. Packet loss for duplicated ACK, and delay for RTT are used as indicator to detect the congestion in the network. Lastly, we consider TCP congestion avoidance (CA) techniques in terms of dependency on congestion window size (cwnd). This allows us to know how the TCP variants work in controlling the transmission rate to avoid congestion.

2.3.2.1.2.1 TCP Variants List

The 34 most commonly used TCP variants are listed here. Throughout the following sub section, the listed TCP variants are classified and compared.

1. **Tahoe [47]**:
   Tahoe is one of the earliest end-to-end congestion control algorithms. It introduces a slow start algorithm, a congestion avoidance algorithm, and a fast retransmit algorithm. The slow start algorithm and the congestion avoidance algorithm allow a TCP sender to detect available network resources and adjust the transmission rate according to the detected limits. Whereas the fast retransmit algorithm can detect the losses in the networks and allows the sender to retransmit the lost data without waiting for the corresponding retransmit time out event. Tahoe provides a great concept in solving the congestion collapse problem, but it has been obsoleted due to high-amplitude periodic phase.

2. **Reno [58]** :
   Reno has improved the Tahoe by adding a fast recovery algorithm into the mechanism. The fast recovery algorithm is to halve the congestion window and hold it from increases until the networks is recovered. Reno is the congestion control standard for TCP due to its simplicity and characteristics. However, the packet loss detection can only performs at once a time.

3. **Vegas [59]**:
   Vegas is a proactive approach that uses the round trip time (RTT) value to determine the networks congestion status. If the RTT is increased, the Vegas TCP will assume the networks are congested and reduce the transmission rate; if the RTT is smaller than the minimum RTT, the congestion window will increase to raise the transmission rate.

4. **SACK [60]**:
   Although NewReno solved the problem of multiple packet losses in the fast recovery stage of Reno, it does not solve the problem of prolonged the fast recovery stages. In fast recovery stage, the sender only transmits a single packet upon error detection. Therefore, SACK (Selective ACKnowledgement) has been presented to allow the receiver to inform the sender about the block of data packet that is successfully

transmitted. As a result, the sender can easily determine lost packet block and retransmit them immediately. Unfortunately, SACK cannot work well in multiple packet loss problems.

5. **TD-FR [61]**:
   TD-FR is developed to mitigate the out-of-order packet delivery problem. The causes of out-of-packet delivery problem are such as erroneous of software or hardware behaviour such as malfunctions or misconfiguration, enforcement of diverse packets handling services in a router, etc. In TD-FR, a receiver will holds the out-of-order data packets respond for few milliseconds depending on reordering pattern. This delay is to prevent the sender from over-react and to identify the actual losses. However, if the delay is too long, the "fast" loss detection mechanism will become slower than the standard loss detection based on retransmission time out (RTO).

6. **NewReno [62]**:
   NewReno enhances the Reno by improving its response when multiple packet losses occur in the single congestion event. It confirms the fast recovery stage only end after all lost packets are retransmitted. Conversely, a feedback message is in the form of cumulative acknowledgement, which consists of limited available information.

7. **DSACK [63]**:
   DSACK provides information of each receipt of a duplicate packet to the sender. There are two types of duplication: a part of acknowledged continuous data stream, and a part of isolated block. For the acknowledgement continuous data stream, a DSACK includes a duplicate range of sequence numbers in the first block of SACK option. For the isolated block, the receiver attaches the isolated block at the second position in SACK option. As a result, DSACK provides a way to report packet duplication. The drawback of DSACK is the receiver can send faulty information causing the sender to make a wrong decision to increase or reduce the congestion window size.

8. **Westwood [64]**:
   Westwood modifies the NewReno congestion control algorithms to suit the wireless environment that consists of non-congestion related losses. Westwood estimates the available bandwidth through the ACK arrival rate and improve the fast recovery stage by an optimal value of congestion window. The weakness of Westwood is the bandwidth estimation technique provides inaccurate results in certain network conditions: overestimating network bandwidth during congestion and underestimating network bandwidth in the presence of random errors.

9. **Veno [65]**:
   Veno improves the throughput utilization of Reno algorithm by using the Vegas bottleneck buffer estimation technique for early detection of congestion status. However, Veno algorithms tend to stay longer in the congestion avoidance state with larger congestion window value.

10. **TCPW CRB [66]** :
    TCPW CRB (Westwood with Combined Rate and Bandwidth estimation) is to

resolve the problem of high inaccurate of bandwidth estimation in Westwood algorithm at certain network conditions. TCPW CRB has included the rate estimation for long term bandwidth estimation in estimation algorithm to prevent the overestimation of a congested network. However, TCPW CRB is unable to differentiate whether the data packet losses are due to random error or buffer overload.

11. **Nice [67]** :
Nice is proposed to minimize the interference between high-priority and low-priority flows. Nice has considered all TCP flows that carry high priority data and attempts to use the network resources if nobody else uses them. Nice defines a queuing delay threshold to compare with the arrival of each non-duplicate ACK packet. Nice counts the number of times that the queuing delay exceeds the threshold values for each RTT period. The counted value estimates the number of ACK packets which have been delayed due to interference between changing traffic of high priority data packet flows and low priority data transfer. If the estimate delay exceeds a predefined threshold, the congestion window size will be reduced to halve.

12. **LP [68]** :
LP is developed for a low priority data service for background applications. LP uses the Timestamp option and heuristic approach in estimating the one-way propagation delay to calculate queue delay for each RTT period. LP provides early congestion detection by employing a simple delay threshold based method. LP retains the minimum and maximum of one-way delays during the connected lifetime. The current one-way delay estimate is compared with a predefined threshold which is a fraction of queuing delay plus a minimum of one-way delay. Once the early congestion detection event is triggered, LP reduces the congestion window (cwnd) size to half of the current value and start the inference timer. If another early congestion detection event is triggered within the timer elapse, LP is assumed the presence of high-priority flow and reduced the congestion window size to minimal value. Otherwise, LP resumes the Reno type of congestion avoidance algorithm.

13. **TCP-Real [69]** :
TCP-Real implements a receiver oriented approach that can reduce an unnecessary transmission gaps and designates recovery strategy responsive to the packet losses. The multiplicative decrease in congestion avoidance state causes the transmission gaps to degrade goodput and experience jitter in real time applications. TCP-Real introduces another parameter γ to determine the window adjustments during congestion avoidances to balance the trade of additive increase and multiplicative decrease parameters. Moreover, TCP-Real receiver uses a pattern called "wave" to observe the level of contention and/or packet loss. This approach is to mitigate the congestion control algorithms from over-react to the packet losses that due to short-lived flows or wireless interferences. This wave approach consists of few fixed size segments which are sent back-to-back similar to the way TCP handles packets. The wave size is used to observe and estimate the gap of the missing packet. These observations use an ad interim method to identify transient random error from congestion. The error is verified at the next RTT by comparing the perceived rate and the previous rate.

14. **DOOR [70]** :
DOOR is developed to resolve the congestion control from overact to the route changes in wireless networks especially mobile ad-hoc network. During route changes, some data packets are dropped, the congestion control algorithms will assume the packet loss due to congested network and reduce the rate flow instead of re-transmitting the loss packets. Therefore, DOOR has a feature of out-of-order detection and feedback to sender for temporary disable the congestion control actions during the route changes. In addition, DOOR can revert to the original state of the congestion window and retransmission timeout values, if the congestion control has recently reduced the sending rate due to loss detection. This action is called instant recovery and able to alleviate the drawbacks that cause by previous detected rerouting event. However, the transmission rate in DOOR may become inappropriately after route change and performs poorly in substantial persistent packet reordering.

15. **TCPW BR [71]** :
TCPW BR (Westwood with Bulk Repeat) adds a special loss type detection mechanism in Westwood algorithm. It uses the queuing delay estimation threshold and rate gap threshold algorithms to increase the estimation precision and reduce the number of false. Moreover, TCPW BR modifies the retransmission timer back off algorithm by restricting the maximum timer with a predefined constant during non-congestion related losses. TCPW BR handles the multiple packet losses in the same way as NewReno, therefore TCPW BR has the same drawback of NewReno, where the feedback message is in the form of cumulative ACKs.

16. **STCP [72]** :
STCP replaces the additive increase and multiplicative decrease (AIMD) of controlling methods to a multiplicative increase and multiplicative decrease (MIMD) methods. During congestion avoidance state, the congestion window size increases by a fraction of window size with each RTT. During fast recovery state, the congestion window size decreases by a different fraction β upon detecting a loss. The drawback of MIMD policy is an STCP flow becomes extremely unfair.

17. **HS-TCP [73]** :
HS-TCP is an alternative to STCP in solving the effectiveness of data transfer problem in high speed / long delay networks. The increase coefficient α in congestion avoidance and decrease factors β in fast recovery are acted as a function of congestion window size to avoid the used of unrealistic low loss rate in high Bandwidth Delay Product (BDP) network. In addition, HS-TCP included a complementary algorithm that bounds the maximum increase step for the slow start state. HS-TCP has a fairness problem if the flows consist with different RTTs.

18. **FAST TCP [74]** :
FAST uses a constant-rate cwnd equation based update. The new cwnd size is calculated by the equation, $w = (w.RTT_{min})/RTT + \alpha$; where w is the current window size, RTT and RTT_{min} are current and minimum RTT, and α is protocol parameter. The selection of α has effect on the scalability and stability of TCP. This is still an open issue, even though the authors have concluded that α should be a constant value. The drawbacks of FAST are it depends highly on the minimal RTT value and the congestion window update rule is not friendly to standard TCP.

19. **TFRC [75]** :
    TFRC is developed for applications that have a constant packet size with varying transmission rate in response to congestion. TFRC sender and receiver are collaborated in handling the congestion control mechanism. The receiver measures the loss event rate and feedback this information to the sender, whereas the sender use this feedback messages to measure the RTT. Later, the loss event rate and RTT values are fed into the TFRC's throughput equation to determine the acceptable transmit rate in bytes per second. The sender will adjust its transmitting rate according to the calculated rate. The data packets lost is detected when the arrival of at least three packets with a higher sequence number than the lost packet in the receiver. Once the loss event is determined, the sender will reduce the cwnd size into halve during any single RTT. Conversely, TFRC has a strict response to idle or data limited periods.

20. **BIC [76]** :
    BIC extends the NewReno algorithm by including a Rapid Convergence phase in its mechanism. This phase implements the binary search manner to optimize cwnd size through the packet loss detection to indicate the cwnd overshooting. Moreover, BIC not only adopts the HS-TCP's limited slow start phase and also bound the increase in rapid convergence if the search range is more than some predefined value. In a network with large multiplexing, the RTT fairness and inter-fairness values in BIC is low.

21. **Hybla [77]** :
    Hybla is proposed to resolve the RTT-unfairness problem in high speed/long delay networks. Hybla modifies the NewReno's slow start and congestion avoidance phases to partially independent with RTT. The cwnd size is defined as $w = w + 2^{\rho} - 1$ (slow start), and $w = w + \rho^2/w$ (congestion avoidance), where the scaling factor ρ is calculated through the equation $\rho = RTT/RTT_{ref}$. Furthermore, Hybla also includes the techniques of data pacing and initial slow start threshold estimation using packet pair algorithm. The data pacing technique is to smooth the burst-nature of TCP transmissions, and packet pair algorithm is to estimate the network path capacity. However, if the initial RTT is determined wrongly in Hybla, it could risk the fairness of capacity sharing and perform poorly in high speed networks with relatively small delay.

22. **Vegas A [78]** :
    The Vegas A is the extended of Vegas with adaptive mechanism. The threshold coefficients from Vegas algorithm are adjusted according to the actual transmission rate. In addition, Vegas A improve the re-routing and fairness features of Vegas algorithm. The drawback of Vegas is it cannot identify that long RTT is experienced by the congestion or the route had changed. Whereas, Vegas A achieves low throughput performance compared to NewReno.

23. **Casablanca [79]** :
    Casablanca implements the differentiated service (Diffserv) technique into NewReno to distinguish congestion losses from wireless losses and react to the losses problems appropriately. The receiver in Casablanca TCP performs the discriminator func-

tion to identify the congestion state: “likely congestion”, “congestion”, or “corruption”. The initial state after a connection establish is “likely congestion”. If a loss is detected at the receiver, the current state becomes “congestion” if the function of discrimination is less than one; otherwise the current state becomes “corruption” which representing as wireless error loses. If the current state is “corruption”, the receiver will transmit an Explicit Loss Notification on the duplication acknowledgement to inform the sender. However, Casablance requires the related routers to have a differential packet dropping policy.

24. **NewVegas [80]** :
New Vegas enhances the Vegas algorithm by defining a new phase called Rapid Window Convergence. When the estimated network buffering exceeds the threshold in the slow start phase, the rapid window convergence phase allows the exponential-like resource probing to continue with reduced intensity. In addition, New Vegas implements the packet pacing techniques to set up a minimal delay between transmissions of any two consecutive packets to resolve the problem of traffic initialization bursts. The data packets in New Vegas also transmit in pair to solve the problem of estimate bias. The disadvantage of New Vegas is the rapid window convergence phase only improves the early termination of slow start.

25. **Africa [81]**:
Africa combines the aggressiveness of HS-TCP when the network is not congested and the conservative characteristic of NewReno when the network is congested. Africa also implements the Vegas algorithm in determining the congestion status by comparing the estimate of network buffering to a predefined constant. If the estimate of network buffering is less than the predefined constant, Africa move to fast mode and applies HS-TCP rules of congestion avoidance and fast recovery phase. Otherwise, Africa move to slow mode and applies Reno rules.

26. **CTCP [82]** :
CTCP uses a delay based scheme to estimate the network congestion status to associate Reno congestion control with a congestion control that is scalable in high BDP networks. CTCP introduces a scalable component w_{fast} in the congestion window size calculation ($w = w_{reno} + w_{fast}$). If the estimated value is less than the predefined constant, the component w_{fast} is updated according to the modified HS-TCP rules. If the Vegas estimate value exceeds the predefined constant, the component w_{fast} will gradually be reduced by a value proportional to the estimate itself. This reduction is to smoothen the transition between HS-TCP and Reno. The drawback of CTCP is that it is very sensitive to correctness of RTT measurements. For instance, if there is a flow that has higher RTT among the same network with different minimal RTT competing to each other, CTCP will react more aggressively and unfair to the other flow.

27. **MCP [83]** :
MCP uses the same congestion control mechanism as TCP, such as slow start, congestion avoidance, fast retransmit, and fast recovery. However, MCP location control is based on mobile host for all case. The mobile host can be a sender or receiver. MCP limits the transmission rate by sending (sender centric) or requesting

(receiver-centric) within the buffer capacity of both the sender and the receiver. Unlike TCP, MCP does not treat every packet as network congestion. MCP uses a cross-layer scheme to distinguish the type of losses and respond according to the losses. If the packets lost is due to congested network, the sender will decrease the transmission rate and retransmit the lost packets. If the packets lost is due to the non-congestion related losses, the sender will retransmit the lost packets without decreasing the transmission rate. The disadvantage of MCP is it performs poorly in the multi-hop networks.

28. **Illinois [84]** :
Illinois is proposed to resolve the performance degradation of delay based algorithms when the RTT measurements are noisy. Illinois defines the congestion window, w increase steps, α in congestion avoidance and decrease ratio, β in fast recovery with the queuing delay functions. The increased coefficient, α is inversely proportional to the queuing delay, whereas the decreased coefficient β is directly proportional to the queuing delay. The α and β coefficients are updated once for every RTT. In order to reduce the effects of queuing delay measurement noise, the α coefficient can be set to maximum when the numerous consecutive queuing delay values is less than the default value of first threshold.

29. **Fusion [85]**:
Fusion combines the concepts of Westwood's achievable rate, DUAL's queuing delay and Vegas' network buffering estimations. Fusion defines three separate linear functions which are switchable depending on the absolute queuing delay threshold. If the present queuing delay is less than the predefined threshold value, the congestion window is increased at a fast rate at each RTT through the Westwood's achievable rate estimate. If the queuing delay is three times more than the predefined threshold value, the congestion window is reducing to the lower bound. If the queuing delay is relying between the predefined threshold value and three times of predefined threshold value, the congestion window will remain the same.

30. **CUBIC [86]** :
CUBIC enhances the BIC algorithm with RTT-independence congestion growth functions to preserves RTT-fairness and inter-fairness properties of BIC's limited slow start and rapid convergence phase. The function is fast growth when the current window is far from the estimated target window, and become conservative when the current window is close to the estimated target window. CUBIC is the second most used congestion control for TCP due to the good performance and fairness properties. However, CUBIC suffers from slow convergence causing the poor network responsiveness.

31. **Libra [87]** :
Libra is one of the TCP variants to resolve the scalability issues in high speed/long delay networks. Libra modifies the NewReno's increase steps of congestion window in congestion avoidance phase to a function of the RTT and bottleneck link capacity. The packet pair technique is used in Libra to estimate the capacity of bottleneck link. However, Libra is too reliance on queuing delays estimation such as minimum RTT, maximum RTT, and RTT measurement consistency. If the estimation biases occur, the TCP performance will become worse.

32. **TCP-FIT [88]** :
    TCP-FIT is used for the heterogeneous networks that contain high speed/long delay links and wireless links. It implies the parallel TCP connections without modifying any layers in the protocol stack and/or applications software. The AIMD mechanism is used to adjust the congestion control window.

33. **ICTCP [89]** :
    ICTCP performs incast congestion avoidance at receiver. The receiver can adjust the received window size of TCP connections to control the bursts from all the synchronized senders. TCP receiver window is fine tuned according to the ratio of achieved and expected connection throughput over the expected throughput, and the hop of available bandwidth before the receiver.

34. **FWestwood [90]** :
    FWestwood includes the fuzzy controller in the Westwood congestion control algorithm to improve the performance of wired networks with high error rate. The number of timeout events, the number of triple duplicate acknowledgement, the time difference between the last two timeout events, and round trip time are used as the parameters to determine the congestion status.

2.3.2.1.2.2 TCP Variants Classifications

Here, TCP variants is discussed according to three different perspectives: *device entity*, *characteristic* and *association.*

2.3.2.1.2.2.1 Device Entity

TCP variants can be classified into four types according to the controlling device entity, i.e., *sender*, *receiver*, *sender or receiver*, and *sender and receiver.* The classification of the studied TCP variants among the four types is shown in the Table 2.2 and the top part of the Table 2.3.

Sender category is sometimes referred to as sender-centric protocol (SCP). In SCP, the sender performs essential tasks such as reliable data transfer; whereas the receiver only needs to transmit feedback packets in the form of acknowledgement to the sender [91]. The data transfer between the sender and the receiver in SCP is also referred as data-acknowledgement message exchange. Upon receiving this feedback information, the sender tunes the Congestion Window (*cwnd*) based on a window based mechanism to ensure the number of transmission bytes does not exceed the network capacity.

The idea of having the congestion and flow controlled at the receiver side was introduced in 1997 [91–93]. This is also called receiver-centric control protocol (RCP). RCP uses the same window based mechanism similar to the SCP, but the data acknowledgement message exchange is no longer applicable. However, the RCP uses the request-data message exchanges for data transfer, in which a receiver sends an explicit request packet to the sender for requesting the data packets to be sent. Through this way, the sender only can transmit its data packets according to the transmission rate that is requested by the receiver. The receiver uses the incoming data packet as an acknowledgement to its previous request for data. According to [94], the TCP performance can be significantly improved by using the RCP approach.

Table 2.2: A comparison of TCP variants in the aspect of the device entity

TCP Variant	Initiator(s)	Year of Proposal	Device of Control	Improvement Aspect	Nomenclature
Tahoe [47]	Van Jacobson	1988	S	LL	
Reno [58]	Van Jacobson	1990	S	LL	
Vegas [59]	Lawrence Brakmo, et al.	1995	S	LL	
SACK [60]	Matt Mathis, et al.	1996	S	LL	Selective ACKnowledgement
TD-FR [61]	Vern Paxson	1997	R	PR	Time Delayed Fast Recovery
NewReno [62]	Sally Floyd, et al.	1999	S	LL	
DSACK [63]	Sally Floyd, et al.	2000	R	LL	Duplicate SACK
Westwood [64]	Saverio Mascolo, et al.	2001	S	LL	
Veno [65]	Fu Cheng Peng, et al.	2002	S	LL	VEgas and reNO
TCPW CRB [66]	Ren Wang, et al.	2002	S	LL	Westwood with Combined Rate and Bandwidth estimation
Nice [67]	Arun Venkatarammani, et al.	2002	S	LP	
LP [68]	Aleksandar Kuzmanovic, et al.	2002	S	LP	Low Priority
TCP-Real [69]	Vassilis Tsaoussidis, et al.	2002	R	LL	
DOOR [70]	Feng Wang, et al.	2002	S and R	PR	Detection of Out-of-order and Response
TCPW BR [71]	Guang Yang, et al.	2003	S	LL	Westwood with Bulk Repeat
STCP [72]	Tom Kelly	2003	S	LD	Scalable TCP
HS-TCP [73]	Sally Floyd	2003	S	LD	High-Speed TCP
FAST TCP [74]	Cheng Jin, et al.	2003	S	LD	
TFRC [75]	Mark Handley, et al.	2003	S and R	PR	TCP Friendly Rate Control
BIC [76]	Lisong Xu, et al.	2004	S	LD	Binary Increase Congestion control
Hybla [77]	Carlo Caini, et al.	2004	S	LD	
Vegas A [78]	Srijith Krishnan Nair, et al.	2005	S	LL	Vegas with Adaptation
Casablanca [79]	Saad Biaz, et al.	2005	S	LL	
NewVegas [80]	Joel Sing, et al.	2005	S	LD	
Africa [81]	Ryan King, et al.	2005	S	LD	Adaptive and Fair Rapid Increase Congestion Avoidance
CTCP [82]	Kun Tan, et al.	2005	S	LD	Compound TCP
MCP [83]	Liang Zhang, et al.	2005	S or R	LL	Mobile-host Control Protocol
Illinois [84]	Shao Liu, et al.	2006	S	LD	
Fusion [85]	Kazumi Kaneko, et al.	2007	S	LD	
CUBIC [86]	Injong Rhee, et al.	2008	S	LD	
Libra [87]	Gustavo Marfia, et al.	2010	S	LD	
TCP-FIT [88]	J. Wang, et al.	2011	S	LL, LD	
ICTCP [89]	H. Wu, et al.	2013	R	LD	Incast Congestion control for TCP
FWestwood [90]	Z. Alissa, et al.	2014	S	LL	Fuzzy Westwood

Unlike the SCP and the RCP, Y. Shu et al. [83] have proposed an alternative approach, called a mobile-host-centric transport protocol (MCP). In the MCP, the device of control can function either as the SCP or the RCP at the specific period of time. If the mobile station is the sender, then the data-acknowledgement message exchange is adopted. When the mobile station is the receiver, the request-data message exchange is applied. In particular, when the mobile station is the receiver, a flag *m_flag* of synchronization packet is set to one to notify other senders to use the receiver-centric control. Otherwise, when the mobile station is the sender, it sets the *m_flag* to be 0 and then the sender-centric control is adopted.

Another type of controlling device entity is known as hybrid centric protocol (HCP), which was introduced by K. Shi et al. [95]. TCP flow is co-ordinately controlled by both the sender and the receiver at the same time. For example, the receiver participates in flow and the congestion control by computing the *cwnd*. Then, the sender uses the receiver's information to adjust the size of *cwnd*. HCP can reduce the waiting time of the sender to alleviate the impact of timeout problem of MCP [95].

Table 2.2 shows the comparison of TCP variants according to device entity and lists the initiator(s), year of proposal, device of control, improvement aspect, and nomenclature of each TCP variant. 'S' and 'R' are used to denote a sender and a receiver device entity, respectively. As it can be seen from Table 2.2, a majority of the TCP variants is SCP. From the viewpoint of improvement aspect, most of the SCP approaches focus on the low priority (LP), the long delay (LD), and lossy link (LL). In LP, SCP divides the traffic into high and low priorities and SCP enables the sender to ensure enough resources allocated for low priority traffic. In another case, SCP uses the number of received windows to monitor the delay performance of LD networks. Similarly, SCP uses a threshold at the transport layer to trigger the packet loss rate in LL networks. However, MCP uses feedback information at the data link layer to reflect the data packet loss. Another improvement aspect is the packet reordering (PR). Both RCP and HCP allow the non-zero probability of packet reordering, and respond the out-of-order events by increasing the flow rate [57].

2.3.2.1.2.2.2 Characteristic

To elaborate the characteristics of TCP variants in sub section, three parts, feature, complexity degree, and network domain are used. The classification of TCP variants based on their characteristics is showed in Table 2.3.

Feature: Except the Tahoe, all of the TCP variants operate in four fundamental stages. Originally Tahoe had techniques for only 3 of those stages. V. Jacobson refined the Tahoe by adding the fast recovery stage and called the new TCP variant as Reno in 1990 [47]. New Reno is the enhancement of Reno. Reno is unable to control network congestion efficiently because it focuses on detecting single packet loss and can't detect multiple packet loss. Variants that cannot handle multiple packet loss include Tahoe, Reno, and Vegas. This is due to the fact that those variants avoid congestion by using the feedback information (ACKs), which is referred to as primary feedback. To mitigate this problem, New Reno and the following variants include a new feature, called multiple losses handling to detect loss of more than one packet at a time. This feature uses partial feedback information or extended feedback information. The estimation/prediction feature is added to some TCP variants to predict congestion status. The estimation parameters include bandwidth, transmission rate, and queue delay. Table 2.3 shows in the second part, that most of the TCP variants studied include both features; multiple losses handling and estimation/prediction.

Complexity Degree: Complexity degree is defined in this paper as a difficulty in terms of implementation complexity of the TCP variant. High complexity referred to TCP variants using techniques for the core four stages plus both the estimation and multiple loss detection features. While medium complexity lack the estimator feature. The low complexity is restricted to variants with techniques for only the core stages. As shown in

Table 2.3: The characteristics of TCP variants and distribution among the different types of device entity

Characteristics	Tahoe [47]	Reno [58]	Vegas [59]	SACK [60]	TD-FR [61]	NewReno [62]	DSACK [68]	Westwood [64]	Veno [65]	TCPW CRB [66]	Nice [67]	LP [68]	TCP-Real [69]	DOOR [70]	TCPW BR [71]	STCP [72]	HS-TCP [73]	FAST TCP [74]	TFRC [75]	BIC [76]	Hybla [77]	Vegas A [78]	Casablanca [79]	New Vegas [80]	Africa [81]	CTCP [82]	MCP [83]	Illinois [84]	Fusion [85]	CUBIC [86]	Libra [87]	TCP-FIT [88]	ICTCP [89]	Fwestwood [90]
Centric Control Protocol																																		
SCP	■	■	■	■		■		■	■	■	■	■			■	■	■	■		■	■	■	■	■	■	■		■	■	■	■	■		■
RCP					■		■						■																				■	
MCP																											■							
HCP														■					■															
Feature																																		
Slow start	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
Congestion avoidance	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
Fast retransmit	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■		
Fast recovery		■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■		
ACK Feedback	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
Multiple loss handling				■	■	■	■	■	■	■		■		■	■	■	■			■	■		■		■	■		■	■	■	■			
Estimation technique			■					■	■	■	■		■		■	■		■	■			■	■	■	■	■	■	■	■		■	■	■	■
Complexity Degree																																		
Low	■	■																															■	■
Medium			■	■	■	■	■				■	■	■	■		■		■	■	■	■	■		■			■			■		■		
High								■	■	■					■		■						■		■	■		■	■		■			
Network Domain																																		
Wired	■	■	■	■		■			■													■												■
Wireless					■		■	■		■			■	■	■				■				■				■					■		
High-speed/ long delay																■	■	■		■	■			■	■	■		■	■	■	■	■	■	
Low priority data transfer											■	■																						

■ denotes as selected column

Table 2.3, the TCP variants that have high complexity are more than half.

Network Domain: TCP variants are distributed into four overlapping network domains, namely *wired network*, *wireless network*, *high-speed/long delay network*, and *low priority of data transfer network*. TCP is originally designed for wired network whereby the network congestion only occurs due to the packet loss. Therefore, TCP for wired network cannot react adequately to the packet loss of wireless networks. This is because packet loss in wireless networks is mainly due to the lossy nature of radio links. Several solutions have been proposed to resolve this problem, such as network status verification by explicit congestion notification from the congested intermediate wireless terminal. Other solution uses intermediate wireless terminal to distinguish the cause of the packet loss whether wireless or wired environments.

In addition, standard TCP variants may not be very efficient in high-speed/long-delay networks. In this kind of network, the data packets are transmitted but not yet received at the receiving side due to the long distance end-to-end transmission. Those delayed data packets could be mistaken as packet loss [86]. This problem is also referred to as a bandwidth-delay product (BDP).

A low priority data transfer network is the network which consists of high and low priorities data flows. Both TCP-Nice and TCP-LP are designed to provide a guarantee of transmission rate for the low priority data flows even in the presence of high priority data flows.

In Table 2.3, the TCP variants with high complexity degree are often belonging to both wireless networks domain and high-speed/long-delay network.

2.3.2.1.2.3 TCP Variants Association

The association of TCP variants is depicted in Fig. 2.12. Tahoe was the first introduced TCP variant. The core Tahoe congestion avoidance technique reduces *cwnd* to 1 when in the congestion avoidance stage (packet loss is detected). This can lead to significant throughput degradation especially if no congestion occurred. Fast recovery (FR) technique was introduced in Reno to halve the *cwnd* and hold the new value until no duplicated acknowledgements are received within a specific period of time. Reno evolved into five different TCP variants that are specifically targeted for wired and wireless environments. The first two, NewReno and SACK solve multiple losses in the wired environment by introducing the partial ACK and selective ACK respectively. The third, TCP-Real implements contention detection and congestion avoidance in wireless environment. The fourth, Vegas uses the queue length utilization to determine the congestion status and reacts pro-actively. Vegas predicts congestion status before an actual congestion using estimation method, then uses packet delay to update *cwnd*. The last one, TCP-FIT performs gracefully in high speed/long delay wireless networks by parallel TCP techniques.

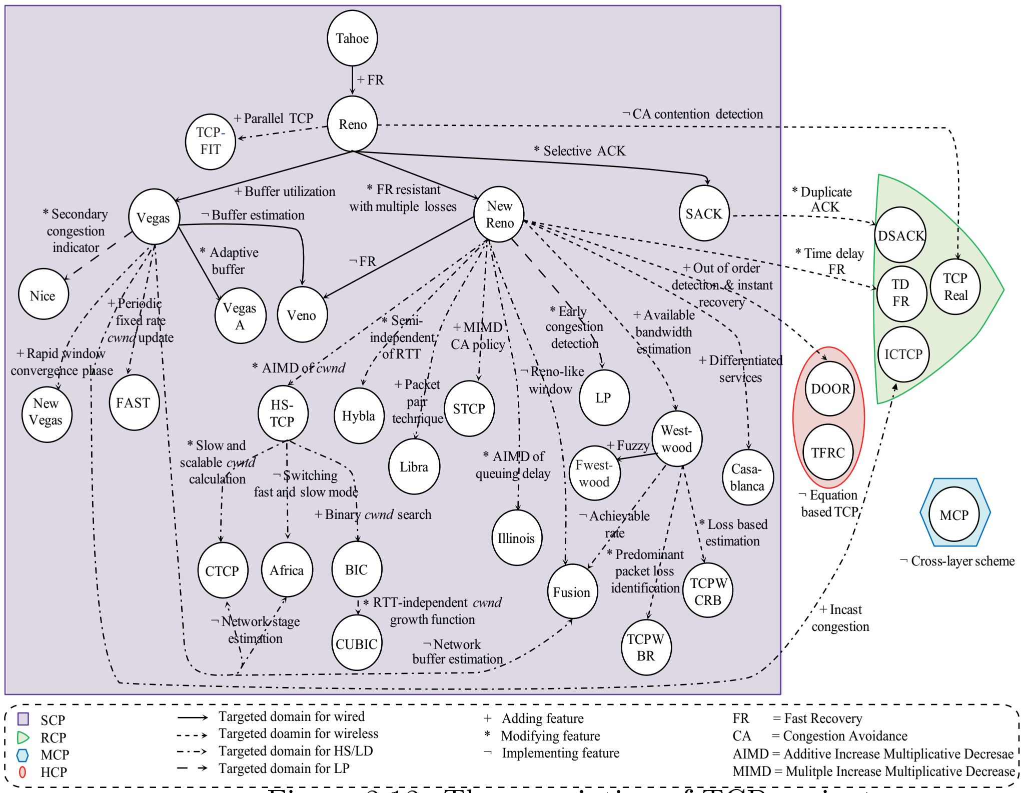

Tahoe
+ FR
Reno
TCP-FIT
+ Parallel TCP
¬ CA contention detection
* Selective ACK
SACK
* Duplicate ACK
DSACK
* Time delay FR
TD FR
TCP Real
ICTCP
+ Buffer utilization
* FR resistant with multiple losses
New Reno
Vegas
¬ Buffer estimation
* Secondary congestion indicator
* Adaptive buffer
Nice
Vegas A
Veno
¬ FR
+ Periodic fixed rate cwnd update
+ Rapid window convergence phase
New Vegas
FAST
* AIMD of cwnd
* Semi-independent of RTT
+ MIMD CA policy
* Early congestion detection
+ Available bandwidth estimation
+ Out of order detection & instant recovery
+ Differentiated services
HS-TCP
Hybla
+ Packet pair technique
STCP
¬ Reno-like window
LP
West-wood
Casa-blanca
DOOR
TFRC
¬ Equation based TCP
MCP
¬ Cross-layer scheme
Libra
* AIMD of queuing delay
+ Fuzzy
Fwest-wood
* Slow and scalable cwnd calculation
¬ Switching fast and slow mode
+ Binary cwnd search
Illinois
¬ Achievable rate
* Loss based estimation
* Predominant packet loss identification
CTCP
Africa
BIC
Fusion
TCPW CRB
TCPW BR
* RTT-independent cwnd growth function
CUBIC
¬ Network stage estimation
¬ Network buffer estimation
+ Incast congestion
SCP
RCP
MCP
HCP
Targeted domain for wired
Targeted doamin for wireless
Targeted domain for HS/LD
Targeted domain for LP
+ Adding feature
* Modifying feature
¬ Implementing feature
FR = Fast Recovery
CA = Congestion Avoidance
AIMD = Additive Increase Multiplicative Decresae
MIMD = Mulitple Increase Multiplicative Decrease

Figure 2.12: The association of TCP variants

A few TCP variants are extended from the Vegas. For example, New Vegas included rapid window convergence algorithm to reduce the convergence time and increase utilization in high-speed/long-delay network. FAST TCP is a scalable TCP variant of Vegas, it defines a periodic fixed rate *cwnd* and a delay-based congestion estimation. Vegas A proposed an additional adaptive buffer mechanism to solve the problem of improper decrement of flow rate to nearly zero under certain conditions in Vegas. The threshold coefficient in Vegas A is adaptively tuned according to the actual transmission rate, with the *cwnd* management mechanism. The variant Nice, targets priority data transfer networks.

The proactive method allows Nice to utilize more network resources for low priority data flows when high priority data flows are not using. ICTCP was designed to handle the problem of incast congestion by adjusting the TCP receiver window before packet loss happens.

One of the Reno family members, NewReno was extended into a large number of new TCP variants for all kinds of the network environment. HS-TCP, Hybla, Libra, STCP, Illinois, Fusion, CTCP, Africa, BIC, and CUBIC are the enhanced NewReno version for the high-speed/long-delay networks. The enhancement features include the Additive Increase and Multiplicative Decrease (AIMD) of *cwnd* and queuing delay, semi-independent of RTT, packet pair technique, MIMD congestion avoidance policy, binary *cwnd* search, and so on.

Besides adaptations of the NewReno for high-speed/long-delay networks, NewReno also was expanded for the wireless networks. For instance Westwood, Casablanca, DOOR, and TD-FR all extended NewReno for wireless networks. In the Westwood, a bandwidth estimation mechanism is added to speed up the fast recovery stage. This bandwidth estimation mechanism has an inherent concept from Vegas. Both use round trip time (RTT) and the amount of transmitted data packets to calculate the data transfer rate. *cwnd* and slow start threshold, *sssthresh* in the Westwood are set near to the estimated data transfer rate when the packet loss is detected. Furthermore, three TCP variants, TCPW BR, TCPW CRB and FWestwood are evolved from Westwood. The TCPW BR and TCPW CRB add the features of predominant packet loss identification and loss based estimation to Westwood's, respectively. FWestwood uses fuzzy controller to enhance performance in wired network with high error rate.

Casablanca adds the feature of differentiated service (*Diffserv*) to enable a sender to identify accurately and react properly to packet loss due to medium contention and interference. DOOR includes out of order detection and instant recovery. TD-FR uses the time delayed fast recovery mechanism to perform packet reordering.

A number of TCP variants were developed by merging the concepts and ideas from different TCP variants together. For instance, the CTCP uses network status estimation like Vegas and slow and scalable *cwnd* calculation like HS-TCP. Africa uses network status estimation of the Vegas and switching fast and slow mode of the HS-TCP. Fusion is integrating Vegas, NewReno, and Westwood. In Fusion, the features of the network buffer estimation for monitoring network status from Vegas and achievable rate from Westwood are merged. Fusion also maintains the NewReno congestion stage control mechanism.

Other TCP variants like the DSACK, which is the extension from SACK for wireless problem of misinterpreted out of order delivery. Two TCP variants do not come from the root TCP variants, those are MCP and TFRC. MCP uses the cross-layer scheme, whereas TFRC uses different mechanism to control stage by fixing the transmitted number of packets. TFRC triggers the data sending rate in terms of the packet per second in response to the network congestion status.

Fig. 2.12 shows that nearly 74% of the 34 TCP variants is the SCP, and only one TCP variant is the MCP. Furthermore, about 37% of the 34 TCP variants is for the high-speed/long-delay environment, about 25% is for the wireless environment, and the rest of percentage is for wired and low priority data transfer environments. In the association of TCP variants, most of the TCP variants of the high-speed/long-delay environment is evolved from the TCP variants of the wired environment.

2.3.2.1.2.4 TCP Variants Congestion Detection

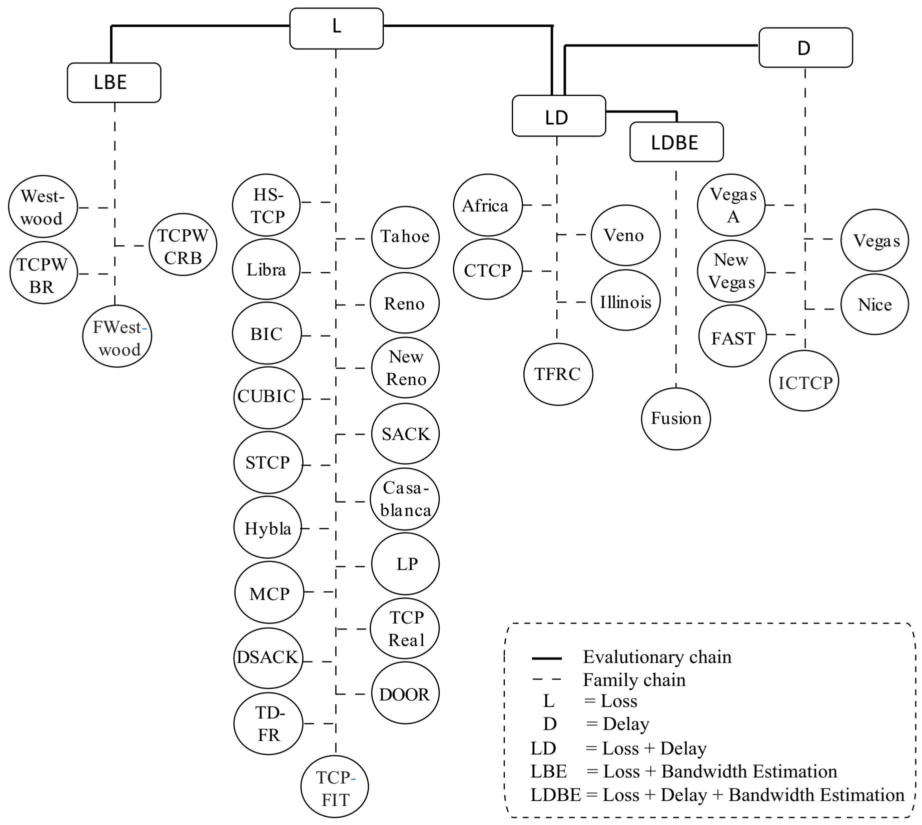


Figure 2.13: An evolutionary chain of TCP variants that shows the detection scope of the congestion control mechanism

The congestion detection technique plays an important role in the TCP variants. In this sub section, the scope of congestion detection of TCP variants is classified into two fundamental categories, namely *loss based* (L) and *delay based* (D). Fig. 2.13 shows the evolutionary chain of the congestion detection in the TCP variants. Loss based is the earliest reactive method used to detect network congestion status. Loss based was built to amend reliable end-to-end transmission at the transport protocol to deal with congestion. In the normal operation of the TCP protocol, the receiver transmits an ACK message to the sender upon successfully receiving the data packet. When the sender that receives three duplicate of ACK messages consecutively from the receiver, this indicates that the transporting network is congested and that the sender already transmitted that packet three times after 2 timeouts [58, 60, 62].

The method used in loss based tends to face long delay to send ACK messages. As a result, the loss based method might not detect the congestion problem early enough. To overcome this problem, delay based uses a proactive method, it was introduced in Vegas. In delay based, the network parameter, i.e., the RTT is used to measure the average end-to-end delay required for the sender to transport the data packets to reach the destination and receive an ACK message of that packet. RTT can be used to know the network congestion status. *cwnd* is calculated as a function of RTT to reflect to the network congestion status.

Some TCP variants use an estimation mechanism to approximate the bandwidth usage

Table 2.4: A comparison of parameters that are used in the congestion detection of TCP variants

Parameter	Device of Detection	Detection Scope	Locality
Duplication ACK	sender	L, LD, LBE, LDBE	global
Round trip time (RTT)	sender	D, LD LBE, LDBE	global
Retransmission timeout (RTO)	sender	L, D, LD, LBE, LDBE	global
Bandwidth	sender and intermediate	LBE, LDBE	global
Packet loss rate	sender	L, D, LD, LBE, LDBE	global
Internal queue length	intermediate	L, D, LD LBE, LDBE	local
Inter packet arrival time	receiver and intermediate	D, LBE, LD, LDBE	global
Retransmission time of packet	sender	L, D, LBE, LD, LDBE	global
Average delay	receiver	D, LD, LDBE	global
Jitter	receiver	D, LD, LDBE	global

of the end-to-end transmission. This mechanism is sometimes called bandwidth estimation technique, and sometimes applied in conjunction with the loss based detection, that will be called loss based estimation (LBE). The bandwidth is estimated upon reception of the ACK message by calculating RTT. The amount of acknowledged data by the ACK message is divided by the time elapsed since the last ACK message was received. The TCP variants that use this detection mechanism are the Westwood, TCPW BR, and TCPW CRB.

Another detection mechanism, *hybrid based* that is defined as the combination of both loss based and delay based (LD) was introduced right after the bandwidth estimation. Three duplication of ACK messages or the RTT parameter cannot solely determine network congestion status. Therefore, hybrid based was developed to overcome the weakness of either loss based or the delay based mechanisms. In the hybrid based, the TCP variants use delay based mechanism to identify the network status. When the network is suspected, congestion is further confirmed by using loss based mechanism. Veno is a typical variant example using hybrid based mechanism. However, the TCP Fusion combines the hybrid based mechanism with bandwidth (LDBE) estimation to detect congestion [85].

So far, the detection scope of the TCP variants is discussed. Selecting an appropriate parameter is essential to detect network congestion. The list of the parameters that are used in the congestion detection of TCP variants is shown in Table 2.4.

The duplication of ACK uses three duplicate ACK messages, which are generated by the receiver. The RTT is the time required for a short packet to travel from a sender to a receiver and back again to the sender. The RTT parameter is also used to determine other parameters, such as queue delay and achievable transmission rate. The retransmission timeout (RTO) is the time required for the same packet to be retransmitted. The bandwidth is the transmission rate of data through a communication link. The packet loss rate is the rate of the data packets travelling across a network that fail to reach their destination. The internal queue length is the buffer size allocated to each terminal in the network. The inter packet arrival time is how much time elapses from when the last bit of the first packet arrives until the last bit of the second packet arrives. The retransmission time of a packet is the time required for a retransmitting packet to travel from a sender to a receiver and back again to the sender. The average delay is the sum of the delays encountered by a packet between the time of insertion into the network and the time of delivery to the destination. The jitter is the variation in latency as measured in the variability over time of the packet latency across a network.

The aforementioned detection parameters used in each detection mechanism are selected based on several factors, including network structure, traffic pattern, transmission rate, network application, QoS requirements, and congestion probability [96]. In this paper, the congestion detection parameters are further classified into device of detection, detection scope, and location. The device of detection specifies which terminal is involved with the congestion parameter. The terminal can be a sender, a receiver, or an intermediate. From the Table 2.4, most of the congestion parameters are at the sender side. Only three congestion parameters are at the receiver or intermediate. It is also found that most of the parameters are used for delay based and estimation mechanism. As it can be observed from Table 2.4, only parameter internal queue length is local. Local means that a parameter is measured within a device and used to detect the congestion status. This allows faster response in the congested node. This requires per-flow state information in the local node, which limits scalability. Whereas global parameter is shared among network devices and used to detect congestion. This will result in slow response to congestion. The parameter information reflecting congestion may have to travel towards all the way to the receiver and even worse sometimes back to the sender.

In summary, several detection parameters are used, but this accuracy might sometimes not fable. Detection parameters such as duplication ACK, RTT, bandwidth and packet loss rate are mainly based on the feedback of acknowledgement packets. The acknowledgement packets can be lost or delayed due to non-congestion factors during the transmission. As a result, congestion might be falsely detected and lead to unnecessary congestion handling.

2.3.2.1.2.5 TCP Variants Congestion Avoidance

In this section, mechanisms used during the congestion avoidance (CA) stage of the TCP variants are discussed. In the CA mechanism, the congestion window (*cwnd*) size is the main parameter to be controlled and monitored. Generally, *cwnd* is started with exponential increase mechanism during slow start stage. When the network congestion is detected using one of the detection mechanisms, the TCP will enter into the CA stage. In the CA stage, the *cwnd* size is reduced or increase according to the additive or multiplicative way. Except the Tahoe, all the TCP variants at the CA stage immediately reduce the *cwnd* size and then increase the *cwnd* size based on the policy of the additive increase (AI) or the multiplicative increase (MI). Tahoe directly sets *cwnd* to zero when the detection mechanism receives three duplicated ACK.

Table 2.5: The dependency of previous *cwnd* size when the CA algorithm of TCP variants is used

Dependency of Old *cwnd* Size	Function, $f(cwnd)$	
	w/ Condition	w/o Condition
Direct Dependent	Vegas, Nice, Vegas A, Veno, New Vegas, STCP, Africa, TCP-Real, FAST TCP, CTCP, Hybla, Illinois, Libra, Fusion, Westwood (Mode 1), TCPW BR (Mode 1), TCPW CRB (Mode 1), ICTCP	Tahoe, Reno, NewReno, SACK, LP, Casablanca, HS-TCP, TD-FR, DSACK, DOOR, MCP, TCP-FIT
Indirect Dependent	Westwood (Mode 2), TCPW BR (Mode 2), TCPW CRB (Mode 2), FWestwood	None

The mathematical representations of the AI and MI used in the CA stage to increment *cwnd* are offered in Table 2.6. The increment of the *cwnd* size gives an impact to the network throughput while the CA mechanism tries to avoid the network congestion.

Table 2.5 classifies TCP variants according to the effect of their previous *cwnd* size when the CA mechanism is used. Dependency in Table 2.5 refers to the function of the *cwnd* that depends on the old *cwnd* size or not in order to decide the new *cwnd*. The condition refers to the function of the *cwnd* that bounds with any constraints or not. From the Table 2.5, it can be observed that most of the function of the *cwnd* depends on the old *cwnd* size. For example, the Tahoe starts to update the new *cwnd* size without condition based on $cwnd_{new} = cwnd_{old} + \frac{1}{cwnd_{old}}$ when the packet loss is detected. But, the Vegas will need to ensure the condition $\alpha > cwnd_{old} \times \frac{RTT - RTT_{min}}{RTT}$ is fulfilled in order to enter the CA stage with $cwnd_{new} = cwnd_{old} + 1$. Meanwhile, only the Westwood, the TCPW BR, and the TCPW CRB are using other parameters to determine the new *cwnd* size with condition in their CA algorithms.

Table 2.6 shows the methods of additive increase, multiplicative increase, and equation-based of the function, *f(cwnd)*. Some TCP variants consist of two modes in their CA algorithms, e.g., Westwood, TCPW BR, TCPW CRB, TCP-Real, Africa, CTCP. These two modes that are operating in the CA algorithms depend on some constraints and conditions. The first component and second component of the function, $f(cwnd)$ is defined as α and β to represent the increment policies of additive and multiplicative, respectively. The α in the additive increase is totally depending on the old *cwnd* size. And the β in the additive increase is depending on the zero, constant, scaling, quotient, and other. If the β is zero, the new *cwnd* size is equal to the old *cwnd* size. If the β is constant, the new *cwnd* size can be a function that varies with a fixed value, such as Libra, Vegas, Vegas A, New Vegas, and Nice. If the β is scaling, e.g., $\gamma \times cwnd_{old}$, the new *cwnd* size can be a function that varies with a factor. For instance, $\gamma = 0.125$ in STCP [72] and $\gamma = 0.01$ in Africa (Mode 1) [81]. If the β is equal to a quotient, e.g., $\omega / cwnd_{old}$, the new *cwnd* size can be a function that varies with ω, in which can be 1. If the β is other, the new *cwnd* size can be a function that varies with an estimation value. On the other hand, the α in the multiplicative increase can be divided into two types, i.e., bandwidth estimation (BE) and rate estimation (RE). But, these two types of the multiplicative increase share

Table 2.6: The methods of additive increase, multiplicative increase, and equation-based of the function, *f(cwnd)*

Method	$f(cwnd)$		TCP Variant
	α	β	
Additive Increase (AI) $cwnd_{new} = \alpha + \beta$	*cwnd*	Zero	Westwood (Mode 1), TCPW BR (Mode 1), TCPW CRB (Mode 1), TCP-Real* (Mode 1), Fusion (Mode 1), FWestwood* (Mode 1)
	cwnd	Constant	Vegas, Vegas A, New Vegas, Nice, FAST TCP, Fusion (Mode 2), ICTCP*, TCP-FIT (Mode 1)
	cwnd	Scaling	STCP, Africa (Mode 2), CTCP (Mode 1), TCP-FIT (Mode 2)
	cwnd	Quotient	Tahoe, Reno, NewReno, SACK, HS-TCP, TD-FR*, DSACK*, Veno, DOOR***, Hybla, Libra, Africa (Mode 1), TCP-Real* (Mode 2), CTCP (Mode 2), Fusion (Mode 3), MCP**
	cwnd	Other	Illinois
Multiplicative Increase (MI) $cwnd_{new} = \alpha \times \beta$	BE	RTT_{min}	Westwood (Mode 1), TCPW BR (Mode 2), FWestwood (Mode 2)
	RE	RTT_{min}	TCPW CRB (Mode 2)
Equation-based	*cwnd*		BIC, CUBIC, TRFC***

w/o footnote mark = SCP, * = RCP, ** = MCP, *** = HCP

the common β of RTT_{min}. An alternative method that is found in the Table 2.6 is an equation-based method. In the equation-based method, the new *cwnd* size is determined by a new form equation, which depends on the maximum or minimum limit of *cwnd* size and the computed end-to-end throughput.

In summary, the function, $f(cwnd)$ of the CA algorithms depends on the network structure, traffic pattern, type of application, and QoS requirement. In other words, the multiplicative increase policy of new *cwnd* size can be recovered the network throughput faster than the additive increase policy. However, the multiplicative increase policy easily causes the network to be congested again.

Overall, congestion management in TCP/IP networks is simple to implement, particularly end-to-end system that requires minimum participation from the edge or core devices. This also allows the TCP congestion management to work over heterogeneous networks. TCP congestion control can be self-pacing, as packet drops is detected, the transmission rate can be adjusted accordingly. This means the packets are constantly transmitted at the optimal bandwidth of the current connection.

However, the congestion management in TCP/IP cannot provide guaranteed bandwidth that is required for QoS. This is why many TCP variants need to enforce the fairness among the connection. Besides, the congestion control in TCP is type of traffic synchronization effects. This is because the TCP variants use dropped packets as their main measurement of congestion status. A burst of dropped packets might be caused by small buffer drop-tail router, but this can lead to cycles of underutilization, window increase, and then tail drop again. The TCP traffic bursts might be also due to reverse path routes of acknowledgement; those acknowledgement packet bursts occupy valuable resources and could cause timeouts and retransmission of others data packets.

2.3.2.2 Traffic Management in Other None-TCP Transport Layer Protocols

There are few traffic management techniques in TCP/IP network that can help in the releasing of the congested network indirectly. For example, Resource ReSerVation Protocol (RSVP), Differentiated Services (DiffServ), queuing techniques, and Multi-Protocol Label Switching-Traffic Engineering (MPLS-TE).

The RSVP is a state establishment for real time services in the Internet. It does not provide network service; it just communicates any end device requirement to the network. Through RSVP, applications inform their needs to the network and their traffic characteristics to the receivers. RSVP is the signalling protocol which establishes and manages the reservation state information at each router of the path. RSVP engages on the top Internet protocol layer, and responsible to QoS of packets forwarded based on routing. RSVP enables the receivers to make different reservations, handles route changes adaptively, allows the users to specify their needs to efficiently utilize the network resource, and control the overhead by specific parameters.

The DiffServ is a protocol that is used for specifying and controlling the network traffic through classifying the types of traffic in the networks and forwarding it according to the predefined way. The DiffServ uses more complicated policy or rule statements to manage the way to forward a given network packet. In DiffServ, the network traffic can be classified into alert message, control message, and monitoring data. The monitoring data includes video data, query based response data, and periodic reporting.

The queuing techniques such as Fist In First Out (FIFO), priority queuing, fair queuing, weighted fair queuing, class based queuing, and so on. The appropriate queuing method use in the network can reduce the packet waiting time and speed up service time. By this, the network throughput will be increased.

The MPLS-TE is designed to reduce the overall cost of operations through efficient use of network bandwidth. MPLS-TE allows the constructed routes for traffic streams within a service provider. This can solve the problem of some parts of the network to be over-utilized and other to be under-utilized. However, MPLS-TE might need extra devices and complex network management.

2.4 Summary

The congestion control mechanism have several similarities and differences to reduce traffic load when congestion is detected by using additive and/or multiplicative decrease to control the sender transmission rate.

First a comparison is due here, between the networks example mentioned in this chapter representing both network types; congestion free and congestion vulnerable networks. ATM networks can provide a flexible grained and fair level for network transmission. However, it requires complicated switches functionality compared to regular Ethernet switches but still far simpler than routers. It also needs longer administration control to negotiate QoS for the guaranteed requirement. TCP congestion control cannot get rid of congestion completely like the case of ATM in the congestion free network, resulting in much lower QoS in TCP/IP transmission. Due to the usual expensive faster links of ATM, the cost to implement and maintain such ATM networks is quite high especially when existing TCP/IP networks should be replaced first. This leads to still preferring TCP/IP over ATM when not much QoS is required. Congestion free network are usually restricted when QoS is vital, by spending more time and cost for design and implementation as well as longer operation periods for QoS negotiation prior to connections establishment. TCP/IP is the most common congestion vulnerable network if not the only one still existing. It has been always the de-facto choice for the enormous Internet. ATM and such congestion free networks are utilized to boost the performance of Internet core among other vital Internet resources.

This chapter have extensively reviewed the congestion management showing that current control of the transmission rate of the sender is the main if not the only way of resolving congestion. Current congestion managements are still not sufficient to totally eliminate congestion from congestion vulnerable networks. That is why congestion free networks were designed. Congestion handling mechanisms are still emerging, hoping to find new ways to resolve congestion faster or better avoidance of its occurrence.

Chapter 3

Data Compression

In this chapter, background knowledge about data compression is presented, where different data compression schemes are discussed and classified. Part-wise data compression is further analysed since it will be utilized in the rest of the thesis.

3.1 Introduction

Data compression is converting data into another format that requires less storage (more efficient) than the original format with some satisfactory accuracy. It is considered as one of the information encoding techniques. Sometimes it is also referred to as bit-rate reduction when applied in networking. In 1848, Morse Code was introduced, which is considered to be the first modern data compression [97–99].

Data compression theory is an extension of the basic information theory, same as all any encoding. Data compression is mainly focused on statistical inference information theory. Compression can be either lossless or lossy. Compressed data from lossless compression must be decompressed to exactly its original value. Compression belongs to algorithmic information theory category for lossless compression and rate-distortion information theory for lossy compression. Both areas were established by Claude Shannon, in the late 1940s and early 1950s as the base for all communication, signalling and data handling [100, 101]. Shannon Fano (SF) coding was the first compression scheme built based on information theory.

When a data unit or sequence of units from the source is compressed, the resulting compressed representation will be referred to as the representing code within this thesis. The representing code should achieve the desired target, either faster transmission, less storage or less energy, performance and degree of compression of unit by unit individually could vary and rarely considered by itself. The aggregate overall compression degree is more significant for the whole data together with the overall performance specially when compressing data from different sources as studied in 3.4.1.

Designers of data compression schemes have to handle tight trade-off between the conflicting targets. Those targets are the degree of compression, the computational resources required (time, temporary storage and energy). Lossy compression faces an additional target, the amount of distortion introduced, which is highly dependent on the degree of compression. The suitable position in the trade-off limited space is usually decided during design or implementation to target the application of the compression scheme according to a specific situation [99, 102].

When data compression is applied for computer networks, compression is traditionally activated manually by the user at the sender end devices before transferring. When the receiver end devices receive the compressed data, the user again activates decompression manually. Sometimes data compression functionality is embedded in computer applications or lower layers of end devices. Either way, compression has been largely restricted to end-to-end use. The compression performance could affect the overall network performance.

3.2 Classifications of Data Compression Schemes

Nine different perspectives are used in this section for classifying the different compression schemes. Generally, the classifications divide the compression schemes into two or more categories which have many-to-many relations between all categories from all the other classifications. The resulting overall categories form a hypercube of degree nine.

1. According to the heterogeneity: the first classification divides compression schemes into mixed (heterogeneous) versus homogeneous data. The classification depends on whether the data compressed originated from different sources or the same source. Heterogeneity also refers to the structure of the data itself, whether homogeneous having the same structure or mixed structures. Homogeneous data usually exhibit much better compression performance and degree compared to mixed data. Since mixing is usually done randomly and out of order without any synchronization, mixed data usually results are around those of the worst part of the mix with lowest correlation with respect to both its compression performance and degree. Thus mixed data are usually handled by part-wise (blocking) compression to reduce the effect of lack of correlation caused by the mixing itself. Almost all known commercial compression tools utilize part-wise compression to handle general data. Whatever the way the different data from different sources are mixed, the resulting mix is totally unpredictable ranging from much higher to much lower correlation that each data source separately. Compressing the data from the different sources separately gives much more predictable performance, will be studied in section 3.4.1. For example, the two data sources *abab* originally having repetition $2 \times ab$ and *cdcd* having $2 \times cd$, if mixed could give *acbdcadb* with no repetition at all.

2. According to the purpose of compression: the two main purposes are either storage or communication. Storage compression schemes target smaller final storage while sacrificing computational costs of compression; compression time, compression computational energy and temporary storage. This class of compression schemes are usually used for archiving. Both categories handle either homogeneous or mixed data. Online archiving overcomes the long delay encountered by utilizing heavy caching mechanisms. On the other hand communication compression schemes target much faster compression time as well as lower compression computational energy. Usually communication compression schemes sacrifice compression degree to achieve the strict requirements. Most network coding utilizes power versus speed trade-off to choose the suitable bit per symbol rate.

3. According to the required accuracy: compression schemes can be generally classified into lossless and lossy. Lossy compression is used when some information loss can be

accepted depending on the requirements of usage. Lossy schemes are used for communication purposes more than for storage purposes. For instance, when compressing still images, the human eye is more sensitive to subtle variations in luminance than variation in colour. Thus JPEG image compression works by "rounding off" less important information to more coarse quantization levels. Lossless compression is used when the data decompressed must match exactly the original data before compression. Lossy compression is also widely used in the lower layer basic network coding compression schemes, which is already highly tolerant to noise i.e., accuracy loss. Lossless compression is used for critical control data among other usages, such as IP header compression.

4. According to the structuring of the data compressed: structured data units usually exhibits repeated (and usually redundant) fields from one data unit to another. That redundancy can be easily totally eliminated by taking advantage of knowing the data structure format. Representing data relatively instead of using absolute reference is the most common methodology to eliminate the redundancy in the repeated data structures. On the other hand, raw data also known as unstructured data lack any data structure format and cannot be handled using those compression schemes. Un-structured data is merely considered as a stream of similar data unit of unknown internal structure. Most of the lossy compression schemes and network encoding are mainly structure data compression associated with other compression schemes. For unstructured data, compression schemes try to detect repetitions of the whole data unit regardless any internal data structure. IP header compression and data referencing in computer programming are among the well-known areas of using structured data compression. Un-structured compression includes a lot of schemes started from the original Morse code passing through the well-known LZ schemes. Generally, unstructured compression schemes try to remove frequent repetitions of data units by encoding that repetition using some smaller structure.

5. According to the repetition distance: the distance between repetitions targeted for compression can be used to distinguish compression schemes. Consecutive compression schemes target repetitions that are consecutive between either single data units or sequences of data units. Distributed encoding structures are used to represent consecutive repetitions more effectively. Consequence compression schemes are better used when some data is available ahead, but still can be used when not available with less efficiency. Non-consecutive compression schemes are used when the distance between the repetitions is longer, other common dictionary like structures are used as look up tables. Methods of building those dictionary structured are varies a lot from one scheme to another, trees (Huffman), arrays, tables, lists and so on. When the data size is quite big, a pre-defined size sample of the whole data is scanned to statistically or probabilistically detect the repetitions. The size of data, which that sample is representing, is also different from one scheme to another. Both consequence and non-consequence schemes can be applied for either structured or unstructured data. RLE scheme, as the one implemented in BMP images, is probably the most common consecutive compression scheme. An example of non-consecutive schemes, colour palettes of most image and video data formats are considered a form of common dictionary for a more efficient representation of repeated colours.

6. According to the encoding/decoding structure sharing: the dictionary like structures generated before or during compression for encoding is sometimes attached with the compressed code for either storage or transmission such as colour palettes of image and video compressed data formats as well as most consecutive compression schemes. The dictionary like structures can be either one big structure as the colour palettes in images or distributed smaller structures such as run-length encoding fields preceding compressed runs of data in RLE schemes. When the dictionary structure is attached, it will affect the compression degree since the compressed code size will include the dictionary size. Accordingly, a practical size limit is usually imposed on the dictionary size so that compressed code size do not to exceed the input code size. Other compression schemes do not need to attach the dictionary structures and dispose the generated structures; LZ schemes are an example of this category. The decoders of those schemes regenerate the dictionary during decompression without any prior knowledge needed about the dictionary used for compression. When the dictionary is regenerated, additional computational resources are utilized, time, energy and temporary storage. Since the dictionary size does not directly affect the compression degree, the dictionary size can virtually grow infinitely to capture more repetition and achieve better compression degree. In some compression schemes, the dictionary is either fixed or managed independent of data by explicit periodic synchronization mechanisms, Morse code, most lossy schemes and network coding are examples of the former while Compressed Real-time Transport Protocol (CRTP) is an example of the latter. Fixed dictionary cannot adapt to different data contents and can miss lot of repetition resulting in poor compression degree. On the other hand, fixed dictionaries are sorted and optimized for faster access, thus faster encoding/decoding. Periodic dictionary synchronization causes a lot of accesses to the replicated dictionary structures, which adds a lot of additional load to the storage or traffic to the network.

7. According to the number of passes scanning the input for compression: optimal compression schemes require a pre-parse of the input code (at least part of it, sample) to build the dictionary structure that will be used for compression. Additionally, some schemes calculate or predict the final compression degree statistically or probabilistically to decide compression feasibility. Some schemes use more than one pass to perform such pre-processing to prepare the required structures before starting the final real compression pass. LZ Sorter Szymanski (LZSS) decides if the feasibility of the expected compression in a separate pre-pass, likewise arithmetic like schemes builds the dictionary structures in a separate pre-pass. Most literature refer to the former multi-pass compression schemes as statistical compression, since only small sample size ratio are used during the pre-passes. Other compression schemes perform only one pass during which the dictionary structure is built simultaneously with the compression itself, such as most of the LZ schemes. Multi-pass compression schemes usually required longer compression time to achieve better compression degree compared with single compression schemes by adding overhead pass(es). Higher sampling frequencies as well as larger sample size ratios can up scale the effect of this overhead on the compression time. Multi-pass schemes generally achieve better compression degree for heterogeneous data compression to better capture the non-homogeneous mix of data being compressed.

8. According to the sampling frequency (blocking): when the data is too big, parts of the data are scanned or sampled independent of other data parts to detect repetitions in that part alone. Morse code like schemes scan or scanned a chosen representative part of the data as a sample to stand for all the data in the world for eternity. On the other hand, LZ like schemes scans every data part to statistically obtain its independent repetition representation. When the data size grows above some limit most schemes start blocking data and handling it part by part (part-wise compression), each block is sampled separately. Different sampling frequencies are usually used in any consecutive and non-consecutive compression schemes whether data is structured or not. The bigger the size of each sample, the lower the frequency. The combination of sampling frequency and sample size ratio for any compression schemes strongly affects its compression degree and performance. The less homogeneous the data is, the higher sampling frequency should be with the same sample size ratio.

9. According to the sample size ratio: the ratio of the size between the scanned sample and the part of the data it is representing. Some data compression schemes scan only a percentage of the whole data (part or block) to find repetitions statistically or probabilistically, the detected repetitions will be used to represent the whole data. Morse code scheme used infinitely growing data size with one time only sample frequency, resulting in almost zero sampling size ratio. Most LZ like schemes used 100% sample size ratio, by scanning the whole data or block to find all the existing repetitions. Huffman like schemes choose lower percentage for sample size ratio. The less homogeneous the data is, the higher the sample size ratio should be within the same sampling frequency. Choosing the suitable combination of both sampling frequency and sample size ratio for any compression schemes is strongly affected by the information entropy and correlation within the compressed data. Periodic dictionary synchronization schemes are usually combined with huge data (blocks). Where fixed dictionary structure schemes are limited for near zero sample size ratios, usually the small sample is good enough to represent the whole data (block).

3.3 Examples of Data Compression Schemes

This section presents six different examples representing a lot of the joint data compression schemes discussed in the previous section. JPEG compression schemes are first discussed to represent both categories of lossy compression as well as structured compression schemes. Consecutive compression is represented afterwards by the RLE compression scheme in the following sub section. Huffman, arithmetic and LZ are explained as general examples in the succeeding sub section. The last sub section introduces an overview of various compression schemes designed for communication purpose.

3.3.1 Joint Photographic Experts Group (JPEG) Schemes

As aforementioned, lossy compression can accept some information loss depending on the quality of data required. JPEG exploits the limited human eye capability of sensing subtle variations in colour (chroma) by “rounding off” such less important information using chroma subsampling while keeping luminous (luma) information intact. In addition, other commonly used compression algorithms are utilized in JPEG schemes. Some of those are transform codecs and predictive codecs. The transform codec compression is applied on blocks of a JPEG image, transforming it from spatial domain to frequency domain using discrete cosine transforms, relying on the fact that frequency domain representation requires far less accuracy than spatial domain representation of most images. Afterwards, predictive codecs reduces the required size for representing from previous and/or subsequent next data or blocks of the image.

JPEG schemes are lossy compression with a few lossless exceptions, which is applied for the generally homogeneous structured data. The pixels colour data structures as well as locality of image areas structure are heavily exploited by structured compression schemes. Still image compress standards of JPEG target storage efficiency, while motion pictures target both storage and communication such video conference streaming applications. Both JPEG standards also utilize consecutive compression schemes. Distance compression schemes (palette dictionaries) are optionally used and attached to the compressed code when sharing. Although one pass compression can be implemented for most JPEG schemes, it can be quite a complicated project. Most JPEG schemes divides big image data into small 8×8 (64 pixels) to which the main compression is applied with 100% sample size ratio per block.

3.3.2 Run-Length Encoding (RLE) Scheme

RLE is a simple and popular data compression scheme, based on replacing long repeated data unit(s) by a shorter representation [103]. For example, consider the source data $AAAABBBC$, the encoded data can be more effectively represented as $4A3B1C$. The original 8-byte message, reduced to 6-bytes (20% saving). Sequences of data units can be similarly compressed from *banana* to $1(b)2(an)1(a)$. RLE schemes compresses the runs of the data and do not modify non-repeated data usually using some flagging symbol mechanisms, i.e., the single repetition is left as is without any of the additional storage shown in the explanation for clarity only.

If the number of repetition is large enough, the size of the output data will be significantly reduced [104]. In contrast, the size of output data can significantly be increased in worst case. Therefore, RLE is usually used as a pre-compression pass for other more sophisticated compression schemes such as Huffman and arithmetic. RLE is usually easy to implement and does not require much compression resources. RLE is only efficient with files that contain lots of repeated consecutive data, such as lots of consecutive spaces in text files or the images files that contain large areas of a single colour as in computer generated images. Microsoft BitMaP (BMP) is probably the most common example of depending mainly on RLE compression schemes.

RLE schemes are lossless compression, which is applied for the generally homogeneous consecutive unstructured data (the internal structure is not utilized). The locality of image areas structures are the only exploited feature to achieve the compression. A lot of network encoding makes use of RLE similar schemes to handle phase ambiguity problems. RLE scheme are also used for efficient storage as in BMP and the now almost history, archive tapes. RLE schemes are quite easily implemented as a single pass coder using distributed attached dictionary like structure. RLE schemes require look-ahead mechanisms that are usually far smaller than to be considered as an extra pass without any real sampling (100% sample size ratio).

3.3.3 Huffman Scheme

The probability of occurrence of a specific value in a data unit is estimated in a pre-pass, the different values of representing codes have different sizes. Smaller values of representing codes are assigned to values of data units of higher occurrence probability, and vice versa. After the probabilities have been computed, a binary tree is built with the data unit values are acting as leaves based on their probability and the paths are acting as the representing code assigned for that specific data unit value. For example, the source data *kkkkhoho* results into the tree in Fig. 3.1. The labels on the edges is the assigned representing code for each data unit value, while the probability of that data unit is shown in the leaf node together with that data unit value. The path from the root of the tree until that leaf is the full assigned representing code for that data unit value.

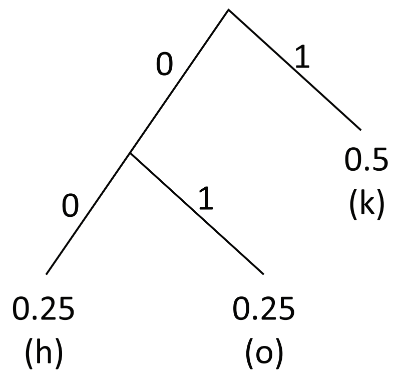


Figure 3.1: Huffman tree example

In this example one data unit is eight bits, making the total source data is 64 bits. By using Huffman coding, the data unit value k is represented only by one bit, while both h and o are represented by two bits. The final compressed size becomes 12 bits. Still, compressed representations must be attached with the formed dictionary like structure, either in tree form or table. The size of that structure varies a lot and depends on its implementation, the final compression degree depends on the summation of the size both compressed code representation as well as the accompanying dictionary structure.

Two types of Huffman schemes are widely used; static Huffman and adaptive Huffman. Static Huffman is the original simple form that has just been mentioned. Adaptive Huffman, the probabilities and trees keeps changing as the input data is scanned, a flag symbol is used to inform the decoder of positions tree updates. This will require more complicated tree bookkeeping but avoid the need of attaching the tree (dictionary) with the compressed representation and damaging the compression degree when the tree is quite big [103, 105, 106].

Huffman schemes are lossless compression, which is applied for the generally homogeneous consecutive unstructured data. Due to the long pre-pass required or enormous tree bookkeeping, Huffman schemes are restricted to storage usage only. Adaptive Huffman schemes with tree regeneration can be done in a single pass unlike the regular static Huffman with he attached dictionary. Huffman schemes must scan the whole source data in the pre-pass (100% sample size ratio), some practical considerations can divide large input into smaller blocks according to the available resources and limitations.

3.3.4 Arithmetic Scheme

Arithmetic compression scheme uses only one very long high accuracy fraction to represent many source data units [103, 105, 106]. The occurrence probability and cumulative probability are used to calculate that representing code fraction. Arithmetic coding provides a good compression degree, but it is more complicated and needs higher compression resources. The attached dictionary structure with the single high accuracy fraction should still be much small representation than the original input.

Arithmetic compression is mainly lossless compression unless fraction accuracy loss can be accepted. Arithmetic compression is applied for the generally homogeneous consecutive unstructured data. Due to the long pre-pass required, arithmetic schemes are restricted to storage usage only. The attached dictionary represents a significant percentage of the compressed code size. The whole source data must be scanned first in the pre-pass (100% sample size ratio), again for some practical considerations it can divide large input into smaller blocks according to the available resources and limitations.

3.3.5 Lempel Ziv (LZ) Schemes

This scheme is the inspiration of the compression scheme proposed in Chapter 6, and will be more thoroughly discussed. LZ is mainly designed for lossless compression but can be easily modified to for lossy applications. LZ is applied for the generally homogeneous consecutive unstructured data. LZ is used in regular storage purposes; it is virtually the standard of practical lossless data compression. LZ is also suitable for communication application owing to the facts that its schemes (fast-LZ and ultra-fast-LZ) are probably the fastest existing compression schemes and also its dictionary regeneration capability. With no pre-pass needed in most of LZ schemes the input source data is parsed only one time and the dictionary keeps including all the inputs scanned resulting 100% sample size ratio with once per input data block sampling.

The first LZ algorithm was proposed by Abraham Lempel and Jacob Ziv in 1977 and 1978 [105]. It was first designed for text data with intensive string matching. LZ adds each newly encountered data unit (character) to the enormous lookup dictionary while parsing the source data. It eliminates long repeated characters representing it entirely by much shorter pointers from the dictionary. If a match is identified, the reference to the position in the dictionary is used as the representing code, otherwise, the input data string is output as it is and appended to the dictionary. This process is repeated until the source data string is entirely encoded. Although the basic algorithm assumes infinitely growing dictionary, a lot of practical considerations limit its size. Many approaches have been utilized in the different LZ schemes to manage the drawbacks of limiting the dictionary size or allowing more efficient dictionary structures. Different LZ schemes were also proposed to improve different goals, compression degree, compression time, temporary storage or other computation resources. Fig. 3.2 shows the evolution of LZ algorithm.

The LZ schemes can be generally divided into two types, sliding window and parsing. In sliding window based LZ schemes, the size of both dictionary entries and data unit sequences are limited, while parsing schemes limit the number of entries in the dictionary.

The LZ77 scheme is the basic sliding window LZ scheme, later a lot of schemes were proposed to extend it. Lempel-Ziv-Rodel (LZR) is an extension of LZ77 to trying to achieve a linear time compression, using special pointer mechanism which consumes infeasible memory. Lempel-Ziv-Storer-Szymanski (LZSS) scheme is another extension of

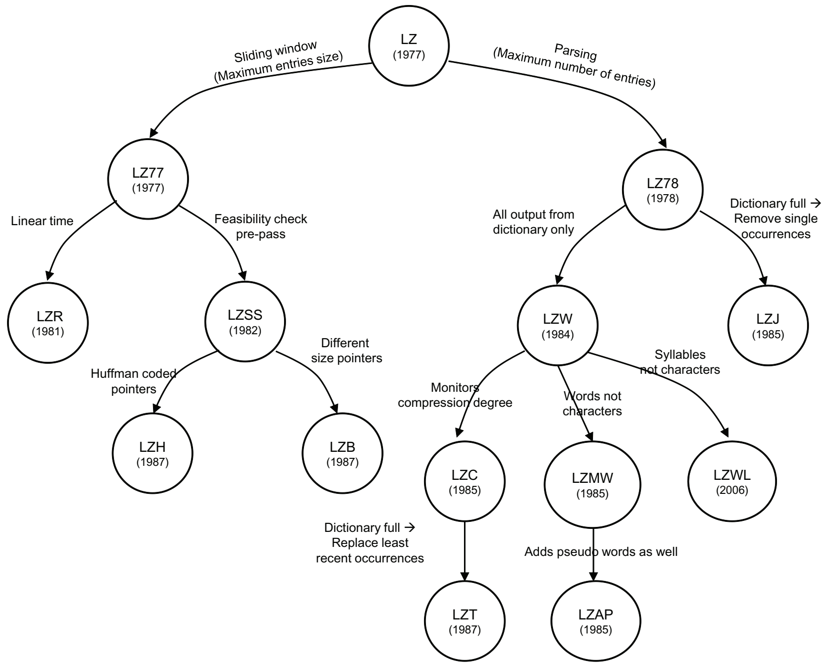

LZ (1977)
Sliding window (Maximum entries size)
Parsing (Maximum number of entries)
LZ77 (1977)
LZ78 (1978)
Linear time
Feasibility check pre-pass
All output from dictionary only
Dictionary full → Remove single occurrences
LZR (1981)
LZSS (1982)
LZW (1984)
LZJ (1985)
Huffman coded pointers
Different size pointers
Monitors compression degree
Words not characters
Syllables not characters
LZH (1987)
LZB (1987)
LZC (1985)
LZMW (1985)
LZWL (2006)
Dictionary full → Replace least recent occurrences
Adds pseudo words as well
LZT (1987)
LZAP (1985)

Figure 3.2: The evolution of LZ algorithm

LZ77 with compression feasibility pre-pass and uses fixed size pointer (dictionary indices). Lempel-Ziv-Huffman (LZH) and LZB are both extensions of LZSS which represent the dictionary pointer in with more efficient coding. LZH uses Huffman scheme to compress the dictionary pointers (indices) to increase the compression ratio, while LZB [107] uses different pointer sizes and assigning shorter pointer to more frequently occurring data unit sequences. Both LZH and LZB required an addition pass after the regular LZSS pass to optimize the obtained pointers representation in the compressed code than the original LZSS fixed size pointers. Both LZH and LZB achieve higher compression ratio than LZSS, but take more time due to the extra encoding pass for pointers. The comparisons between both LZH and LZB haven't settled which one outperforms the others.

Parsing based LZ schemes search for the longest match of data unit sequences without limitation like sliding window schemes. LZ78, the first parsing based LZ scheme, tries to match the entire previous sequence of data unit in the dictionary. Instead, the parsing schemes restrict the dictionary according to the number of entries allowed at any moments. Lempel-Ziv-Welch (LZW) is the most commonly used LZ78 extension; it makes the output entirely out of pointers by including every character in the dictionary before starting. Lempel-Ziv-Compress (LZC) extends LZW by including monitoring function of the compression degree; the dictionary is discarded and rebuilt once reaching a certain threshold. Lempel-Ziv-Ticher (LZT) improves LZC by replacing the least recent dictionary entries when the dictionary is full. Lempel-Ziv-Miller-Wegman (LZMW) operates on word basis instead of data unit basis (character) which makes it limited to only text input. Whereas Lempel-Ziv-All-Prefixes (LZAP) enhances LZMW by also adding additional words and pseudo-words to dictionary when two words are matched from dictionary right end-to-end. Similarly, LZWL [108] is a syllable based compression scheme. Lempel-Ziv-Jakobson (LZJ) removes entries occurring once when the dictionary is full.

3.3.5.1 LZW Example

LZW is a mature compression scheme with many implementations making it the most commonly used parsing LZ scheme for both of binary and text applications [105, 106].

If 8-bits data unit size is used, the dictionary indices would be reserved by all the combinations of the basic 256 characters. Therefore, a larger dictionary should be allocated from the beginning to accommodate the sequences of data units that would be added later, thus at least 9-bit indices should be used. This dictionary will be appended whenever a new (word) sequence of data units is encountered during compression. For example, consider a text document that has the repeated word "hello". It will be appended to the dictionary the first time it appears in the data stream, and will be used in the output compressed data stream as it is. The "hello" entry is 5 bytes (40 bits) plus some additional dictionary bookkeeping overhead. The next times the word appears in the data stream it will be represented by the index (representing code) of its entry in the dictionary in the example that would be 9 bits. The compression degree of those two occurrences is 80/(40+9), i.e., 39% saving for this word only. The overhead added to other symbols that were not compressed should be outweighed by the achieved compression for this compressed word and others. In case of no repeated sequences at all, the algorithm will add 12.5% to the original size instead of compression.

LZW compression doesn't use much computations or even sorting for compression, only simple search. In other words, LZW provides long sequences to be stored in the dictionary quickly. Also the decoder can restore the dictionary quickly from the encoded data upon receiving. Since the dictionary needn't be transferred, the compression degree will not suffer from the overhead of adding it to the data stream. At the same time the dictionary theoretically can infinitely grow to obtain higher compression ratio without any synchronization requirements between encoder and decoder or such overhead. Still, searching such big unsorted dictionary would consume quite a long time and considered the only or main factor of compression time. The processing time is directly proportional to the size of the dictionary, since the unsorted dictionary sequences must be searched with brute force techniques. Thus processing time is inversely proportional to compression degree, which requires larger dictionary size as mentioned earlier. Most implementations try to achieve some satisfactory trade-off point. Being of temporary function in compression, the dictionary could be restricted to some size by sacrificing the achievable compression degree.

Many of the stored sequences might not be used much for later compressing, sometimes never used. The dictionary might get to be bigger than the source data size. Due to practical memory limitations and also size effect on speed limitation, implementations place some restrictions on the dictionary size. When the dictionary gets bigger than specified thresholds, it will be deleted [103]. Then another new block of compression is started with a fresh dictionary. Different signalling has been used to ensure that the decoder resets the compression and dictionary at the same stream position. This could reduce compression time by sacrificing compression degree.

At some point the instantaneous dictionary size when added to the remaining input size can be extremely huge. For such cases, a new compression scheme for lightweight temporary memory is targeted. To achieve this goal, in chapter 6, LDC is proposed as a new compression algorithm with finite size dictionary. The memory reduction is further maximized in the decoder side for slow and small memory terminals.

3.3.6 Compression Schemes for Communication in Transport Layer

Source data (packets) to be compressed is considered mixed structured data consisting of; header (signalling information) and user information which also called as data or payload, except for MSR (chapter 5) where packets are considered as homogeneous unstructured data. The compression data targeted can be categorized into three; header (homogeneous), data (homogeneous) and both (mixed). Header compression must be lossless, due to the importance of its contents for control and signalling. For example, the Internet protocol (IP) header consists of information for routing the data to its destination. If some information is lost or changed, the packets will fail to reach its destination. In data compression, the information is compressed according on the user requirements. For instance, in the case of lossy compression, the images quality is reduced by permanently eliminating certain information. To compress both information fields, lossless schemes are commonly used. Some examples of header compression technique used in networks are described below. MPLS and ATM are among the state of art network protocols with compression scheme concepts as its core design philosophy.

Van Jacobson's Header Compression (VJHC) is the first internet compression scheme that compresses the TCP/IP header in low-speed serial links [109]. It reduces the normal 40 byte TCP/IP packet headers to 3-4 bytes for the average case by sending the differences in the header fields instead. By this way, VJHC can get nearly 50% of compression of the header.

The IP Header Compression (IPHC) extends VJHC. It is commonly used for packets over Transport Control Protocol/ Internet Protocol (TCP/IP) and User Datagram Protocol/Internet Protocol (UDP/IP) in low speed links, for TCP streams, IPHC is identical to VJHC [110]. Since UDP streams are connectionless, IPHC introduces the concept of context with some unique CID for each data stream. This context is used to save the data stream header fields that are static or have little change among the continuous packets shared between both the encoder and decoder.

Compression Real-time Transport Protocol (CRTP) was developed to compress streaming multimedia data packets of the Real Time Protocol (RTP). However, CRTP also can compress UDP and IP headers, the 40 bytes of RTP/UDP/IP packet headers can be compressed to only 4 bytes [111]. For multimedia data quality is reduced by lossy compression of the data field of the packets. CRTP wastes a lot of bandwidth when for synchronizing between the encoder and decoder. CRTP can perform well on the small round trip time (RTT) link. In long RTT, the encoder and decoder cannot achieve a good synchronization, which lead to a series of packet loss.

RObust Header Compression (ROHC) is a standardized method to compress the UDP, UDP-Lite, RTP and TCP header of Internet packets. ROHC uses the IPHC concept of context and manages the context identifier (CIDs) reasonably and effectively [112].

Adaptive Compression-based Technique (ACT) for congestion control uses both lossy ADPCM (adaptive pule code modulation) and lossless RLC (run-length coding) compression. Discrete wavelet transform is also utilized to categorize data priorities and assign each a different frequency to achieve fairness in wireless sensor networks.

Real time adaptive packet compression for high latency networks with limited bandwidth uses the generic zlib library (LZ based) to improve the drop rate of heavy load satellite networks during heavy congestions [125].

3.3.7 Classification of the Examples

The following table shows the result of applying the categorization presented earlier on the examples listed in this section.

Table 3.1: Classification of the compression schemes presented

Scheme	Homogeneity	Purpose	Accuracy	Structuring of data	Repetition distance	structure sharing	No. of passes	Sampling frequency	Sample size ratio
JPEG	Homogeneous + Mixed	Storage & Comm.	Lossy or Lossless	In 2 stages	8x8 ~ 64x64 blocks	Optional (Palettes)	3	Blocking	100% of block
RLE	Homogeneous	Storage & Comm.	Lossless and lossy	None	Practical limit	Optional (Palettes)	1	Max. Run-length	100% of block
Huffman	Homogeneous	Storage & Comm.	Lossless	None	Whole Stream	Coding Tree	2	Whole Stream	Small Percent
Arithmetic	Homogeneous	Storage & Comm.	Lossless	None	Whole Stream	Yes	2	Whole Stream	Small Percent
LZ	Homogeneous	Storage & Comm.	Lossless (lossy)	None	Practical Limit	Mostly No	Mostly 1 (2 or 3)	Practical Limit	100% of block
VJHC	Mixed	Comm.	Lossless	Utilized	Practical Limit	No	1	N/A	N/A
IPHC	Mixed	Comm.	Lossless	Utilized	Practical Limit	No	1	N/A	N/A
CRTP	Mixed	Comm.	Lossless + lossy	Utilized	Practical Limit	No	1	N/A	N/A
ROHC	Mixed	Comm.	Lossless	Utilized	Practical Limit	No	1	N/A	N/A
L. S. Tan et al.	Homogeneous	Comm.	Lossless	None	Practical Limit	No	1	Practical Limit	100% of block
ACT	Mixed	Comm.	Lossless + lossy	Utilized	Block	No	2	Block	100% of block

3.4 Compression Degree (Ratio)

The performance of the data compression schemes depends on the nature of the source data units to be compressed as well as its structure. In order to compare different data compression schemes fairly, the compression degree which is also known as a compression ratio (CR) is commonly used. In this research, the compression ratio is defined as the ratio of the size of the original input source data units (S_o) to the total output. The first form of equation 3.1 represents compression schemes where a centralized dictionary like structures are not attached to the output, while the second form is used for compression schemes that use such attached dictionary. The total output in latter case, is the summation of the dictionary structures size and the size of the compressed representing coded data (S_c). More detailed example of calculating the CR will be shown in Chapter 6, together with the newly proposed compression scheme LDC.

$$CR = \frac{S_o}{S_c} \tag{3.1}$$

$$CR = \frac{S_o}{S_c + D_s} \tag{3.2}$$

The amount of compression is stated as $CR:1$, which could be expressed as when a compression scheme takes in the original source data units size (S_o) and compress it, the output compressed representing code size is (S_c). According to this CR definition, a higher CR indicates smaller compressed representing code size. This reciprocal of CR is also used in some literature with the same name 'compression ratio'. In this research the reciprocal will be referred to as compression percentage to avoid the confusion $CP = 1/CR$, it will be used to represent utilization savings which is discussed in the last section of this chapter.

3.4.1 Overall Compression Ratio for Part-wise Compression

When compressing homogeneous data, the CR and CP mentioned are sufficient to evaluate the compression degree and utilization savings. As for mixed data, the sources are usually split into different parts before compression, to obtain higher correlation within each separate part sufficient for compression. If the input source was originally formed by mixing different sources it is simply split back, otherwise some blocking mechanism is used with or without enough knowledge about the nature of the mixed data. Blocking is also used for large homogeneous data sources that cannot be handled as one part due practical limitations.

When blocking is used, different parts of the source are compressed separately what will be referred to as part-wise compression in this thesis. Part-wise compression yields different compression degrees for each part of the source. To calculate the overall compression degree of the whole source, overall compression ratio and percentage concept are introduced in this subsection.

Overall compression percentage CP_{ov}, shown in equation (3.3), is the weighted sum of all the compression percentages scaled probabilistically by the fraction Pc of data parts being compressed, respectively. Since different parts of data get compressed with different compression percentages (compression ratio), the overall compression percentage obtained here represent another virtual homogeneous input data size that was totally compressed using that uniform compression percentage. The output size after compression of this virtual data is still the same as the summation of the sizes of all the compressed data parts, while the size of the virtual input data is different from the summation of the sizes of the data input parts.

$$CP_{ov} = \sum_i Pc_i \times CP_i \qquad : \sum_i Pc_i = 1 \tag{3.3}$$

$$CP_{ov} = Pc \times CP + (1 - Pc) \tag{3.4}$$
$$CP_{ov} = \frac{Pc}{CR} + (1 - Pc)$$

In this research, data is considered to be divided simply into only two parts. First the homogeneous compressible part whose Pc_1 will be referred to as only Pc for short. The compression percentage of the first part will also be referred to as only CP for short. The other part, totally incompressible whose fraction would be $1 - Pc$, will be having zero compression, which also means unity compression ratio. This special case is represented in both forms of equation (3.4). The two forms of overall compression ratio, shown in equation (3.6), are simply the reciprocal of the overall compression percentage derived as shown from equation (3.5).

$$CR_{ov} = \frac{1}{CP_{ov}} \tag{3.5}$$

$$CR_{ov} = \frac{1}{\frac{Pc}{CR} + (1 - Pc)}$$

$$CR_{ov} = \frac{CR}{Pc + CR - Pc \times CR} \tag{3.6}$$

$$CR_{ov} = \frac{1}{Pc \times CP + (1 - Pc)}$$

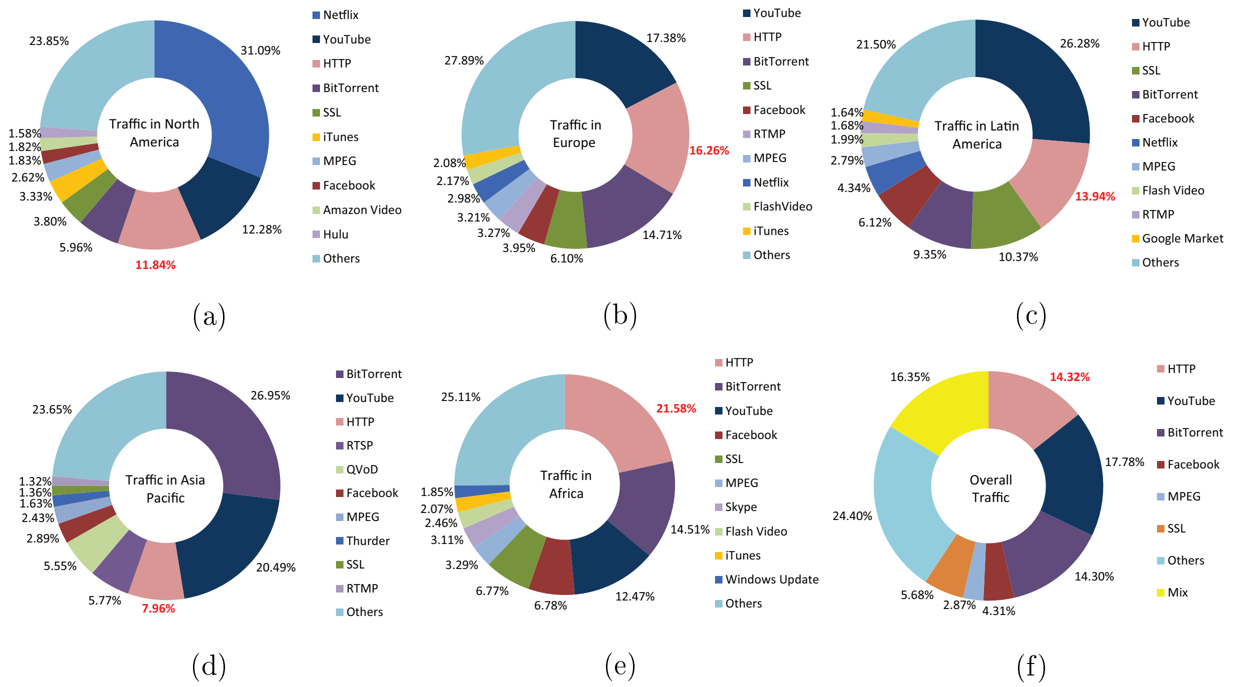


Figure 3.3: Top servers traffic during the studied sample period

The Pc in this research is estimated according to the report of 2014 global Internet Phenomena, which is provided by Sandvine Intelligent Broadband Networks [120]. Fig. 4.13 (a) to (e) showing the traffic percentages during a randomly selected sample period for studying the top 10 servers accessed. The top servers include HTTP, YouTube, Bit Torrent, Facebook, MPEG, Secure Socket Layer (SSL). The overall average of the first 5 parts of the figure is shown in part (f).

According to the overall average traffic percentages in Fig. 4.13, most of the traffic from the top servers is incompressible, as multimedia from server like YouTube. That traffic is tightly compressed in the original format and almost no chance at all for any further compression. In this section, it assumes that HTTP and some other serversâĂŹ traffic are compressible. Yielding an upper bound of 26% of compressible data traffic. In worst case, only data traffic of HTTP servers is compressible, that means Pc is approximated to 14% as a lower bound. Fig. 3.4 shows the CR_{ov} versus CR of the compressible traffic with different Pc values. To determine the utilization savings the overall compression percentage is shown in Fig. 3.5 under different Pc values.

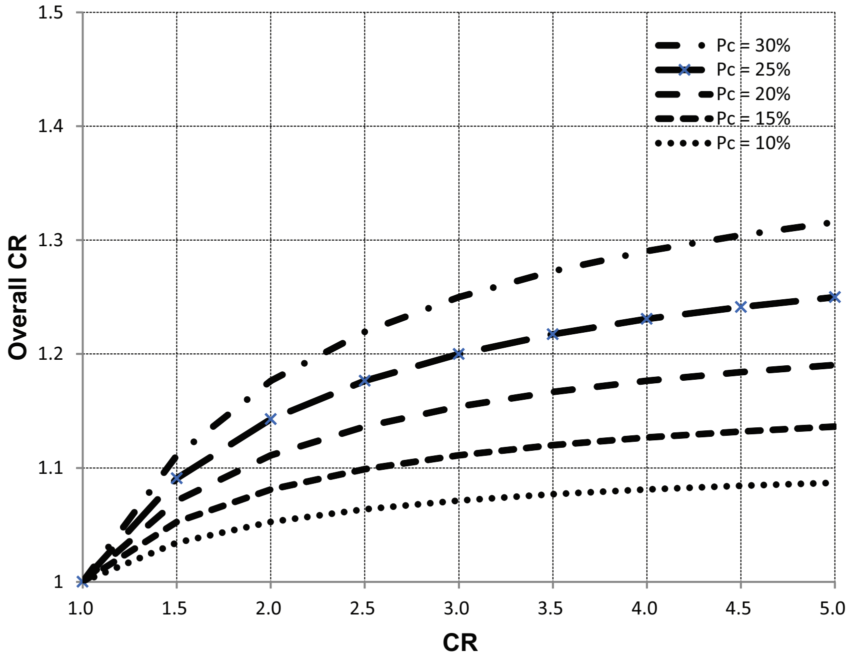


Figure 3.4: CR_{ov} versus CR with different Pc

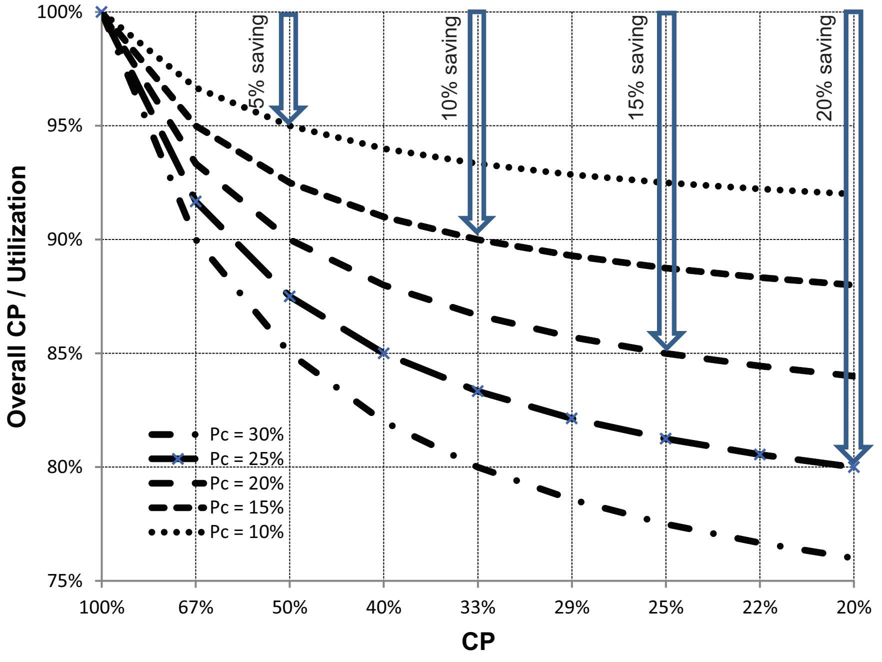


Figure 3.5: Utilization (CP_{ov}) and saving versus CP with different Pc

For reference purposes, Pc range from 10% to 30% will be used to plot the relations between CR, CP from one side and overall overall CR, CP (utilization) and resources saving, respectively. Figures 3.4 and 3.5 show that the overall CR and overall CP are both much more affected by the changes in Pc values, than any change in CP. In Fig. 3.5 shows the case when CP is 20% in the far right of the graph. At that time, the compressible part of data is reduced by 80% resulting only 20% overall resources savings when Pc is nearly maximum at 25%. In other words, aggressive compression when "$CR = 5$" is still restricted by "$Pc = 25\%$" resulting $CR_{ov} = 1.25$.

Accordingly from now on, this thesis will only use the range (1 : 1.3) for CR referring to the overall compression, which refers to a corresponding utilization/compression percentage range (100% : 75%), respectively. The respective resources savings range will be (0% : 25%). Compression percentages, utilization and resources saving will not be mention directly in the next chapter, when required reference will be made to use the formulas in this chapter to calculate the corresponding values from CR.

3.5 Summary

The different compression schemes discussed in this chapter show the general tendency of most compression designers to target homogeneous data or structured data. Most well designed lossy schemes can offer much better compression than lossless depending on the acceptable distortion degree. The nature of data especially in mixed non-homogeneous data is more important than the compression scheme used, even if the compression scheme is extremely aggressive. All compression schemes choose only one of the conflicting targets to try to approach as much as possible, compression speed (time), compression degree (ratio), compress resources (temporary memory and energy) or compression complexity. Most storage targeting compression schemes mostly care about the compress degree rather than any other compression performance. While communication targeting compression schemes are usually more concerned with the energy resource efficient utilization within some time restrictions.

In this thesis, communication targeting compression is addressed trying to improve the efficiency of resources utilization. Both chapters 5 and 6 will introduce new compression support in networking frameworks to improve the efficiency of resources utilization.

Chapter 4

Empirical Model of Congested Network with Compression

This chapter considers the empirical model of network throughput in a congested network. First, the simple TCP throughput model per connection is studied. Then, the model of throughput with compression is derived. The effect of buffer size, incoming rate, compression ratio, and compression time on the throughput and round trip time is analysed.

4.1 TCP Throughput Models

Network throughput is one of the metrics that are commonly used to evaluate the performance of networks. The network throughput measures the maximum number of data packets that is transmitted in a unit time. In this chapter, network throughput is studied to determine the effect of compression in edge/core devices. In order to study compression in networking and evaluating its effect on network performance, one of three common approaches is usually utilized. The first one is using a network simulator and observing the network behaviour, regression statistical models can be used to express the obtained results. The second is to use logs from congested network either real life or experiment test beds, then following the similar regression models such as curve fitting. The problem of those two approaches, enormous time and data is required to get a sample that is sufficiently representable. Any approximation to simplify or speed up would probably yield in adequate model. Besides, a lot of researches have utilized those two approaches to study compression but only statistically or for special networking cases without obtaining much conclusive results. The last approach is to formulate a new empirical model for compression in networks, by deriving an equation from the basic throughput model of TCP. In this research, the last approach is used.

4.2 Basic TCP Throughput

A simple empirical model for the throughput of TCP was presented [113]. This throughput is based on packet loss rate, p and the average round trip time, RTT.

A round is a period of time that starts with the back-to-back transmission of W packets, where W is the current size of the TCP control window. In the basic TCP model, loss indications are exclusively of type "triple-duplicate" ACK (TD). TD period (TDP) is defined as a period of time from a specific TD loss indication and the immediately preceding one, thus including one or more rounds. The similar term round trip time will be exactly equal to a round duration given the assumptions below.

TCP's window size (W) increases each time an ACK is received. Since the processing and transmission times in the edge or core devices are assumed to be zero, the window size is only increased at the end of every round. In other words, all the W packets of any specific round are all transmitted at exactly the same time instant, the beginning of that round. If ACKs of those packets are received without any loss, all ACKs are also going to arrive at the exactly the same time instant, the end of round before starting the new round within the same TD period. If TD is detected, the current TD period ends by this round and the next round marks the beginning of the next TD period as well.

The equations in this section use the variables listed below, those variables are per connection. When needed, individual variables will be further explained. In some literature, a window, round trip time are used interchangeably, in this thesis, I am going to stick to the definitions above.

α	The number of packets from the beginning of the TD period until and including the first lost packet (unit: packets)
β	The number of sent packets in the last round in that specific TD period (unit: packets)
x	The number of rounds in a TD period until and including the round where the TD occurs. It is either the number of rounds in that specific TD period or the same number minus one (unit: packets)
A	The time duration of the TD period (unit: seconds)
b	The slope of additive increase in a TCP congestion control scheme
p	Loss probability of sent packets
RTT	Round-trip time of one packet start when the packet of TCP connection is completely transmitted from the sender until its ACK from the receiver is completely received by the sender again
r	The duration of the round trip time of this specific round in the TD period considered (unit: seconds)
T	Number of packets sent per unit time regardless the eventual fate, either lost or received (unit: packets per second)
R(p)	Relationship between the throughput of TCP connection (R) and loss probability (p)
W	The maximum window size of that specific TD period, which is the window size of the last or before last round, depending on where did the TD occurred
Y	The number of sent packets in that specific TD period

The assumptions used in [113] model are:

1. TCP sender is in the mode of saturation with infinite data.

2. All network devices have infinite buffering resources.

3. TCP's window size is increased infinitely each time an ACK is received.

4. Duration of a round is equal to the round trip time and is assumed to be independent of the window size.

5. All the packets of any specific round are all transmitted almost at exactly the same time instant, the beginning of that round.

6. All ACKs are also going to arrive almost at the exact same time instant, the end of round before starting the new round within the same TD period.

7. Time for sending all the packets in a window is very small tending to zero which is definitely much smaller than the RTT.

8. Packets in a round are independent from any packets in other rounds.

9. If a packet is lost, all remaining packets transmitted until the end of that round are also lost.

10. The last round of a TD period is the one right after its TD round, i.e., the first packet in a round is not the one lost.

11. Packets are not lost because of timeout at all.

Fig. 4.1 shows the basic idea of the TCP model used to derive the throughput expression. Other symbols in the above table and not mentioned here, are to represent related variables that are defined straight forward. As soon as TD is detected either for the first packet of the current round or other packets in the previous round, the TD period will terminate immediately. All the packets following the TD packet in same round are assumed missing as well even they were acknowledged. The round in which the TD occurred is referred to as penultimate round or TD round (the x round). As indicated earlier it can be either the last round in the TD period or the one right before.

A new TD period starts immediately with its own set of rounds. The window size of the first round in the new TD period is set to half the value of the window size of the TD round in the previous TD period. All the packets in that round are transmitted almost at the same instant with inter-packet delay tending to zero according to assumption 4–7. After r period of time for this round, the sender starts receiving ACKs for the packets sent in the beginning of the round. When the first ACK(s) arrives, the next round of the TD period immediately starts without waiting for ACKs of the other packets from its own round. The sender sends one new packet for each ACK received. Since inter-packet is assumed tending to zero, both the ACKs and transmission of the packets of the new round all occur almost at the same instant. When a new round is initiated, the window size is incremented (additively according to linear slope $1/b$). In Fig. 4.1, the window is increased by $1/2$ every round, i.e., it is incremented by one packet every two rounds. The current TD period will continue until the occurrence of another TD (assumptions 2 and 3).

When TD is detected for some packet (the first "X" marked packet from below) in the current round, the round is immediately terminated as in the beginning of this TD period. The TD packet could possibly be one of the packets in the previous round or this round itself. The maximum window size (W) refers to the window size of the TD round

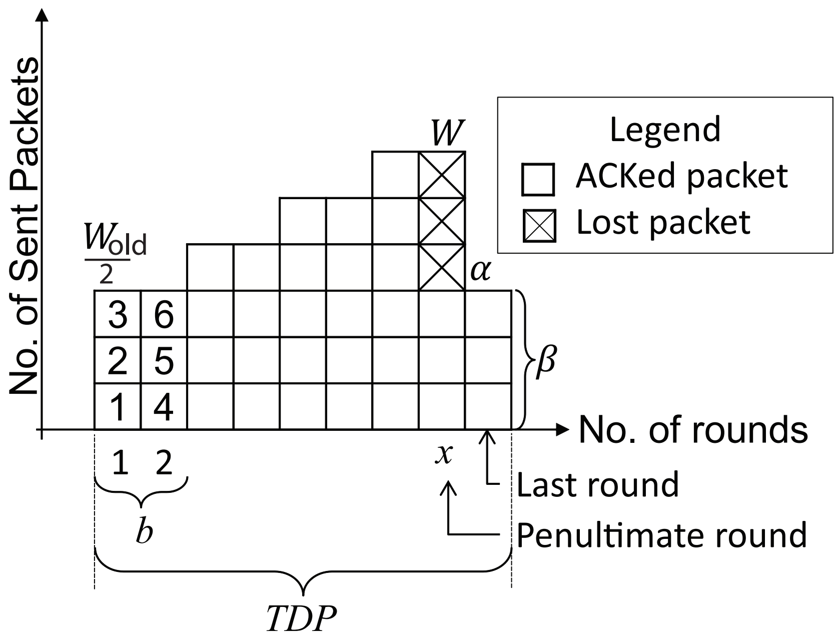


Figure 4.1: TD period of a TCP connection

(the x round), regardless it is the last round or the one before it. Half that of the W in the current TD period will also be used as the initial window size of the first round of the next TD period. The last round of the TD period is most probably the round right after the x round (the TD round) and according to assumption 10 will always be in the before last round. The number of packets successfully acknowledged in the TD round (clear packets in the lower part of the TD round) will immediately cause same number of new packets (β) to be transmitted in the last round, before the TD is detected and both the TD round and period are terminated. The higher "X" marked packets in the TD round are all assumed missing even that their ACKs were received successfully (assumption 9).

The probabilistic expectation function of the throughput $R(p)$ is the probability-weighted average of all possible throughput values [114,115]. According to the probability law of large numbers, for any infinitely repeated random experiment, the arithmetic mean converge to the expected value, only if the experiment repetition go to infinity. Thus, the average throughput of a TCP connection would converge to the expectation of the random probability throughput variable representing it. The model targeted by the basic throughput of a TCP connection is operating only in steady state with infinite input data per connection (assumption 1). Therefore, infinity is guaranteed and the average throughput T converges to the expectation of $R(p)$ as in equation (4.1). From now on in this chapter, the probability random variables will be used to refer to the expectation of those variables, unless stated differently.

Thus in equation (4.1), the throughput is further expanded as the number of packets transmitted divided by the time duration of a TD period.

$$R(p) = E[T] = \frac{E[Y]}{E[A]} \tag{4.1}$$

The derivation starts with the evaluation of the number of packets sent, Y before A. Y in a specific TD period is obtained by observing the last two rounds of that TD period. The number of packets transmitted (Y) is equal to the number of packets in the TD

period until the first lost packet (α) added to the number of lost packets. Since the model assumes all packets after a TD are lost even if acknowledged later (assumption 9), then the number of lost packets in that TD period is equal to the maximum window size of that specific TD period (W). The first lost packet is already counted in both definitions, thus the summation is adjusted by deducting one.

$$E[Y] = E[\alpha] + E[W] - 1 \tag{4.2}$$

According to assumption 8, the probability random processes of the number of packets in the TD period until the first lost packet (α) is obviously a sequence of the well-known independent and identically distributed (i.i.d) random variables. As shown in the equation below, the probability of α being equal to some k value, is represented by the probability that the number of packets acknowledged until the TD equals $k-1$.

$$P[\alpha = k] = (1-p)^{k-1}p, \qquad k = 1, 2, ... \tag{4.3}$$

and the expectation of α can be calculated by equation below

$$E[\alpha] = \sum_{k-1}^{\infty}(1-p)^{k-1}pk = \frac{1}{p} \tag{4.4}$$

By using equation (4.4) to substitute for α into equation (4.2), Y is expressed in the equation below

$$E[Y] = \frac{(1-p)}{p} + E[W] \tag{4.5}$$

Next, the duration of TD period (A), will be investigated before continuing the evaluation of the Y expression. The TD period consists of x rounds, each of them lasting for r time. According to the assumption (4–7), a round will be identical to RTT. Thus r is the probability random variable representing RTT, which is independent of the round or its size, i.e., the number of packets sent in the round. The "A" equation (4.6) can be expanded by using joint probability rules into equation (4.7). According to the previously used probability law of large numbers, the average value of RTT will tend to the expectation of r value as the number of rounds tends to infinity.

$$A = \sum_{j=1}^{x+1} r_j \tag{4.6}$$

$$E[A] = (E[x] + 1)E[r] \tag{4.7}$$

So far, the expression of relationship between the throughput of TCP connection (B) and loss probability has been expanded by substituting from both equations (4.2) and (4.7) to reach the form below.

$$R(p) = \frac{\frac{(1-p)}{p} + E[W]}{(E[x]+1)E[r]} \tag{4.8}$$

Next, the derivation of the number of rounds (x) in a specific TD period is based on the change in window size. The linear relation between window size per round and x is expressed by equation (4.9). W is the maximum window size of the current TD period, and W_{old} is the maximum window size in the previous TD period before TD, which was halved as an initialization in the beginning of the current TD period. The slope of line segment of the linear relation is defined as $1/b$.

$$W = \frac{x}{b} + \frac{W_{old}}{2} \tag{4.9}$$

Equation (4.5) of the number packets in the current TD period (Y) will be used to find an expression for the maximum window size (W) in the next derivations. Y is the sum of all window sizes in all the rounds of the current TD period, it can be calculated using either the summation of an arithmetic sequence or the area trapped under the curve (line). The second method is shown in equation (4.10), the area is split into three regions. First the region is the area of rectangle below in Fig. 4.1, followed by the triangle above and finally the packets in last round (β) are added separately.

$$\begin{aligned} Y &= x\frac{W_{old}}{2} + \frac{1}{2}x\left(W - \frac{W_{old}}{2} - 1\right) + \beta \\ &= \frac{x}{2}\left(W + \frac{W_{old}}{2} - 1\right) + \beta \end{aligned} \tag{4.10}$$

By calculating the stationary distribution of the probability random variable (W) as a Markov process based on equations (4.3, 4.9 and 4.10), the details are omitted for simplicity. The probability density of the number of rounds in the TD period (x) is approximated by assuming that both x and W are mutually independent sequences of i.i.d random variables. The following equations (4.11 and 4.12) were deduced using the calculated and mentioned assumption together with equations (4.5, 4.9 and 4.10), the steps are also omitted here.

$$E[W] = \frac{2}{b}E[x] \tag{4.11}$$

$$\frac{1-p}{p} + E[W] = \frac{E[x]}{2}\left(\frac{E[W]}{2} + E[W] - 1\right) + E[\beta] \tag{4.12}$$

Since the number of packets sent in the last TD period (β) is uniformly distributed from 1 to W, thus $E[\beta] = E[W]/2$. Using the expectation of β together with equations (4.11 and 4.12), the expression below in equation (4.13) for W is deduced.

$$\begin{aligned} E[W] &= \frac{2+b}{3b} + \sqrt{\frac{8(1-p)}{3bp} + \left(\frac{2+b}{3b}\right)^2} \\ &= \sqrt{\frac{8}{3bp}} + o(\frac{1}{\sqrt{p}}) \end{aligned} \tag{4.13}$$

For small values of p, the expectation of the maximum window size is approximately $E[W] \approx \sqrt{\frac{8}{3bp}}$. When substituting for the approximated expectation of maximum window size (W) in equations (4.7 and 4.11), both the expectation of the number of rounds (x) and the expectation of the TD period duration (A) are obtained as in equations (4.14

and 4.15). In equation (4.15) the expectation of the round trip time is replaced by RTT as explained earlier according to the probability law of large numbers.

$$\begin{aligned} E[x] &= \frac{2+b}{6} + \sqrt{\frac{2b(1-p)}{3p} + \left(\frac{2+b}{6}\right)^2} \\ &= \sqrt{\frac{2b}{3p}} + o(\frac{1}{\sqrt{p}}) \end{aligned} \tag{4.14}$$

$$E[A] = RTT \left(\frac{2+b}{6} + \sqrt{\frac{2b(1-p)}{3p} + (\frac{2+b}{6})^2 + 1} \right) \tag{4.15}$$

By substituting from both equations (4.14 and 4.15) into the throughput/packet loss relation equation (4.8), the final form of that relation is obtained in the equations below.

$$\begin{aligned} R(p) &= \frac{\frac{1-p}{p} + \frac{2+b}{3b} + \sqrt{\frac{8(1-p)}{3bp} + \left(\frac{2+b}{3b}\right)^2}}{RTT\left(\frac{2+b}{6} + \sqrt{\frac{2b(1-p)}{3p} + \left(\frac{2+b}{6}\right)^2 + 1} \right)} \\ &= \frac{1}{RTT}\sqrt{\frac{3}{2bp}} + o\frac{1}{\sqrt{p}} \end{aligned} \tag{4.16}$$

For p tending to zero, the throughput/packet loss relation is approximated as in the final equation below. In most literature, the final form is commonly referred to as simple throughput of a TCP connection (T).

$$R(p) = \frac{1}{RTT}\sqrt{\frac{3}{2bp}} \tag{4.17}$$

4.3 TCP Throughput with Finite Buffer

The basic TCP throughput was modified to limit the unrealistic assumption of infinite buffering all over the network, by limiting the buffering resources of one of network devices to a finite size B [116]. The additional assumption introduces by [116] handles that RTT is only considering propagation delay in the basic TCP model with infinite buffering. When introducing the limited buffering in only one network device, the queuing delay of that device is added to the RTT. The other network devices with infinite buffering stay with absolute zero queuing delay.

First the packet loss rate (p) is approximated using the state probability of the M/M/1 queuing theory. The long derivation details were omitted in the reference and are also not shown here. In the equation below, λ is the average input rate of that TCP connection and μ is the service (processing) rate of that network device. The service rate can be the routing rate in case of routers or the segmentation/encapsulation in case of end devices.

$$p \approx \frac{\lambda + \lambda B - B\mu}{\mu} \tag{4.18}$$

Then the round trip time (RTT) is also approximated by applying Little's theorem to the average queue length of the M/M/1 model again. The lengthy derivations were similarly omitted from both the reference and here. The equation below shows explicitly the overall average propagation delay of that TCP connection (τ).

$$RTT = \frac{1}{\mu - \lambda} + \tau \tag{4.19}$$

After simplification of the substitution of both p and RTT in the basic TCP throughput equation (4.17, the final throughput obtained in [116] is shown in the equation below.

$$T = \frac{(\mu - \lambda)\sqrt{\frac{3\mu}{2b(\lambda+\lambda B-B\mu)}}}{1 - \lambda\tau + \mu\tau} \tag{4.20}$$

4.4 TCP Throughput with Finite Compressed Buffer

The TCP model with finite buffering introduced in the previous section is extended in this section to represent the compressibility. The basic finite buffering model considers restricted buffer size in one of the network devices, that device is further enhanced in this section by adding the ability to compress a specific part of the data stored in that finite buffer. This section assumes that the network is heavily congested, thus compression is only performed in such network conditions. This model also assumes that compression time (t_c) is only in the compressing limited buffer network device, it is independent of the other factors of RTT. Thus t_c is only an additional constant offset to the original RTT expression obtained in equation (4.19).

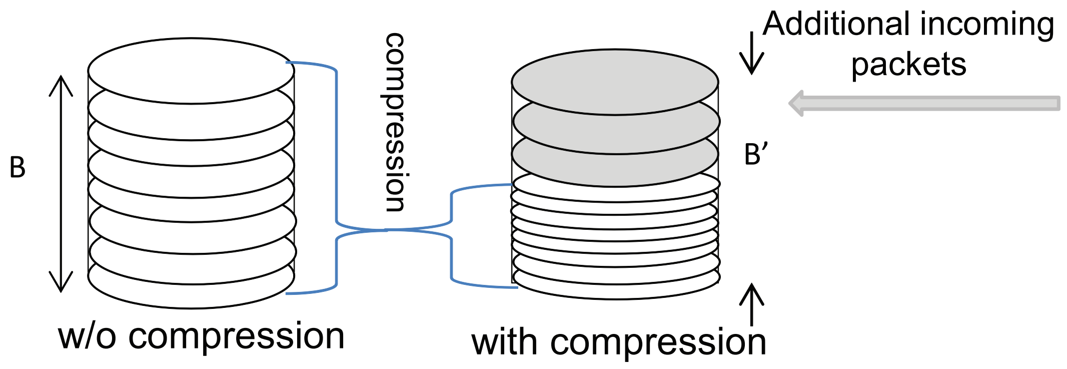


Figure 4.2: Buffer without and with compression function

When the data packets in a buffer is compressed, the total amount of data is decreased, thus the available buffer size increases. In this case, more data packets can be received and stored in the buffer, as shown in Fig. 4.2. When the compression is performed, the number of packets that can be stored in the limited buffer will increase according to the compression degree achieved. Some of the packets in the buffer will be in compressed while others will stay in the normal uncompressed form. The final number of packets in the buffer B', regardless compressed or not, can be approximated by the product of the overall compression ratio (CR) and the real physical buffer size, equation (4.21).

$$B' \approx CR \times B \tag{4.21}$$

RTT from equation (4.19) is rewritten to include the compression time (t_c) to the original expression as in the equation below.

$$RTT = \frac{1}{\mu - \lambda} + \tau + t_c \tag{4.22}$$

By using equation (4.21 and 4.22) to substitute into equation (4.20), the obtained throughput (T) expression is shown as in the equation below.

$$T = \frac{\mu - \lambda}{1 + (\tau + t_c)(\mu - \lambda)} \sqrt{\frac{3\mu}{2b\left[\lambda + B \times CR\,(\lambda - \mu)\right]}} \tag{4.23}$$

4.4.1 Curves of the Empirical Model

The constant and ranges of variables used in this section are listed below:

Constants	
Window slope, b	$= 2$
Average overall propagation delay, τ	$= 150$ ms
Service rate, μ	$= 150000$ packets/sec

Ranges of variables	
Input rate, λ	$= 149900 \sim 149990$ packets/sec
Buffer size, B	$= 500 \sim 900$ packets
Overall compression ratio, CR	$= 1.0 \sim 1.3$
Compression time, t_c	$= 0 \sim 600$ ms

The ranges were deduced during the derivation of the equation to eliminate the following conditions:

- Zero division ($\lambda > \mu$)
- Negative square root
- Negative probabilities
- Probabilities more than one
- CR less than 1 (the size of original data is enlarged)

Some of the combinations of those conditions were not seen straight forward from the equation, the ranges were adjusted during the plotting of the curves below. All of the values that are used in this analysis represent a very small fraction of actual values in real networks. For example, 1% of the transmission rate of 15 Gbps that is 150 Mbps.

In the next sub section, Fig. 4.3 through 4.10 show the *Throughput* and RTT curves plotted against either B and λ for different CR and t_c. The reference case of unity compression is included in the curves and labelled (CR=1) or no compression time (t_c=0). That includes the case of deciding not to compress (CR=1) after checking, which consumes some time for the checking (t_c>0). The checking time is considered as part of the compression time. The case of extremely fast compression is also considered with different compression ratios (CR>1) and no compression time (t_c=0).

4.4.2 CR and t_c effect on Throughput and RTT with Different Buffer Size

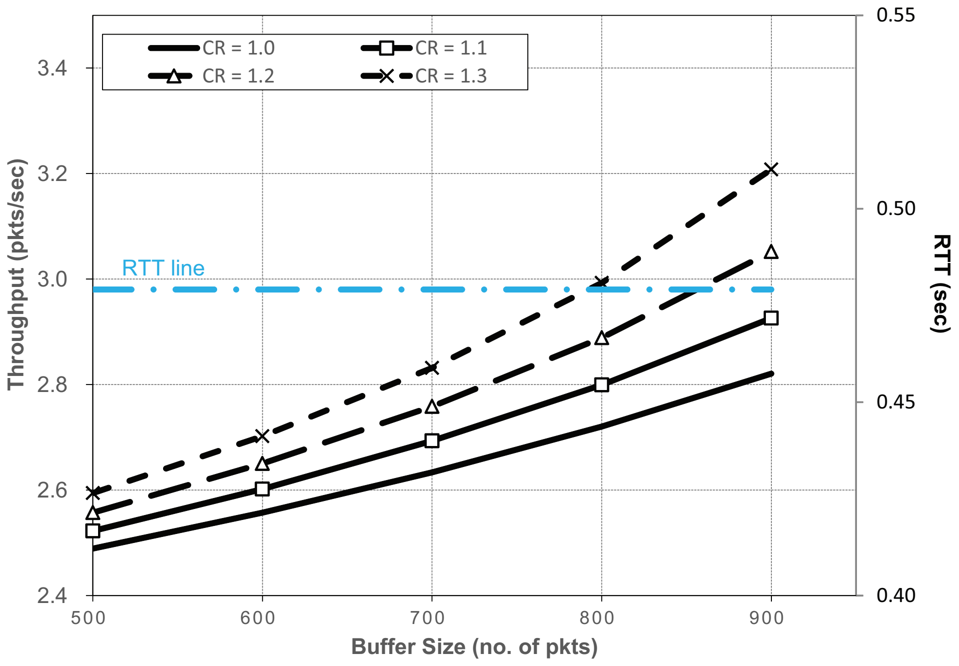


Figure 4.3: Throughput and RTT versus B under different CR

Fig. 4.3 through 4.5 show T and RTT curves plotted against B for different CR and t_c. From Fig. 4.3, it is clear that RTT is not affected at all by different CR or B. RTT was found totally constant almost at 479 ms. On the other hand, throughput clearly improves with small slope with both B and CR. The higher the CR, the better the improvement caused by the B. In other words, increasing B slightly improves throughput by 12% when not compressing and almost the same in lower CR. Increasing B will improve throughput until 23% when CR is maximum. Increasing CR will also improve throughput between 4% to 14% provided the same B. When combining the increase in both B and CR, throughput improves almost by 28%.

In Fig. 4.4, throughput without compression increases as the buffer size increases. This conforms to the traditional queuing theory basic assumption. The increase in throughput is almost linear with low slope. Throughput also slightly increases as the compression speed improves (compression time t_c decreases). When combining the effect of buffer size increase and compression time decrease, the obtained slope is slightly lower than that of increasing the buffer size alone. Once compression is used, the effect of longer compression time has a very small effect on improving the throughput.

Fig. 4.3 and 4.4 clearly confirm that throughput depends on B, CR and t_c. The effect of small changes in B (80%) clearly makes a considerable difference (23%) in throughput which increases when combined with CR. While 200% faster compression (smaller t_c) is required to get only 60% throughput improvement. Throughput improvements caused only by CR lies in between both factors. It can be seen that the maximum 13% CR increase can improve 4% to 14% throughput depending on B. In summary, slower more aggressive compression when added on top of the increasing B, enhances throughput.

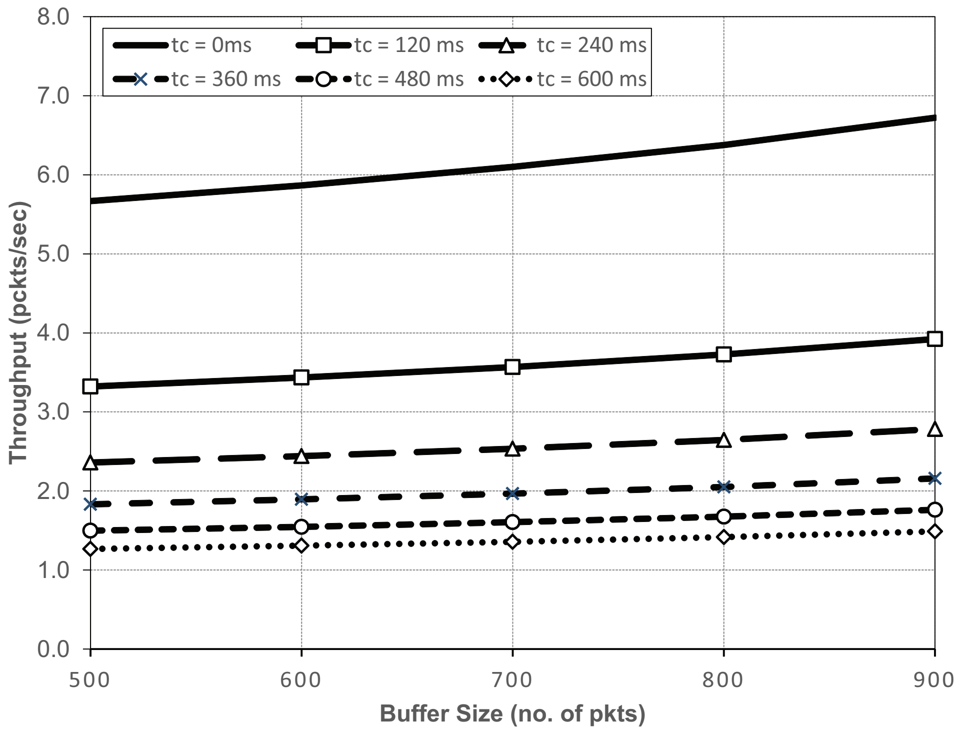


Figure 4.4: Throughput versus B under different t_c

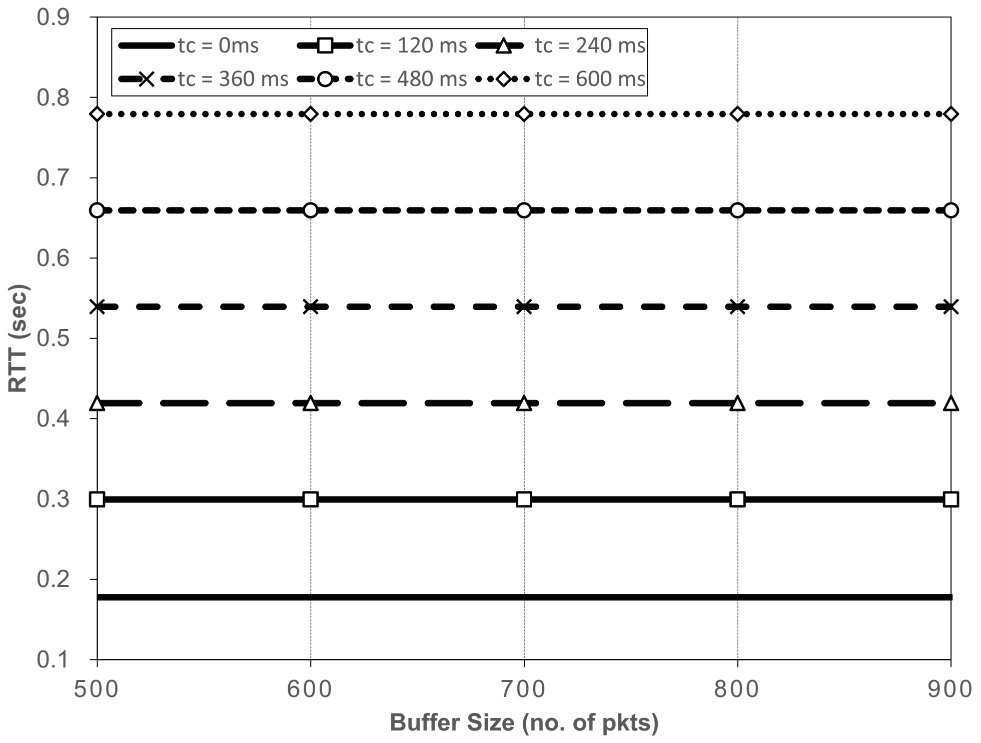


Figure 4.5: RTT versus B under different t_c

In Fig. 4.5, RTT is still independent from B with perfectly horizontal lines for different t_c. Still a different RTT value is visible for each t_c. Since RTT already includes t_c as a simple additional term, thus different t_c only shifts the horizontal RTT vertically. The shifts can be clearly seen to be of equal steps (same as the added t_c). It is clear from the curves that no other factors are affecting RTT. In other words, RTT is not affected at all by B or CR. RTT is only affected by t_c as a linear constant to offset the RTT lines.

4.4.3 CR and t_c effect on Throughput and RTT with Different Incoming Rate

When talking about increasing input rate, throughput is expected to start dropping since the additional packets will contend for the limited buffer space. Thus, the packet loss rate will increase.

In Fig. 4.6, 4.8 and 4.7, both throughout and RTT are plotted against λ for different CR and t_c. The longer RTT and lower throughout with increasing λ is confirmed by those figures. This is used to validate the correctness of the earlier derived model in this chapter. The result obtained from the previous B dependence figures is further affirmed here. With respect to increasing B, RTT was totally independent of CR, having all RTT exactly on top of each other for any CR. While in the λ dependence figures RTT curves are not constant, instead they increase with λ. (0.06%) additional incoming packets will increase the contention for the limited buffer size, then RTT rises (20%). The behaviour of the curves without compression or with any kind of compression is almost exactly the same. Throughput curves here are similar to the throughput curves with respect to increasing B. Almost 50% decline in throughput occurs when the λ increases, compression manages to slow down this decline. The gap between the RTT curves of different t_c is almost constant as in the case of RTT versus B.

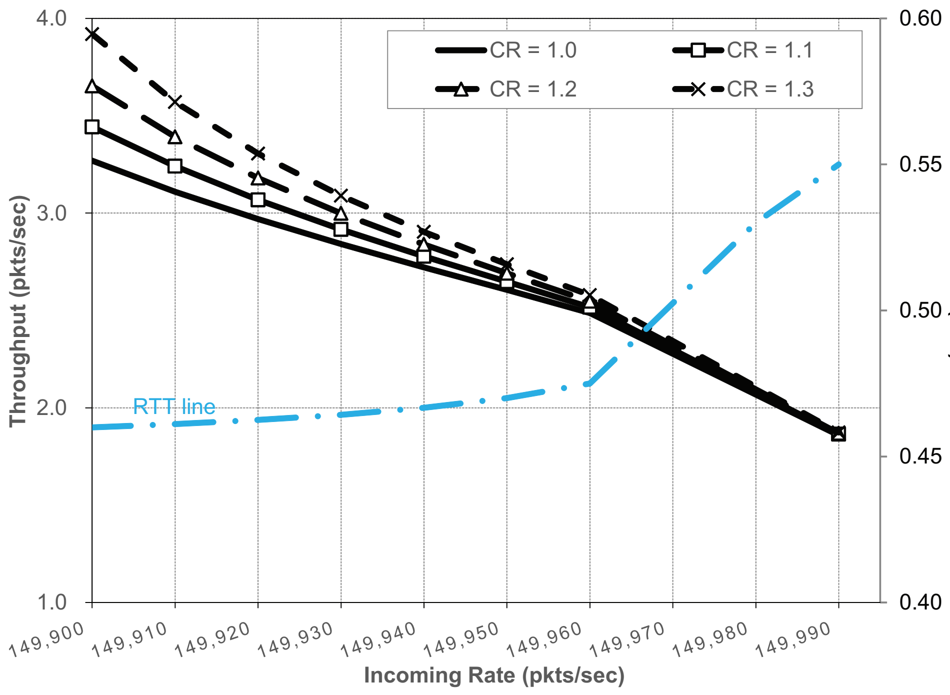


Figure 4.6: Throughput and RTT versus λ under different CR

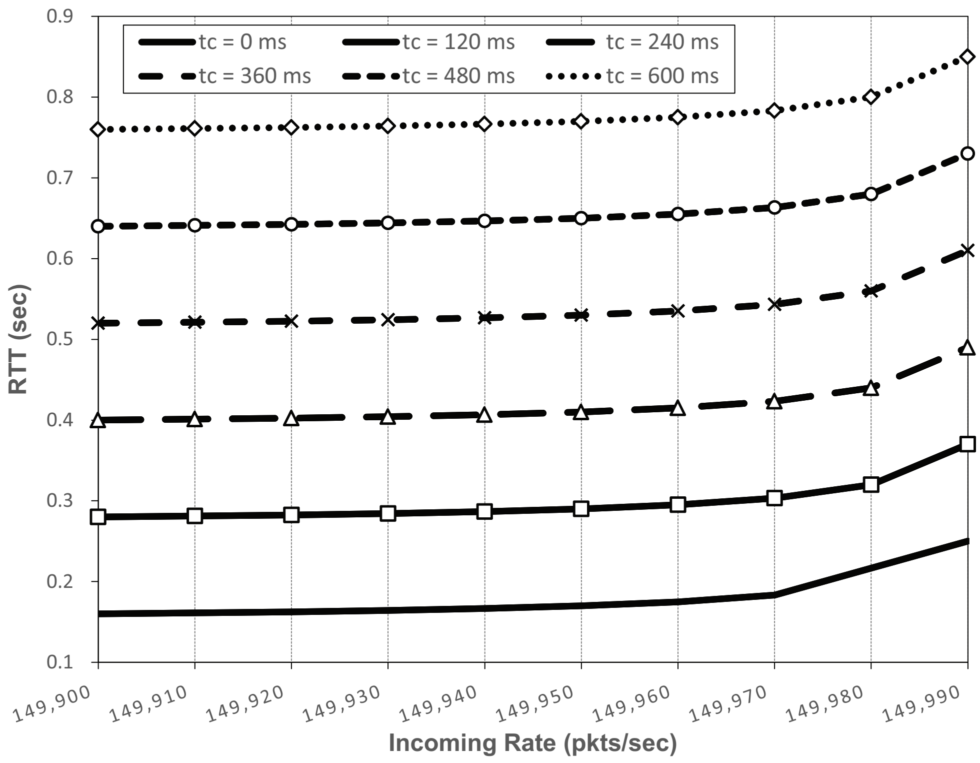


Figure 4.7: RTT versus λ under different t_c

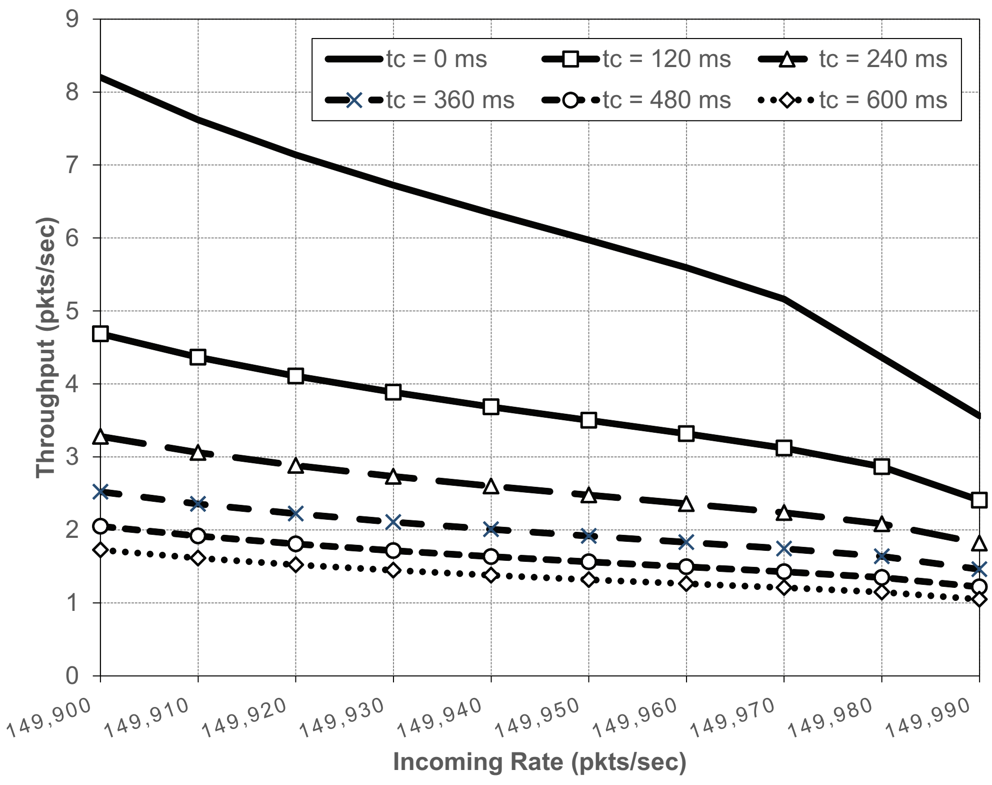


Figure 4.8: Throughput versus λ under different t_c

The λ figures can be summarized as, decreasing the effective λ is the only way to improve or at least maintain throughput in case of increasing the real λ, while RTT can be decreased by finding extremely new compression schemes. Those schemes should be extremely fast approaching the zero time of no compression while at the same time providing good compression ratio (CR). Although the effect of increasing buffer size cannot be taken lightly, together with the other factors it can be used to improve throughput especially when large memory is not an option.

4.4.4 CR and t_c effect on Probability loss with Different Buffer Size and Incoming Rate

In Fig. 4.9 the effect of different CR and t_c on the packet loss is plotted against B. The figure shows the improvement in p of a heavily congested network. The figure shows that increasing B is the main reason to decrease p. Doubling B can almost drop p by 40%. The relation is almost linear as shown in the figure. The role of both CR and t_c is restricted to being a linear offset to shift the linear relation between B and p. The black group of lines is the changes caused by different CR, while the blue group is the changes due to the different t_c. The t_c changes (blue lines) are almost in the middle as an average for the overall system with very low variance, i.e. weak effect. While the variance due to the CR changes (black lines) is clearly shown to be with much wider variance. Accordingly it can be deduced that the effect of CR is much stronger than the effect of t_c on reducing the p. When compared to B, highly aggressive compression with high CR can reduce the p by 14% which is only slightly better the reduction caused by simple B increases.

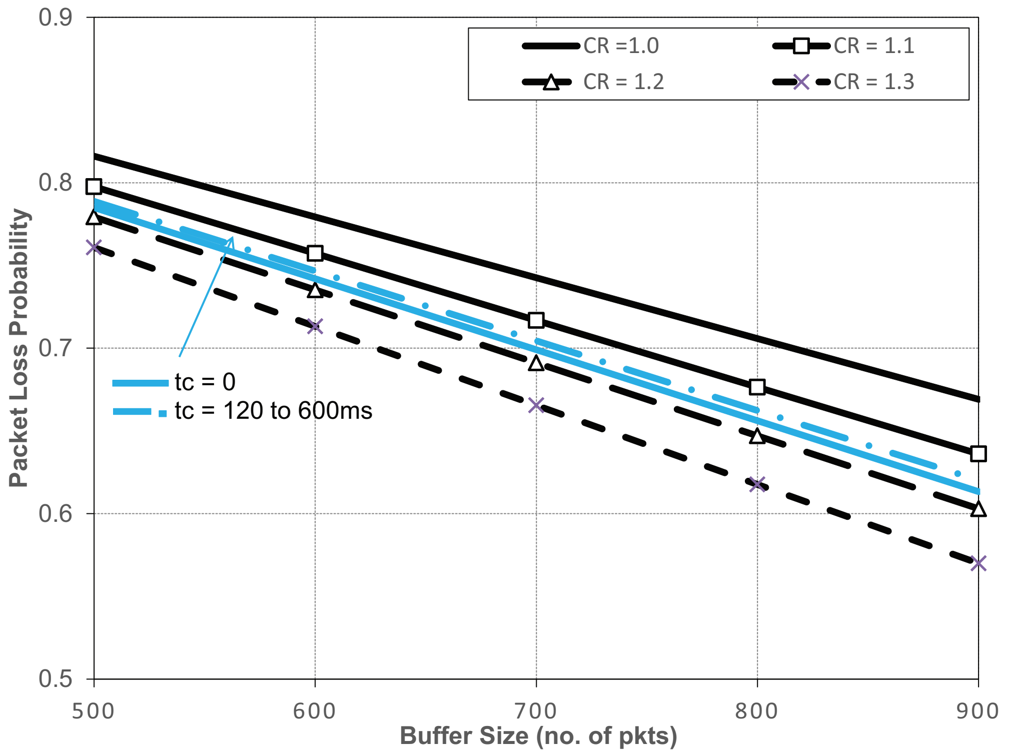


Figure 4.9: p versus B under different CR and t_c

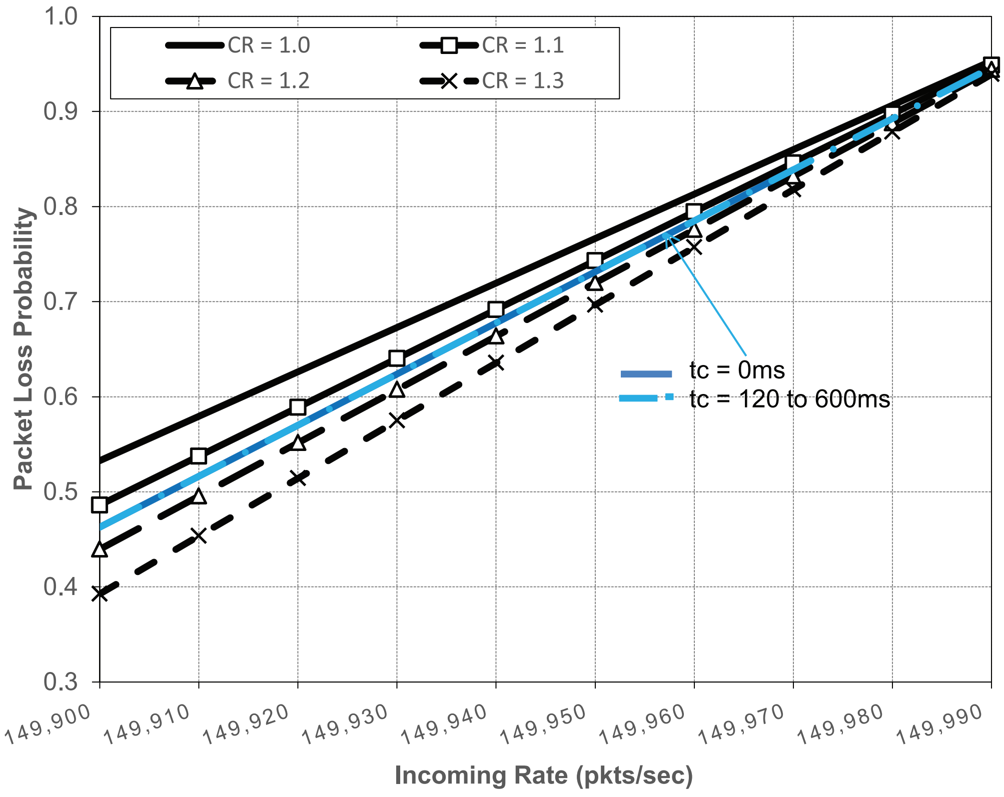


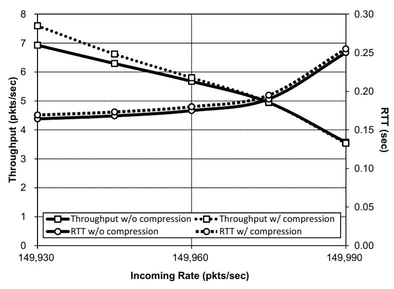
Figure 4.10: p versus λ under different CR and t_c

The effect of different CR and t_c on the p is plotted against λ. As expected the increase in λ will further deteriorate p in a heavily congested network. Faster compression (lower t_c in blue lines) can slightly reduce p as shown in Fig. 4.10. While more aggressive compression (higher CR in black lines) can further reduce the degradation of p, with stronger effect than t_c. The difference in variance is similar to that explained in Fig. 4.9.

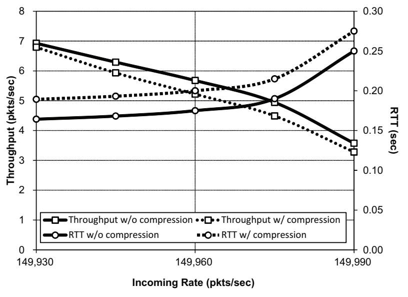
4.4.5 Emphasized Curves

Some of the curves that were presented earlier in this section are of special interest to this research. Those curves are separated and highlight in this section. The constant values and variable ranges of those curves are listed below:

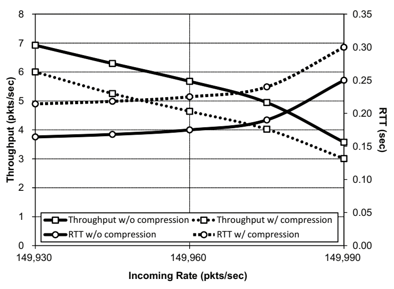
Window slope, b	$= 2$
Overall Compression Ratio, CR	$= 1.3$
Buffer size, B	$= 900$ packets
propagation delay, τ	$= 150$ ms
Compression time, t_c	$= 5$ ms, 25 ms, and 50 ms
Service rate, μ	$= 150000$ packets/sec
Input rate, λ	$= 149930 \sim 149990$ packets/sec



(a) $t_c = 5$ ms



(b) $t_c = 25$ ms



(c) $t_c = 50$ ms

Figure 4.11: Case study of throughput versus RTT with constant of CR, τ, and B

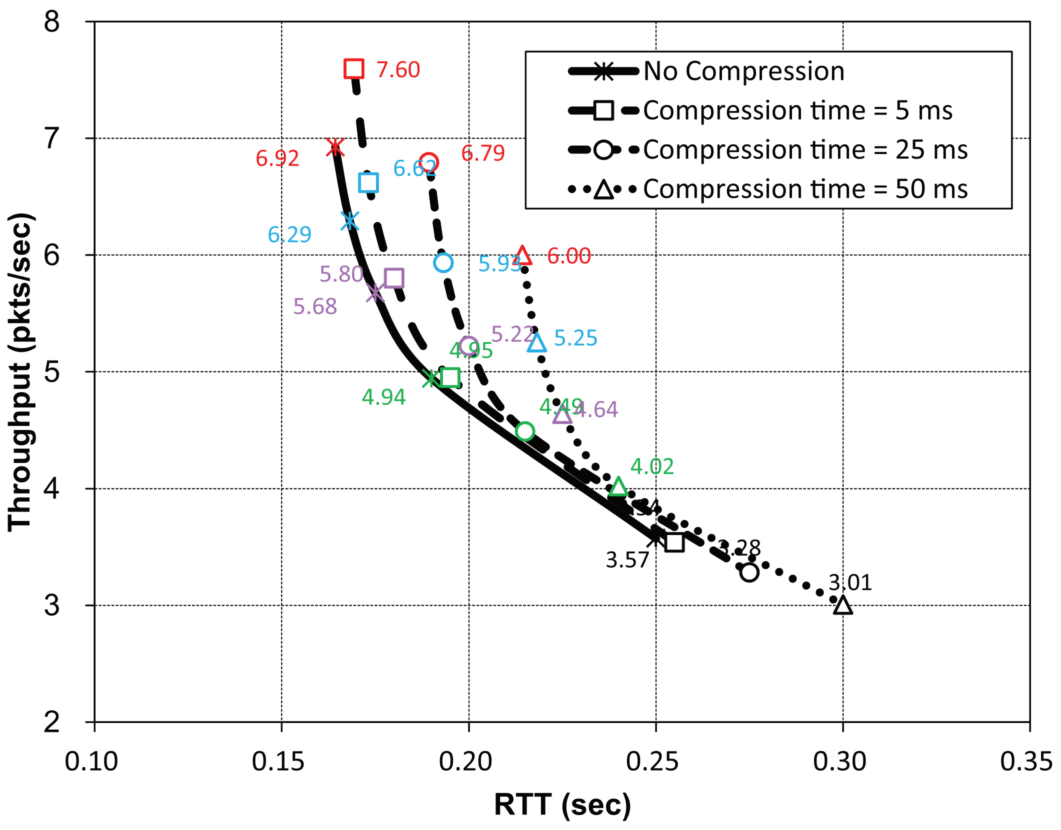
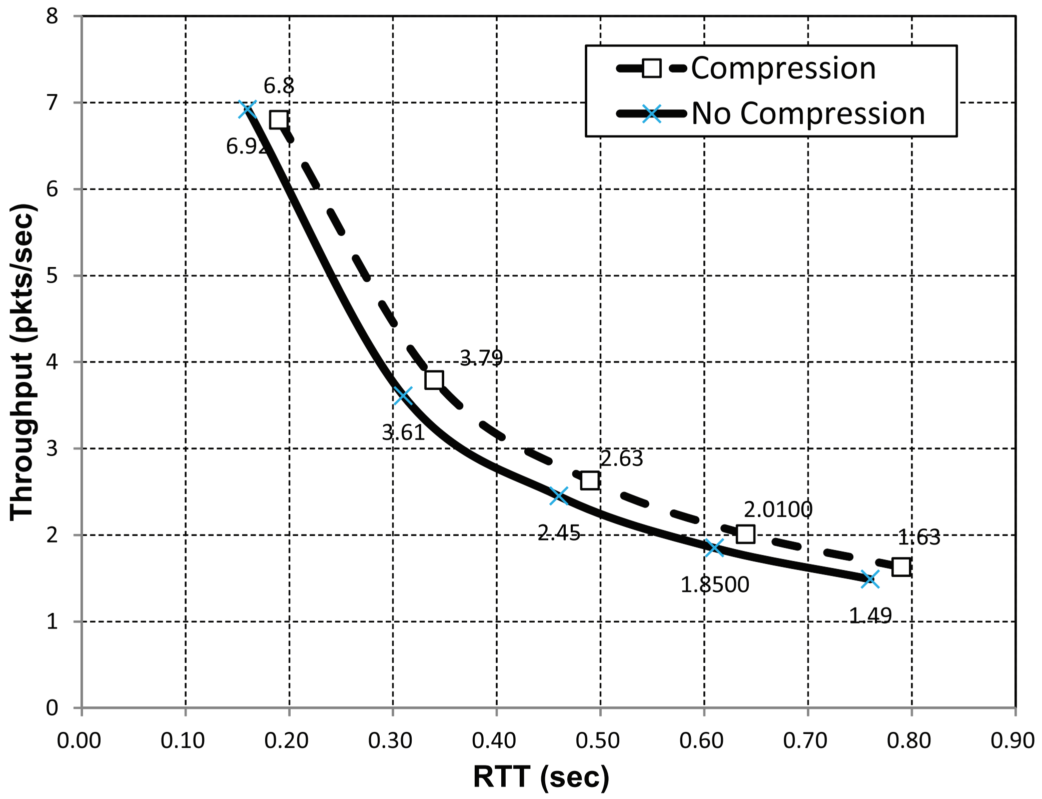
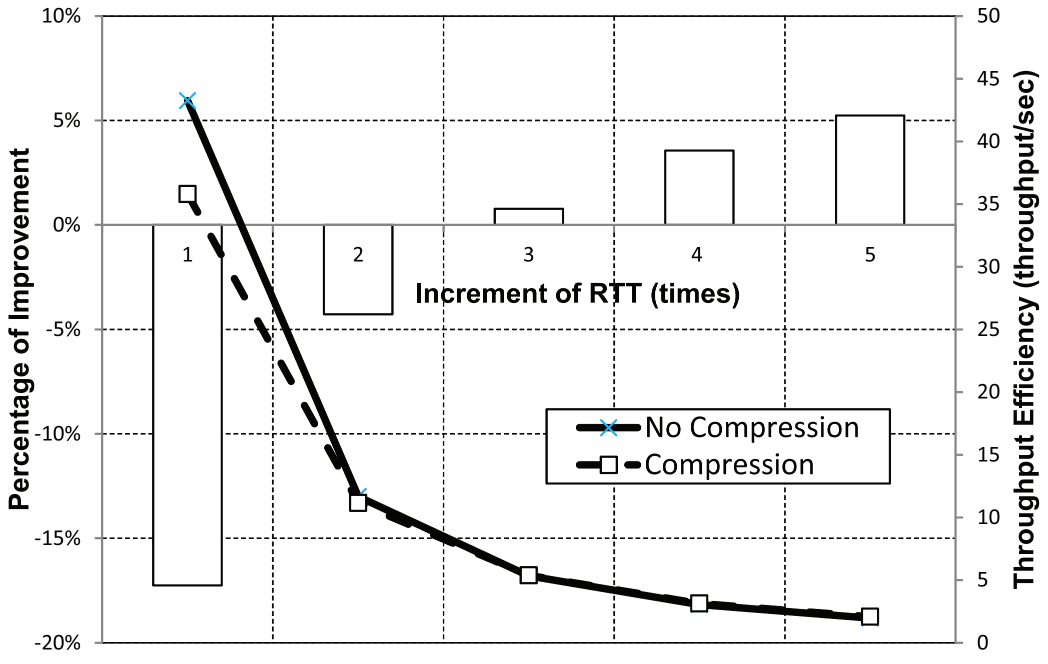


Figure 4.12: Case study of throughput versus RTT with different t_c



(a)



(b)

Figure 4.13: Case study of throughput versus RTT with different τ

Fig. 4.11 shows throughput and RTT curves versus λ for the three different compression times (t_c) above. As mentioned earlier, throughput with or without compression will always decrease as λ increases, RTT with or without compression gets worse as well. RTT with compression will always be linearly shifted higher from without compression according to the t_c value and always be worse in case of compression. With extremely swift compression, throughput is only better when incoming rate is already is very low. This soon changes when the input rate increases even slightly; the advantage of compression immediately disappears. In case of more realistic compression delay even in the future, both throughput and RTT are always way worse than without compression.

Fig. 4.12 shows the scatter graph of throughput versus RTT to visualize the correlation between the two different output for the different compression times (t_c) mentioned above. Each curve shows 5 constant input rates scenario λ (149930, 149945, ..., 149990) pkts/sec. In case of 149,930 pkts/sec (the red marked points), compression time with 5 ms has 9.82% better throughput than without compression, although RTT is 3.0% more. While the compression time with 25 ms, throughput is almost the same with or without compression and RTT is even longer for the same λ scenario. The throughput curve with 5 ms continues to have higher throughput and longer RTT than without compression until λ becomes 149,975 pkts/sec, where the throughput is almost equal to the throughput without compression curve. After that, the throughput of compression goes below throughput without compression. The realistic case of compression time with 50 ms is far worse with respect to both RTT and throughput. In summary, if the compression time can ever reach 5 ms or less while still maintaining very high CR, there is a chance for compression to improve throughput, but only when both B and λ are quite small and cannot be increased.

Fig. 4.13a shows the scatter graph of throughput versus RTT to visualize the correlation between the two different output for the different propagation delay, τ from 150 ms $\sim$ 750 ms, with 25 ms of constant compression time, t_c in RTT. The RTT results of compression are always way worse than without compression, but throughput are getting better when τ in RTT is increased. To clearly determine the effect of τ in RTT to the throughput, throughput efficiency with unit of throughput per second is computed [117]. Fig. 4.13b shows when the increment of RTT reaches 3 times of the starting τ (450 ms), throughput efficiency of compression outperforms the throughput efficiency of no compression.

4.4.6 Summary

All the curves show that throughput decreases with any of the following factors sorted according to importance; λ increase, B decrease or t_c increase. Throughput improvement is mostly affected by λ then B more than it is affected by CR and t_c. Assuming that the input rate (λ) is uncontrollable, then CR and t_c can be used to improve the effect of B on throughput when more memory is not possible. Compression with more aggressive schemes even if it is slow, can really enhance throughput when added on top of increasing B. For RTT, it can only be decreased by finding extremely fast compression which approaches zero time, while still providing good compression to improve throughput.

Chapter 5

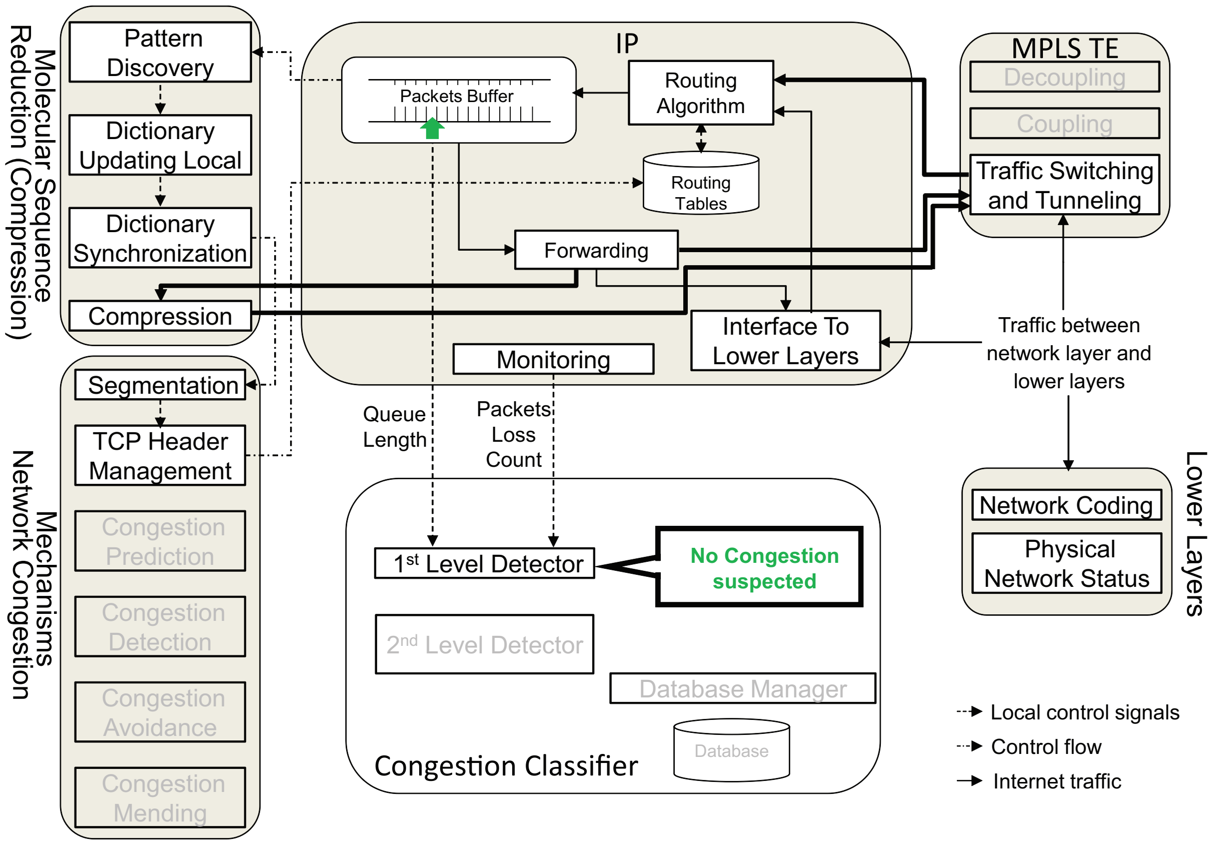
Efficient Congestion Management (ECM) Framework

5.1 Introduction

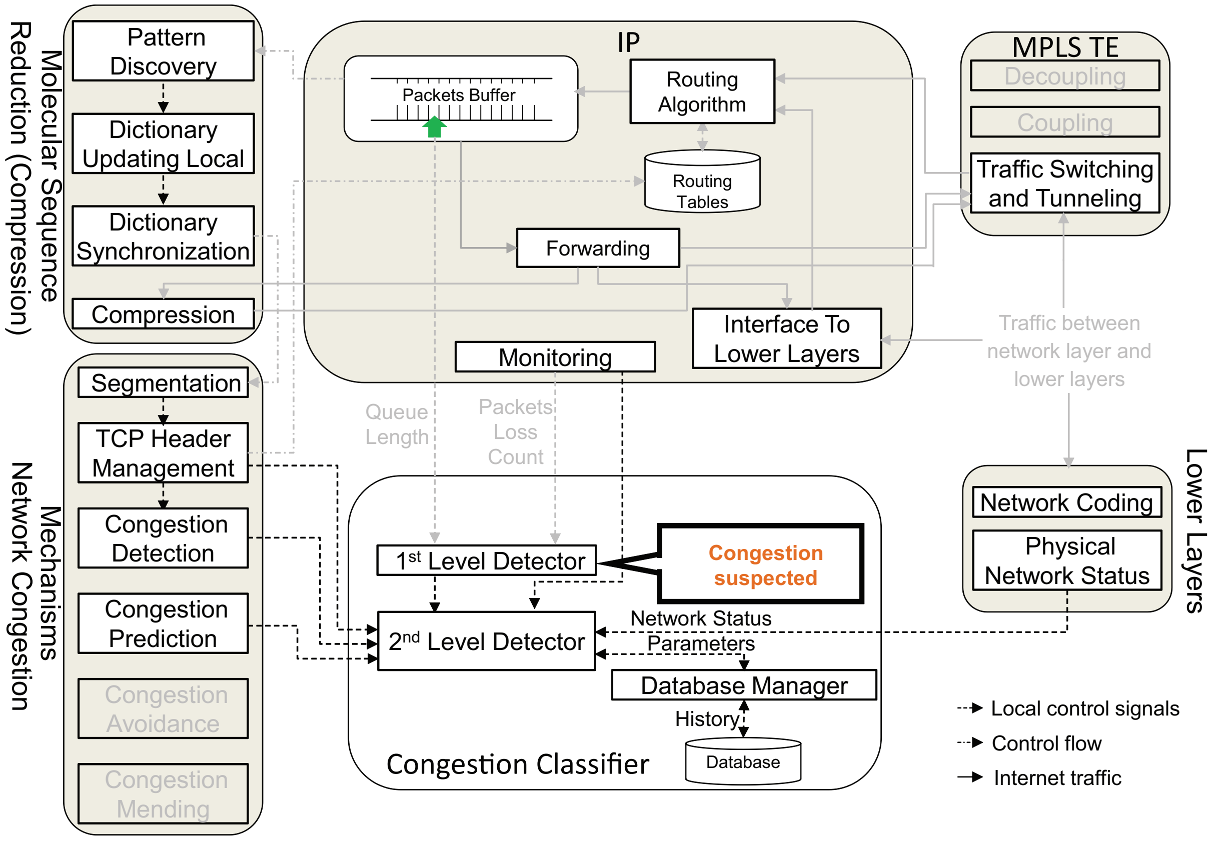
The purpose of ECM framework is to efficiently implement different existing congestion avoidance and congestion control approaches in one adaptive framework to minimize the impact of congested network while better utilizing network resources. There are numerous existing congestion avoidance and congestion control approaches, many of which have been already discussed in the Chapter 2. Most of those approaches were designed with one or more particular purpose in mind, which makes this framework useful as multipurpose used in all network conditions. The ECM framework offers adaptive selection to manage the different congestion solving approaches according to adaptive learning from history. ECM is mostly a congestion management framework, sending control messages to other congestion control mechanisms to handle the different network situations.

Figure 5.1 shows the block diagram of proposed ECM framework in the different steps of traffic or operation flow. The framework is an overall network manager coordinator between the different layers of the network stack, ranging from application layer (e.g., Molecular Sequence Reduction) till the physical layer (e.g., network coding). The additional congestion classifier introduced in the framework together with the accompanying control signalling and traffic forwarding links are used to achieve the coordination of the framework.

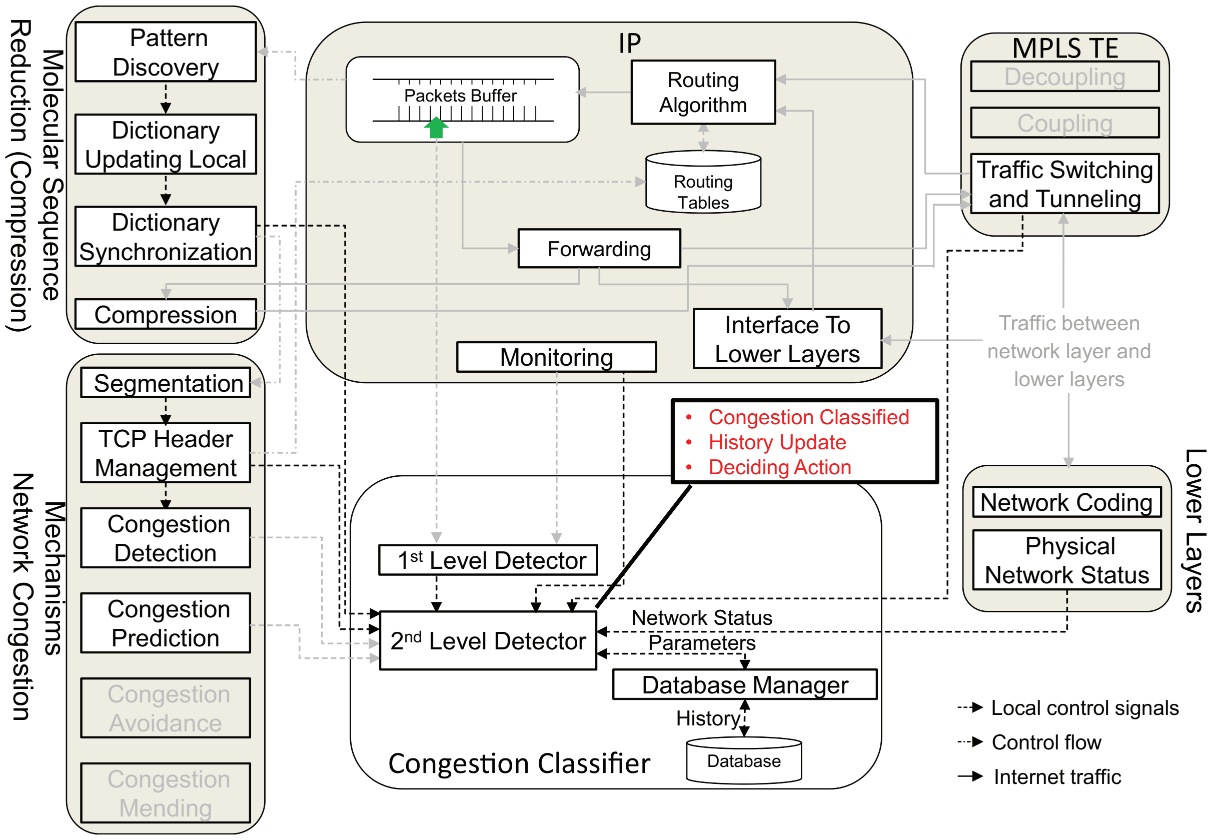
The congestion detector in the classifier is actually split into two detection levels. First, the initial detector or first level, roughly and swiftly checks congestion occurrence. Accuracy is sacrificed in the first level in favour of both speed and energy. When disabling the rest of the unused framework modules until congestion is suspected, energy that would have consumed by those extra modules could be saved. Additionally, taking those modules of the critical path of the framework operation helps speed up the framework operation in normal traffic status.



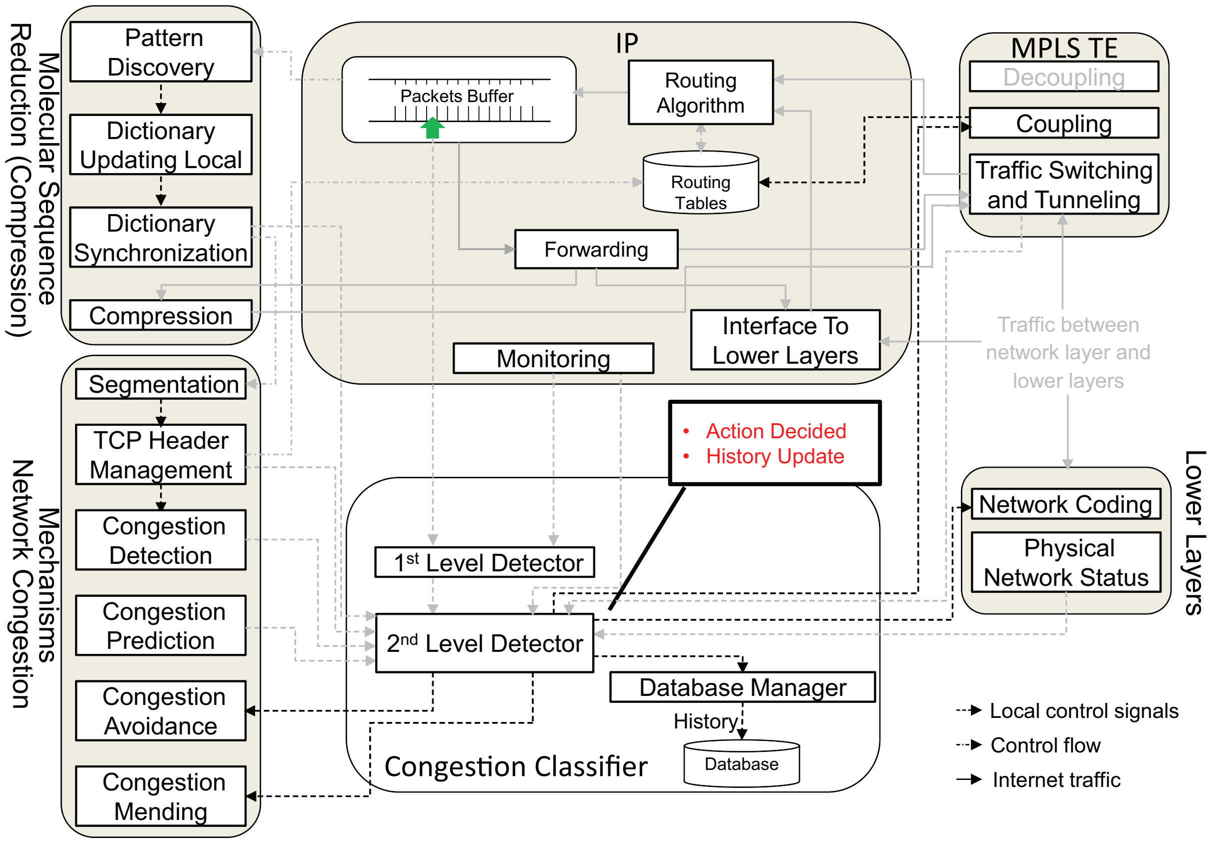
(a) Flow diagram of ECM Framework in case of Normal Network Traffic



(b) Flow diagram of ECM Framework in case of Congestion Suspicion



(c) Flow diagram of ECM Framework in case of Congestion Classified



(d) Flow diagram of ECM Framework in case of Action Choice

Figure 5.1: ECM operation

5.2 ECM Operation

In the beginning, the first level congestion detector simply monitors queue length and the time it last changed combined with the packet loss count. The more sophisticated classification is used in the second level congestion detector (the actual congestion classifier) to compensate for the inaccuracy of the first level detector. The congestion classifier detects traffic load properties by monitoring a lot of network status parameters from all over the different network layers. The obtained traffic load properties are then compared with the history kept in a dynamic database of the previously encountered network situations, to help make a more accurate classification of the current network situation. The current network situation descriptor together with its classification and the correctness of that classification will be also added to the database to help improve the classification the future cases.

The database also records the history of the actions have been performed previously when the same (similar) network situations were classified. The impact of those actions will also be recorded to know the effectiveness of the action chosen in resolving the corresponding congestion.

Fig. 5.1 is split into four parts showing the steps of managing congestion when encountered by the proposed framework. Part (a) shows the flow of normal traffic before congestion occurs. The first level congestion detector of the framework will keep checking for possible congestion symptoms. As long as the preliminary check done by the first level detector is negative, the remaining modules of the framework are idle and indicated with grey colour text or background. Control and flow lines that are not used in this mode of operation are also omitted in this part of the figure and shown when first used and explained in the respective figure parts.

At the same time data packets flowing out from MPLS connections end and also up from other lower networking layers are passed by the routing algorithm first. The routing table is checked to select the destination per packet and save that selected destination together with that packet in the network layer packets buffer. The destination can simply be one of the existing physical links of the device or a virtual link (tunnel) connection as MPLS or ATM (only MPLS is shown in this model here for simplicity). When the routing table chooses which to link the packet should be forwarded, regular load balancing mechanisms are utilized to improve the network utilization and general performance as well as further minimize the chances of congestion occurring.

Each packets saved in the network layer packets buffer will wait, until its turn comes to be forwarded to its assigned destination link. Some links, as MPLS, can be much faster than the others and can accept more than one packet simultaneously while other links are busy and their packets are still waiting. Compressed MPLS virtual links can even be faster since it starts by copying memory to memory of whole block of packets destined to same sub-network address. After copying the whole block of packets, MSR starts to filter out the packets that will damage the required correlation and reduce the required compression rate. Those packets are immediately forward on to the MPLS without any compression. The selected packets for compression are then compressed to result a new set of considerably smaller size before also being forwarded to the same designated MPLS connection.

Traffic to be decompressed proceeds in the opposite direction but omitted from the figure for simplicity. The compressed traffic emerging from the end device of the compressed MPLS connection is directly forwarded to MSR decompression module by-passing the previously mentioned routing of regular MPLS connections. The decompression module will check for uncompressed packet and forward it back to the routing algorithm. Packet sequences that are marked as compressed will be uncompressed first before also passing to the routing algorithm.

Concurrently the MSR additional traffic to handle the continuous local update and inter-device synchronization is encapsulated by the help of the local TCP layer of the device and then also forwarded to the routing algorithm. MSR pattern discovery module is continuously scanning the packets in the network layer buffer regardless its destination and updating its local dictionary when needed. MSR pattern discovery module is also collecting statistics about the current content of the buffer per destination sub-network. The collected statistics will be used to estimate the expected compression ratio, that estimation is used by the congestion classifier to choose the suitable action to take according to the detected network congestion classification.

Fig 5.1 part (b) shows the control flow when congestion checking done by the first level detector is suspecting congestion. Greyed lines are used to indicate control and flow lines or modules already explained in the previous part of the figure and not any different in this figure. The same gesture is applied in the next two parts of the figure.

Since the detector rule to classify congestion is very primitive and inaccurate it cannot be trusted as a final congestion detector and the second level detector is invoked. The second level detector starts by collecting more network parameters from different layers of the network of the device. Network coding level, SNR is useful to judge if the extra delay encountered is congestion of due to high noise bursts in the physical network. Extra transport and network layer statistics are also used to provide a more in-depth knowledge of the current network conditions and traffic patterns. The output of one or more traditional congestion detection mechanism is also used to guide the classifier together with congestion predicting mechanisms.

The collected network monitors are used to generate some network condition representation vector. The vector is passed to the database manager to find if the same (or similar) network condition occurred before in the recorded history. If not found the newly encountered condition is added to the database and its classification will be added later. When similar network condition was encountered, the (immediate, partial or full) history is retrieved together with its previous classification and score (probability) of each classification. The retrieved score history is used to make the classification of the existing network condition by voting mechanisms. If the network condition did not have history in the database, a group of the different network status categories can be assigned default or random values as an initialization.

Later when the classification is confirmed by future traffic, some voting evaluation formula is used to assign a score to the classification and saved to its record in the database. No further action is needed, if the classification decides that the current network conditions indicate no sign of congestion for sure or with some confidence degree. Other more advanced actions can be utilized to improve energy saving and other requirements but not currently considered.

Fig 5.1 part (c) shows the control flow when congestion checking done by the second level detector decides the existence of sure congestion with some confidence degree. The congestion category detected is also ranked with different levels; ranging from heavy congestion to light. Depending on both the congestion level and the confidence degree, congestion can either be predicted to occur soon in the future or detected in the current time instant.

After completely classifying current network status, the congestion classifier collects more monitoring statistics from the MPLS management module as well as the MSR module. The newly collected statistics are aggregated with the previous statistics used by the detectors to help choose the suitable actin for this network status. The database manager is also used to provide historical records of what was done to handle similar situations before, together with the score of how effect those actions were in improving the different network performance.

Fig 5.1 part (d) shows the control flow when the action is finally chosen to handle the current network status. One thing is that the database is again updated to record the action selected and attaching it to the record of the current network status. At the same time, the congestion classifier sends control messages either local to other mechanisms in the different network layers of the device or to other network devices. Depending on the chosen action(s) and the host network device, the control signalling links are indicated in the figure.

For example, one or more TCP congestion avoidance mechanism can be initiated if the network status was classified as congestion predicted. Similarly congestion mending manoeuvres can also be initiated if the status was classified as some level of congestion. Network coding can also be controlled either to higher faster level (more energy) or vice versa, depending on the obtained classification of the current network status. Lowering the quality of data (accuracy or QoS) is also considered as one of the possible actions for congestion management when handling lossy data streams. The degree lowering of QoS can be chosen depending on the level of congestion detected or predicted.

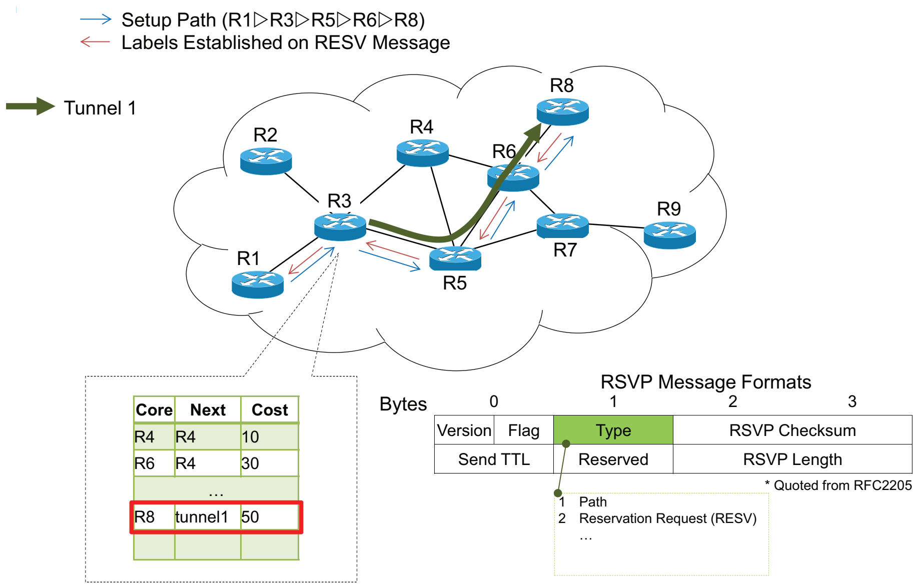
The newly added manoeuvres in this framework, is dynamically establishing MPLS connections to assist in the management of network congestion. Since MPLS connection requires some substantial time, this action is restricted as an option to be performed only when predicting near future congestion with enough time to establish the connection. So that by the time the predicted congestion occurs, the MPLS connection would have been established. The MPLS coupling module is responsible for performing this action together with the afterwards bookkeeping of modifying the routing tables for the future traffic to start flowing into the MPLS channel either explicitly or load balanced with other links.

The MPLS coupling module is also responsible for establishing compressed MPLS connections to other compression capable network devices. Similar to the regular MPLS connection, the routing tables are updated with some indication of the compressibility of this MPLS connection when needed. The action of decoupling an existing MPLS connection is currently not included for simplicity relying on some MPLS connection timeout to tear down long unused MPLS connections.

Depending on the severity of current network status category, more than one action can be combined or speed up the resolving of the encountered congestion. The next time the second level detector is engaged, the time elapsed since the last decision is checked. If the time gap is considerable enough, the previous classification and action(s) are assumed to have been correct and efficient to resolve the respective network status. That combination of classification and action(s) is assigned a high score (probability). This way it will be favoured for classification and handling the next time the same situation is encountered. On the other hand, if the gap between two invocations of the second level classifier is small enough, the previously action(s) and may be classification are assumed in adequate and assigned lower score. The lower score decrease the likelihood of repeating the same mistake.

More sophisticated history sequence dependence can be introduced in the second level classifier to provide better classification based on the knowledge of previous states preceding the current state. It can be useful in cases like failing to avoid congestion after prediction, which should help classifying the current state as sure congestion that is too heavy that the avoidance mechanisms were not enough to stop it from occurring. Thus next time better apply more severe avoidance mechanisms. Similarly when the prediction was too late, the classifier should update a larger set of network status in database as the set preceding congestion. This way congestion can be predicted much earlier, for instance to give enough time to establish an MPLS connection. An opposite example when overacting occurs, by choosing an action that is too strict or over estimating the severity of compression. The general network performance will drop without any real reason. By updating the respective classification and/or action scores, the network utilization should be improved the next time the same situations are encountered.

5.3 ECM Operation over MPLS



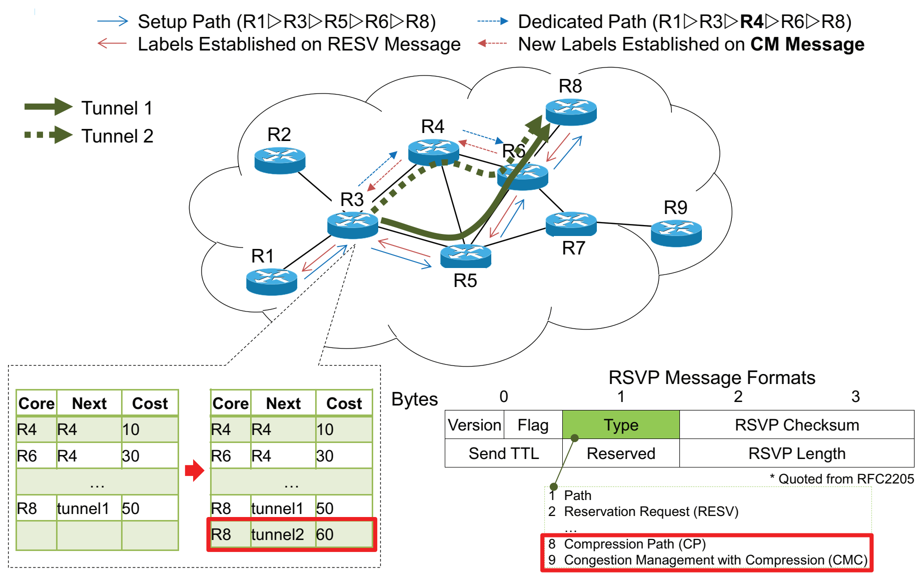
Core	Next	Cost
R4	R4	10
R6	R4	30
…		
R8	tunnel1	50

Version	Flag	Type	RSVP Checksum
Send TTL		Reserved	RSVP Length

1 Path
2 Reservation Request (RESV)
…

Figure 5.2: Normal operation of MPLS-TE

Routers use the MPLS for underutilized links caused by Internet IP routing/forwarding paradigms. Traffic engineering (TE) component is employed to steer the destined traffic from following the optimal path. TE enables better bandwidth management and utilization between routers. The key to implementing a scalable and efficient TE component in core networks is to gather information on the traffic patterns and traverse the network to guarantee bandwidth is better utilized. The scope of TE is limited to networks with congestion which happens none frequently and only in few routers. TE virtual links between two distant routers to better use the available bandwidth. Thus, TE might relieve those temporary congestion.


Setup Path (R1▷R3▷R5▷R6▷R8)
Labels Established on RESV Message
Dedicated Path (R1▷R3▷R4▷R6▷R8)
New Labels Established on CM Message
Tunnel 1
Tunnel 2
R1
R2
R3
R4
R5
R6
R7
R8
R9
RSVP Message Formats
* Quoted from RFC2205

Core	Next	Cost
R4	R4	10
R6	R4	30
…		
R8	tunnel1	50

Core	Next	Cost
R4	R4	10
R6	R4	30
…		
R8	tunnel1	50
R8	tunnel2	60

Bytes	0		1	2	3
	Version	Flag	Type	RSVP Checksum	
	Send TTL		Reserved	RSVP Length	

1 Path
2 Reservation Request (RESV)
...
8 Compression Path (CP)
9 Congestion Management with Compression (CMC)

Figure 5.3: ECM operation of MPLS-TE

TE is inspired by constraint based routing (CBR), which takes into account the possibility of multiple paths between a specific pair of source and destination routers. The resource availability and link status information are computed using a constrained shortest path first (CSPF). CSPF uses bandwidth among the cost metrics to compute an ordered set of IP addresses as the next hops along the path of the TE tunnel or LSP. TE is used to transfer information through to virtual tunnels pre-configured on a router. TE uses resource reservation protocol (RSVP). As illustrated in Fig. 4.8, TE tunnels are configured in between ingress (headend) routers, R3 and egress (tailend) router of the core network. The TE tunnels are assigned specific label switched path (LSP) routed via the pre-configured paths. Only tailend routers perform path selection based on the ordered sets computed by CSPF. Those routers reserves resources by RSVP to update the LSP path and labels in the tables of the involved routers [119].

The four main messages used in RSVP for TE are PATH, RESERVATION REQUEST (RESV), ERROR and TEAR. The PATH message is generated by the headend routers. In other words, the headend router asks for information from the routers in the network through the RSVP signalling to ask for bandwidth availability for the tunnel. At each hop, the PATH message checks the available bandwidth and stores this information into the message. The list of routers is calculated upon receiving on the tailend router. Other routers in the network do not perform CSPF computation. The RESV message is created by the tailend router and it is used to verify the reservation request that was sent with the corresponding PATH message.

In the network depicted in Fig. 4.8, router R8 will generate the RESV message in response to the PATH message. The ordered set of IP addresses indicates the routers of the path R1, R3, R5, R6, and R8 are chosen. As a result, *tunnel1* is established in the core network.

The TE is extended to enable the implementation of ECM with compression by using the RSVP signalling. The modifications are new RSVP messages, two additional types of message are proposed. Compression Path (CP) message and Congestion Management with compression (CMC) message are added to the new RSVP protocol extension. The CP message is used to request an uncongested path with compression supporting routers; it is generated by the headend router. Then the CMC message allocates the bandwidth to the new compressed TE tunnel and it is generated by the tailend router. The tunnel between the headend and the tailend router is referred to as coupling data flow. Fig. 4.9 shows two paths, i.e., *tunnel1* and *tunnel2* are established between routers R1 and R8, *tunnel1* is the regular MPLS tunnel while *tunnel2* is added using the CP and CMC messages to transfer compressed data flows.

When decoupling data flow, the tailend router uses the TEAR message to clear the path reservation instantaneously. The reservations of the links are cleared from the routers so that bandwidth can be reused for other traffic.

In general, one of the TE functions is to reduce the congestion in the core network. In particular, the TE does guarantee the congestion can be perfectly solved. This is because the future congestion, which is depending on the amount of data traffic admitted into the core network, can be predicted by TE. When the newly ECM with compression is applied on the MPLS, the compression path is intelligently performed to reduce the total bandwidth. As a result, the extra bandwidth is available to transport other traffic. By this way, the TE tunnel is efficient to carry more traffic in the core network. Furthermore, some optimization algorithm can be used to provide a better load balancing management to solve efficiently the encountered congestion in the entire networks.

5.4 Summary

The proposed framework is mostly a sophisticated congestion classifier tightly coupled with network management of other network layer mechanism. The classifier can categorize the different network status with high accuracy, based on the monitored statistics collected from the different network layers. The management functionality of the proposed congestion classifier have orchestras the involved network layers, modules and mechanism to better manage network congestions. Aided with a detailed recorded history, the classifications and actions decided by the congestion classifier are continuously corrected to provide continuous improvement of the performance of the framework. Most of the framework functionality is performed off the critical path of the routing functionality away from the main traffic flow. Thus the network performance is almost not affected at all by the framework in case of normal flow conditions. The effect of the framework should be only obvious in congestion situations, where the framework tries to coordinate the different layers that perform some congestion management for better responsiveness.

Chapter 6

Efficient Congestion Management with Compression (ECM/C) for Small Devices

6.1 Introduction

It has been shown in Chapter 4 that reducing the incoming rate of data traffic into a congested network is the main and may be the only way to solve or avoid congestion. The problem is reducing the incoming rate risks leaves some network resources under-utilized to keep safety margins from congestion. The motive of using compression in the model presented in this chapter is finding a way to take the advantages of reducing the incoming rate without really reducing it. By reducing the effective incoming data size while keeping the actual incoming input rate as it is, congestion can be avoided or relieved.

Compression would still prolong *RTT* and thus the network performance including throughput will be degraded, but definitely much better than in case of regular congestion management mechanisms. Regular compression schemes as LZW, are characterized by extremely large memory footprint and compression delay. While the delay is mostly data dependent as well as technology dependent and cannot really be improved, the memory requirements can be compromised to some extent.

Lightweight DeCompression (LDC) is a customized compression technique that requires minimal additional memory for decoding compared to LZW. Some devices have smaller buffering and bandwidth, LDC is introduced for those low memory devices. LDC decoder is a lightweight process that can be implemented in hardware unlike all the other existing compression schemes, which make it more appealing for limited memory devices. In addition, such small devices cannot perform long complicated calculations or memory intensive operations as required by LZW. That was the design goal for LDC. LDC managed only in half duplex compressed data stream to save incoming bandwidth to those small devices, as shown in Fig. 6.1. Half duplex compression here means one way traffic only is compressed while the opposite way is not compressed, unlike the regular meaning of full and half duplex transmissions. The powerful encoder device will be responsible for the complicated memory intensive compression process. The reverse traffic will not be compressed, since the limited small device does not have enough resources. The tuning of the LDC algorithm is also described in the end of this chapter.

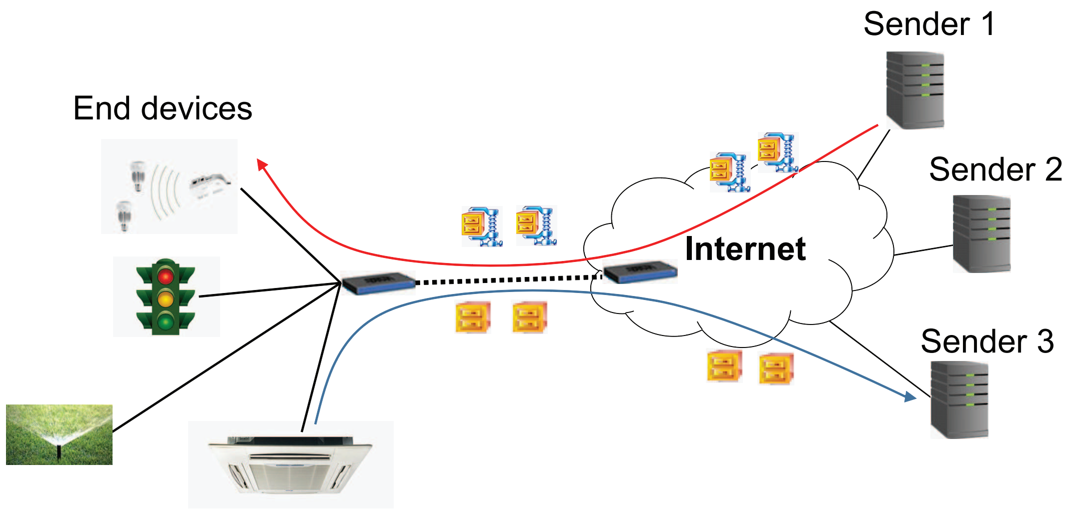


Figure 6.1: LDC network topology

6.2 New Dictionary based Compression Scheme with Lightweight DeCompression (LDC)

Simply reducing the transmission rate at the transport layer of the end devices would avoid a lot of packets retransmission which would probably suffer from timing out in congested networks. Reducing packet rate might make it difficult to maintain high throughput and utilization when congestion is not that severe or was not really there from the beginning. Data compression can be one of the viable solutions to help in congestion handling to maintain high throughput with reducing the effective incoming rate instead of the real one.

The most widely used data compression algorithms of the LZ family are simple to implement, and has long provided promising compression ratio. Still, LZ needs huge additional memory during dictionary that can easily reach higher than the square of the data size itself, which might not be provided by some devices. In this chapter, LDC is introduced for low memory devices. LDC decoder is a lightweight process that can be implemented in hardware of limited memory devices. LDC decoding needs small additional memory, while encoding requires comparable memory to LZW and much longer compression delay and lower compression ratio. Since data streams are compressed outside of the network (above end devices TCP layers) not to cause additional packets timing out. When those streams are compressed as much as possible, the required bandwidth is reduced. The spared bandwidth is consumed for the retransmissions to resolve possible existing congestion beside the virtually reduced input rate. The effective throughput in that case would be higher than without compression.

An overview of the ECM/C model operation with its physical location either on the sender (compressor) or the receiver (decompresser) is shown in Fig. 6.1. Small limited memory devices are always acting as decoders receiving compressed stream of data from much powerful compressing devices. The network topology is also shown with half duplex compression traffic. In Fig 6.2, the stages of LDC compression are also shown.

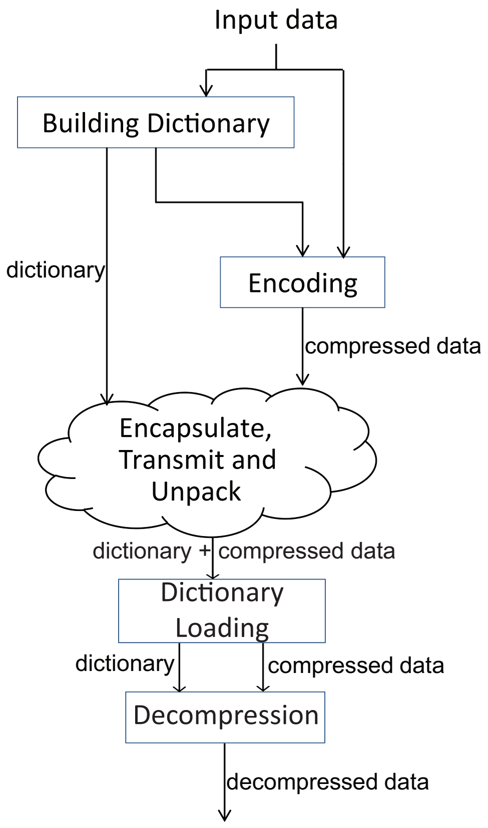


Figure 6.2: LDC Stages

List of Definitions:

Definition 6.2.1. (Codeword) *A codeword (c) is a group of two or more concatenated data unit. If the codeword has two data units and each symbol has two characters, then the maximum number of the codewords is* 256^2*, which is equivalent to* 65536*.*

Definition 6.2.2. (Codeword length) *A codeword length* (l_c) *is the number of bits of a codeword. If the codeword has two symbols and each symbol has two characters, then the codeword length is* 2 *symbols* $\times$ 2 *characters* $\times$ 4 *bits* $= 16$ *bits.*

Definition 6.2.3. (Code) *A code (C) is a mapping from a symbol or a codeword to a set of finite length of binary strings. (**representation code)*

Definition 6.2.4. (Code length) *A code length* (γ) *is the length of a code. If the code has* γ *bits, then it can encode at most* 2^γ *of symbols and codewords.*

Definition 6.2.5. (Fixed length code) *A fixed length code is a code such that* $\gamma_i = \gamma_j$ *for all* i,j*.*

Example: Suppose there is one symbol, {AB} and two codewords, {9F1B, 3E70B2}. In the fixed length code of 2 bits, the code would be $C(AB) = 00$, $C(9F1B) = 01$, and $C(3E70B2) = 10$.

Definition 6.2.6. (Dictionary) *A dictionary (D) is initialized to contain the single codeword corresponding to all th possible input characters. The dictionary is identical to the input source data.*

Table 6.1: List of notations

Symbol	Description
S_o	The size of source data (unit: byte)
S_c	the size of encoded data (unit: byte)
R	The sum of repetitions of all output symbols and codeword
P_s	The sum of repetitions of all output symbols that were not compressed
P_c	The sum of repetitions of all input codeword that have been compressed
Q	The sum of entries in the dictionary
α	The size of one input symbol in bits
$l_{c_{max}}$	The codeword length that has the maximum number of bits of a codeword length
$l_{c_{min}}$	The codeword length that has the minimum number of bits of a codeword length
$\bar{l_c}$	The codeword length that has the average number of bits of a codeword

To reduce the temporary memory requirements of the dictionary structures during encoding process, the dictionary size is limited. In the LDC compression, the dictionary needs to be transmitted to the decoder for decoding process. The codeword entries of the dictionary are limited to 256. It goes without saying that such small dictionary is the main reason LDC compression ratio will be much worse than that of any LZ variant, but when well utilizing the small dictionary the degradation of compression performance can be reduced. To achieve this, the dictionary is filled with higher repetition codewords first regardless of their position of occurrences. To calculate the repetition of the codewords before starting any compression, an additional pass of scanning the whole input which tends to double the compression time more than any other LZ variant. Each of the codewords allowed in the dictionary must fulfil the minimum repetition threshold (R_{thre}) below. If those codewords are too short and non-frequently used, the overhead in the dictionary can end up increasing the stream size.

$$R_{thre} = \frac{\alpha(l_{cmax} + 1)}{(\alpha l_i - \lambda)} \tag{6.1}$$

α is the size of one input symbol in bits. l_{cmax} is the maximum number of codeword length. l_i is the codeword length that between the maximum and minimum of codeword length. λ is a code length.

A fixed code length is used in the proposed LDC, which is a feature from an simple implementation engineering viewpoint [121, 122]. Therefore, the codeword in the dictionary must at least start from λ=9 bits to have enough space for the dictionary besides the original uncompressed data space. If the λ is 9, the dictionary will have 512 entries. Since the first 256 entries are reserved for the basic symbols, only 256 entries are usable. LDC algorithm operates in two passes, dictionary building and encoding.

6.2.1 Dictionary Building Pass

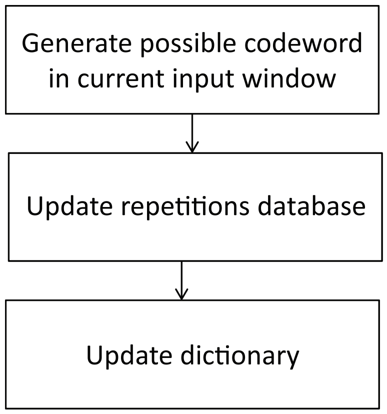
The dictionary building, shown in Fig. 6.3, is conducted according to Algorithm 1, where codewords are constructed by all possible combinations of input stream symbols sequences.

The constructed codeword are up to the maximum allowed codeword length. The repetitions of those codewords are counted to represent possible repetition in the final modified input stream. Only the codewords with highest repetition are added into the final limited size dictionary. A fixed threshold for minimum repetition is used to keep codewords with low repetition from entering the dictionary. If those codewords are too short the overhead in the dictionary can end up increasing the input stream. The threshold could be further set higher to speed up the process of inserting and replacing codewords in the temporary database while avoiding the need to sort the big list of all possible input stream symbols.

Fig. 6.4 shows an example of possible codewords forming. All the possible codewords are formed from $l_{cmin} = 2$ until $l_{cmax} = 4$, if needed they are added in the input database and/or its repetition count updated. The window keeps shifting until the end of the input stream. In the example, the generated codewords are 1234, 123, 234, 12, 23, 34, 2345, ..., and so on. In this example, no repetitions occurred and all counts are one assuming all combinations never appeared previously in the input.

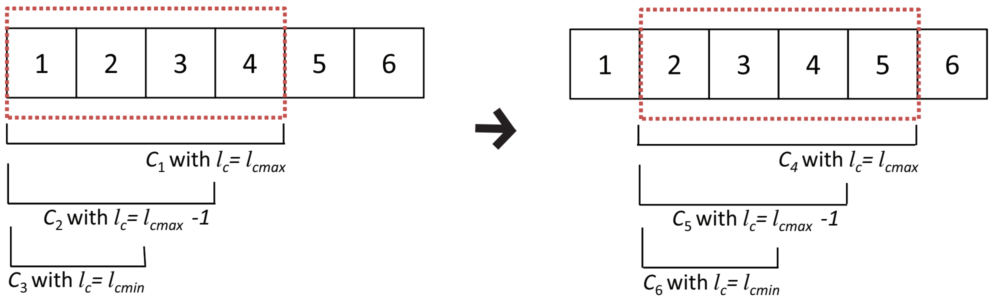
Algorithm 1 Dictionary Building

```
1:  Input window = read ($l_{cmax} - 1$)
2:  while (input window.append(read one byte)!=empty) do
3:      for (codeword=next possible longest codeword) do
4:          if !(codeword in input list) then
5:              insert (input list, codeword)
6:          else
7:              input list [codeword].count+=$l_c$
8:              if (codeword in the dictionary) then
9:                  continue
10:             end if
11:             if (input list [codeword].count < $R_{thre} * l_c$) then
12:                 continue
13:             end if
14:             if (dictionary full and input list [codeword].count==$R_{thre} * l_c$) then
15:                 continue
16:             end if
17:             if (dictionary full) then
18:                 remove (dictionary,lowest count codeword)
19:                 insert (dictionary, codeword)
20:             end if
21:         end if
22:         remove first byte window
23:     end for
24: end while
```



Generate possible codeword in current input window
Update repetitions database
Update dictionary

Figure 6.3: The flow of dictionary building process



1
2
3
4
5
6
C_1 with $l_c = l_{cmax}$
C_2 with $l_c = l_{cmax} - 1$
C_3 with $l_c = l_{cmin}$
C_4 with $l_c = l_{cmax}$
C_5 with $l_c = l_{cmax} - 1$
C_6 with $l_c = l_{cmin}$

Figure 6.4: An example of possible codeword formation

6.2.2 Encoding Pass

Fig. 6.5 and Algorithm 2 show the encoding algorithm. In this pass, the input symbol sequences are encoded by searching for them in the dictionary. If a match is found in the dictionary, the index of the matching codeword position in the dictionary forms the representing compressed data. If no match is found, just bit encode the input data unit into the corresponding enlarged representing compressed size.

Algorithm 2 Encoding Process

```
1: while (inputWindow = readNextInputWindow)!=empty do
2:     for (codeword = nextLongestPossibleCodeword (inputWindow)) do
3:         if (foundIndex=codewordExistInDictionary ()) then
4:             output (bitPacking(foundIndex))
5:             removeCodewordFromInputWindow()
6:         end if
7:     end for
8:     output (bitPacking(firstByte))
9:     removeFirstByteFromInputWindow()
10: end while
```

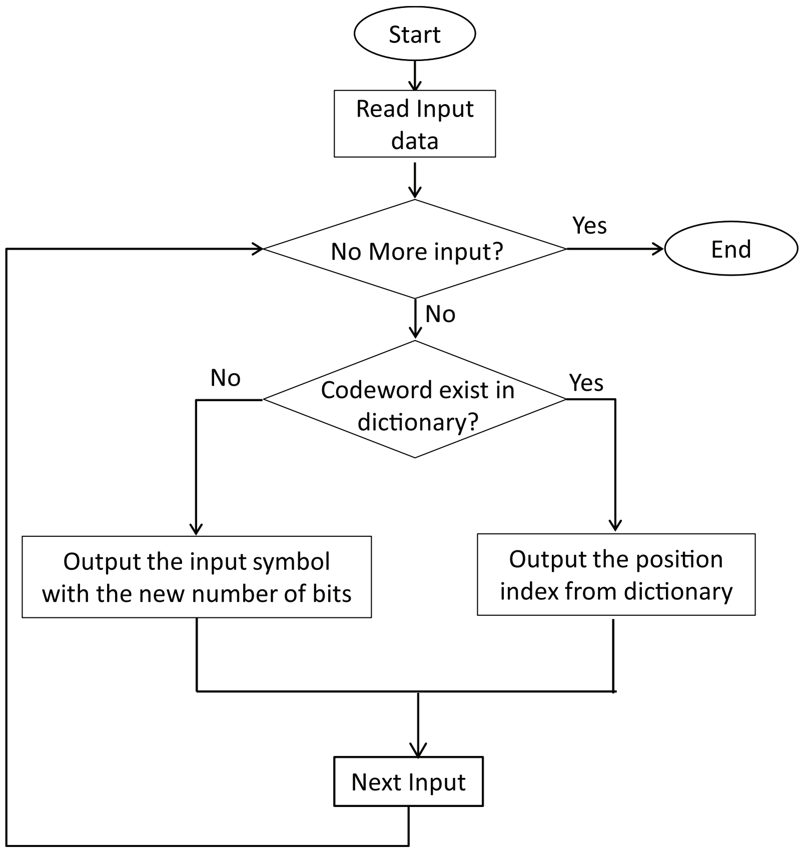

Start
Read Input data
No More input?
Yes
End
No
Codeword exist in dictionary?
No
Yes
Output the input symbol with the new number of bits
Output the position index from dictionary
Next Input

Figure 6.5: The flow of encoding process

6.2.3 Decoding

The decoding process consists of two steps, dictionary loading and decompression. The dictionary is first received from the encoder. After checking the validity of the dictionary, it is loaded to prepare for the decompression step. The end of the dictionary packet(s) is signalled from the encoder by a flag/token or separating dictionary packets from the compressed stream. The dictionary can be shared among different input blocks and updated explicitly if it is separated from the compressed stream. This can further help in improving the compression performance faster (less bandwidth) and better compression ratio. That approach is similar with the MSR dictionary synchronization feature. LDC has a much smaller dictionary which makes it easier to just transmit the whole dictionary when needed instead of generating a lot of additional traffic and complicated resource hungry dictionary synchronization of MSR. The new dictionary generated by the compressing device can be compared to the previous one, if not much different and the compression ratio improvement by not transmitting the dictionary should be compared to the compression ratio degradation of using a little different dictionary.

Decompression step is shown in Algorithm 3. Compressed data stream bits are unpacked, the bit flag is checked. If the data is from the dictionary, output the corresponding symbols sequence of this codeword. If the flag indicates it is not from dictionary, just bit decode the compressed data.

Algorithm 3 Decompression

```
1: while (true) do
2:     index = unpackBits (compressedStream)
3:     if (index < DictionarySize) then
4:         output (index)
5:     else
6:         output (dictionary [index-256])
7:     end if
8: end while
```

6.3 Parameters Tuning

6.3.1 Code Length

When the code length, $\gamma = 8$ bits, there is no space in coding bits to represent any additional representing code from the dictionary. The data block can only be compressed when the repetition of the codeword is high. In order to have space for representing the dictionary indices, γ must start from 9 bits upwards. In this study, the upper bound of compression ratio in the proposed compression algorithm is computed with the sum of repetitions of all output symbols that were not compressed, $P_s = 0$, the fixed codeword length with minimum value, $l_c = 2$ is used, and total number of entries in dictionary is fully utilize, which is $E_t = 2^\gamma - 2^8$. When $P_s = 0$, it means the data packet is perfectly compressed, the S_o is only encoded by the codeword. In this case, the best compression (CR) that can achieve is given by

$$CR\mid_{P_s=0} = \frac{8 \times l_c}{\gamma} + \frac{P_c}{E_t(l_c+1)} \tag{6.2}$$

The upper bound of CR is given by $CR \leq CR\mid_{P_s=0}$. The worst case of CR in the proposed compression algorithm can be computed when the $P_c = 0$ is assumed. The worst compression (CR) computes by

$$CR\mid_{P_c=0} = \frac{8}{\gamma} + \frac{P_s}{E_t(l_c+1)} \tag{6.3}$$

The lower bound of CR is given by $CR \geq CR\mid_{P_c=0}$. The upper bound of CR for the proposed compression algorithm is showed in Fig. 6.6. The CR decreases, when the code length increases. In order to have better CR with fixed codeword length of 2, 9 bits was found to be the best selection for the data tested here. From Fig. 6.6, the best performance for the proposed compression algorithm can reach 1.778. This means the compression algorithm can save up to 44% of the resources for that specific input giving a chance of other streams to make use of the freed bandwidth. In some cases, fixed codeword length might enlarge the compressed data size. The worst performance of this algorithm reaches is 0.889, which enlarges nearly 12% of the original data size. However, this problem can be solved by pre-determination to reject compressing bad compression ratio data and just revert back to the original data after compressing.

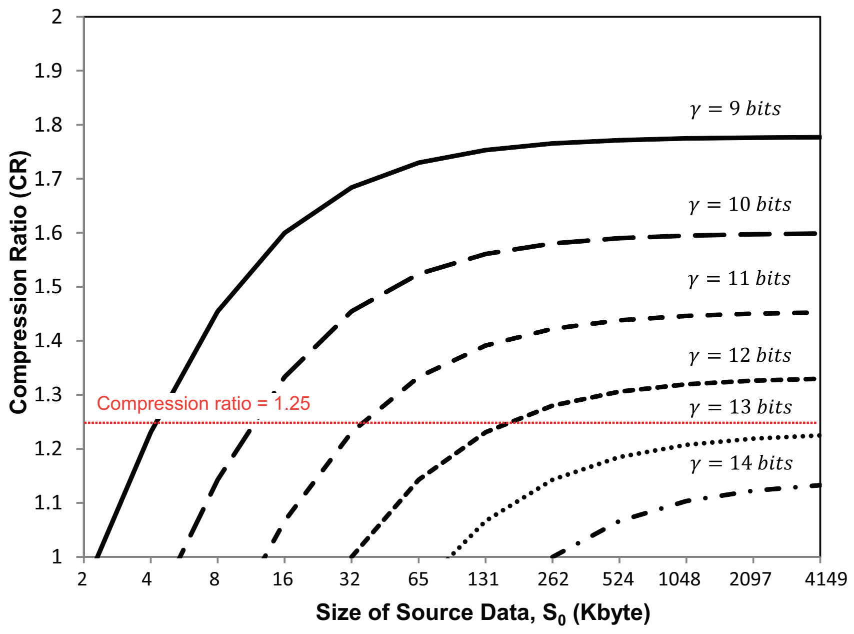


Figure 6.6: The upper bound for the proposed compression algorithm

6.3.2 Codeword Length

The codeword length, l_c is used to allow variable length of the codeword string. The longer of the codeword string, the more data bits compressed per symbol. However, the time needed to determine the possible codeword will be higher. Example of CR for fixed codeword length and variable codeword length are showed in Fig. 6.7(a) and Fig. 6.7(b) respectively. The codeword length showed in Fig. 6.7(b) are varied between the l_{cmax} and l_{cmin}.

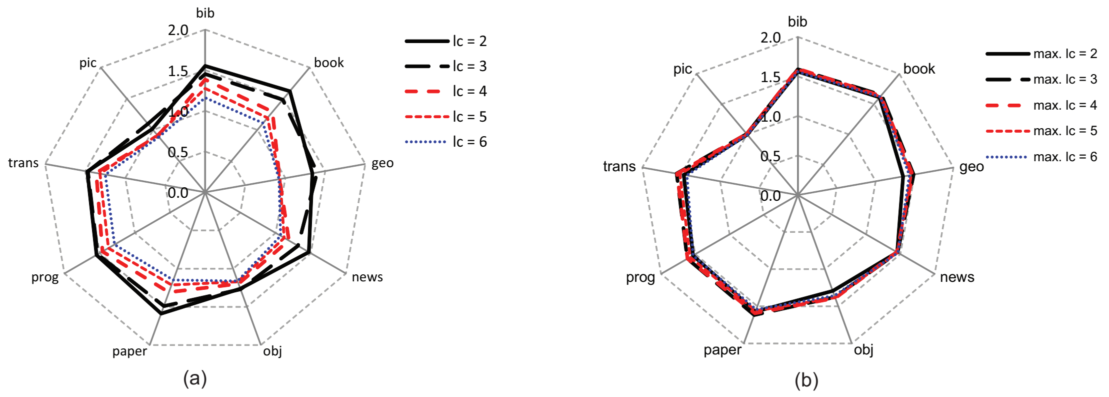


Figure 6.7: The example of CR with fixed and variable codeword length respectively

The test samples are from the benchmarks of Canterbury corpus [123] and Silesia corpus [124]. CR for book is the average of book1 and book2, obj is the average of obj1 and obj2, paper is the average of paper1 and paper2, and prog is the average of prog1, prog2, and prog3. From Fig. 6.7 (a), $l_c = 2$ gives better compression performance than the others. This is because when the l_c increases, the probability of false matching with the code also increases. In the variable codeword length, CR for all the test samples are over than 1. When l_c increases, the compression will become better, because the algorithm will try to search for the longest codeword to encode. However, this will take longer processing time. Meanwhile, $l_c = 2$ for fixed codeword length is nearly as the variable codeword length with $l_{cmax} = 6$. This example is just to show the effect of fixed and variable codeword length of LDC for the test samples used; they cannot be used to represent any general case.

CR in variable codeword length is better than with fixed codeword length. Difference in CR with variable codeword length is not obvious for small codeword length but still extremely time consuming. Fixed codeword length will always have the fixed code word (2) as the upper CR limit for any higher codeword length. On the other hand, the variable codeword length will only have code word (2) as the lower limit. Any variable codeword length will provide higher CR than the best corresponding fixed codeword length.

6.3.3 Block Size

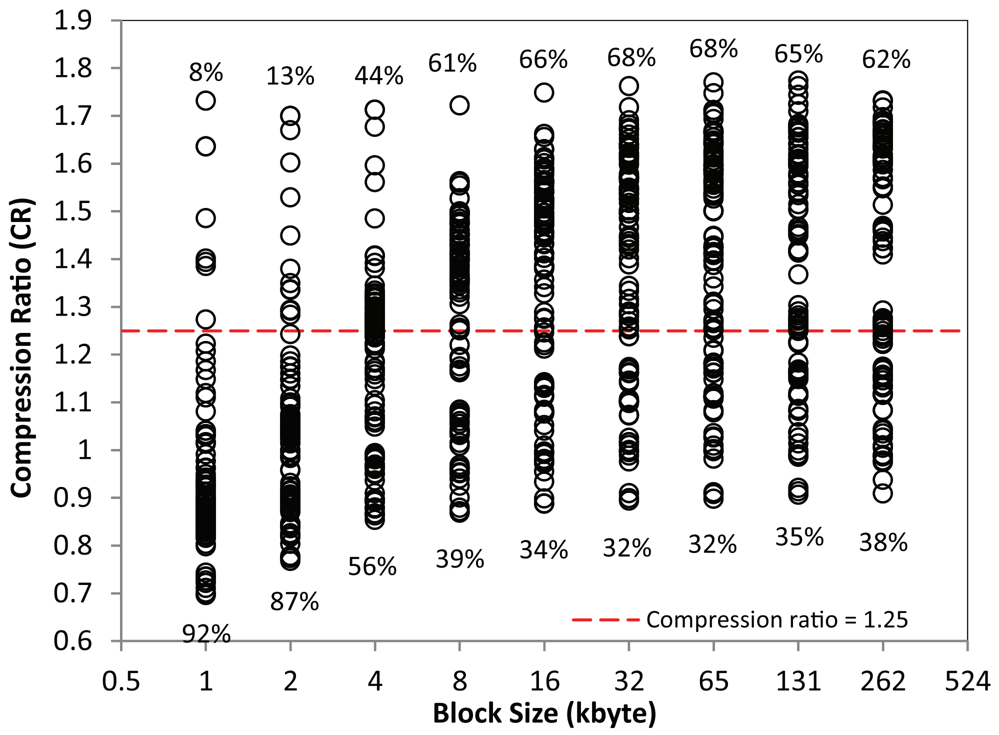


Figure 6.8: CR of 100 test samples with $\gamma = 9$, $l_c = 2$ in various S_o size

CR of different source data size is calculated to determine the optimal block size for the proposed compression algorithm. 100 benchmark samples with 50% binary format such as mp4, jpeg, and wmv, and 50% text format such as txt, html, and .c are tested. The result is shown in Fig. 6.8. $CR = 1.25$ is used as a reference that needs to be achieved for minimum CR. In this case, compressing each of block data can at least save 20% of the network bandwidth occupied by that data and can help the overall compression of the whole network.

From Fig. 6.8, when the source data is $8KB$, 61% of the test samples give CR with more than 1.25. When the block size is $32KB$ and $64KB$, 68% of the test samples can compress more than 20% of the original file size. However, the test samples that were compressed more than 1.25 decreases when the block size is more than $128KB$.

In summary, the chosen parameter values of fixed LDC for this research are

- Code length, γ : 9 bits
- Codeword length, l_c : 2
- Block size : 16 Kbyte

6.4 Performance Analysis

The overall performance of LDC is quite clear from the tables below and pretty self-explanatory.

Table 6.2: The Used Benchmarks

Test sample	Size (Kbytes)	Category
alice29	152	English text
book2	611	Non-fiction book
Ice10	427	Technical writing
paper1	53	Technical paper
news	377	USENET batch file
obj2	246	Object code
progp	49	Source code
geo	102	Geophysical data
trans	93	Transcript of terminal

Table 6.3: Dictionary Size and Compression Ratio

Test sample	Memory (Mbyte)		Compression Ratio	
	LZW	LDC	LZW	LDC
alice29	69	2	2.44	1.76
book2	72	5	2.54	1.60
Ice10	71	3	2.62	1.73
paper1	69	3	2.12	1.59
news	71	6	2.13	1.45
obj2	70	6	1.92	1.42
progp	69	2	2.57	1.69
geo	69	7	1.32	1.55
trans	69	2	2.45	1.56

The simulation conditions used to obtain the above results are listed below:
Evaluation reference: The code of LZW compression is developed by V. Antonenko [126].
Test environment: The simulations is performed using the GNU/Linux 3.13, 32-bit operating system, Intel Core 2 Duo 1.20GHz CPU, 4GB main memory, 160GB 66MHz hard disk. The code of LZW and LDC are compiled with gnu C version 4.8.1.

Data Set: Nine benchmarks from Canterbury Corpus [123] are used. The sizes and categories of these test samples are given in Table 6.2.
LZW Parameter: Code length, $\lambda = 8bits$ (7bits + 1 flag bit), Maximum Codeword length=128, dictionary size=128 and $\alpha = 7bits$.
LDC Parameters: Code length, $\lambda = 9bits$, Maximum Codeword length, $l_{cmax} = 6$, Minimum Codeword length, $l_{cmin} = 2$, dictionary size=256 and $\alpha = 8bits$. R_{thre} was used as the minimum for each codeword length.

Table 6.3 shows the additional memory needed during decoding and CR for both LZW and LDC. The additional decoder memory needed by LZW is 10 or more times that of LDC. Generally, LZW achieves better CR than LDC, except for *geo* benchmark, in which LDC is 0.25 higher than LZW.

In summary, LZW offers better CR than LDC. The important thing here is that during decoding, LZW requires larger memory compared to LDC. The classification of both ECM/C and LDC according to categories of compression schemes presented in Chapter 2 is shown in the table below with some of the compression schemes mentioned earlier as references; LZ family, L.S. Tan [125] and MSR.

Table 6.4: Classification of the proposed compression schemes

Classification Scheme	Homogeneity	Purpose	Accuracy	Structuring of data	Repetition distance	structure sharing	No. of passes	Sampling frequency	Sample size ratio
LZ	Homogeneous	Storage & Comm.	Lossless (lossy)	None	Practical Limit	Mostly No	Mostly 1 (2 or 3)	Practical Limit	100% of block
L. S. Tan	Homogeneous	Comm.	Lossless	None	Practical Limit	No	1	Practical Limit	100% of block
MSR Juniper	Homogeneous	Comm.	Lossless	None	Intra-data blocks (Infinite)	Distributed	1 + (decoupled off critical path)	N/A	N/A
LDC	Homogeneous	Storage & Comm.	Lossless	None	Whole data block	Yes	2	Whole data block	100% of block
ECM/C	Homogeneous	Comm.	Lossless	None	Whole data block	Yes	2	Whole data block	100% of block

Chapter 7

Conclusion and Future Work

7.1 Conclusion

This research started by a comprehensive review of network congestion related mechanisms and protocols including congestion free networks like congestion ATM and MPLS. Variety of mechanisms was developed for congestion vulnerable networks like TCP/IP to provide avoidance or mending of the experienced congestion situations. In-depth understanding of how to address the congestion problem was presented according to the insight gained from the mentioned review. This dissertation have investigated congestion control management mechanisms and introduced a new direction for aiding in solving network congestion. The current congestion management mechanisms are always embedded into flow management mechanism which ensure that network resources are been fully utilized all the time. The previously proposed mechanisms directly or indirectly help to eliminate the congestion problems. None of those mechanisms has been shown to be perfect in achieving full utilization of network resource with keeping safe out of congestion situations.

Since compression is the new mechanism introduced in this research to help in congestion management, an in-depth review of traditional and common compression schemes is presented. Thorough classification of compression schemes is studied and applied to the common compression schemes in use nowadays. The basic compression metrics are also discussed and extended to obtain some general reasonable ranges of any compression scheme, which were utilized in the analysis compression of this thesis.

A empirical model was derived for TCP connection throughput with compression functionality of the contents of the buffering resources. The model was used to study the feasibility compression in networks. A model has been derived mathematically from the original TCP connection throughput model and a recent variant. The curves of the model has been analysed to show the theoretical bounds of compression in networks. The analysis has shown that compression may improve throughput of heavily congested networks in some cases. When combined with memory (buffer) upgrades, compression can increase the improvement caused by buffer enlargements.

The proposed ECM framework can be applied to minimize the impact of network congestion by orchestrating a big number of already existing mechanisms that handle different situations of network congestion. In order to resolve the congestion situations much better and faster than applying any of those mechanisms individually, combinations and sequences of those mechanisms can be utilized in different situations or locations simultaneously. Thus maximizing network resources utilization by reducing the period of time where congestion mechanisms are increasing safety gaps from fully utilizing the resources. The ECM framework presented in this research offers an overall idea of how to utilize the exiting congestion management mechanism to improve congested networks throughput. Additionally the framework introduces compression over MPLS as new mechanism for helping in congestion situations, especially heavily congested slow networks. MPLS-TE was one of the tool coordinated by the framework and was selected to be the carrier of the compression new mechanism to handle some compression related issues. ECM framework is a general framework that can be corporate with a lot more mechanisms for handling network congestion. It can be extended with any newly emerging mechanism whether originally designed for congestion management or can aid in the process. For example, the commercial Juinper hardware compression was plugged in the framework with some guiding foresight on how to do such integration for other mechanism and tools. Even when the commercial product offered little details due to market privacy, still it was shown how it can be used while offering promising contribution in the proposed ECM framework.

ECM/C model was also presented in this research to offer an overall idea of how to feasible compression mechanism for limited resources devices. Small devices as actuator with slow connections, small buffering capacity and limited computational resources cannot afford to run any of the resource hungry compression schemes. ECM/C model was designed to address the needs of those small devices. A new simple compression scheme LDC was designed to be incorporated into the ECM/C model. LDC and the ECM/C model enable small devices to carry out decompression with minimal resources. By sacrificing both compression ratio and compression delay in favour, still the final transfer time would benefit from any possible compression. Since LDC requires less memory for decoding, it is the only chance for those small devices to reduce the required bandwidth compared with the normal traffic. The analysis of LDC has hinted the possible overall performance when applied in network stacks. LDC was further tuned to offer better compression performance. Even by sacrificing some compression performance, LDC can finally offer some contribution to help in congested networks of small devices.

7.2 Future Work

Dedicated research further evaluates ECM framework and ECM/C model using computer simulations, emulation and finally real testbed experiments.

Investigate the effectiveness of ECM framework and ECM/C model from the viewpoint of application, peak traffic applications (e-voting, e-shopping, etc.) can be studied to cause on-demand congestions. Evaluation of energy consumption of the framework and model presented is critical for the practicality of the usage.

More congestion management mechanisms can be incorporated into the ECM framework and studied to check which are more valuable for solving congestion situations, so that ineffective mechanisms be removed to speed up the framework. Although the MSR compression mechanism used in the framework, mandates the usage of MPLS, other similar protocols or tools can be investigated such as ATM.

The dictionary management in both MSR and ECM/C can be further studied to improve the additional traffic or reach better compromise for synchronization traffic. The size and structure of the dictionaries are also a big research topic to reach a trade-off for better compression versus additional resources being of temporary nature.

Other networking performance metrics calculation models or more general one can also give a better idea on the effect of the different congestion management mechanisms including compression. The development of such models, although quite complicated, can settle once and for all the effectiveness of the different mechanisms in solving congestions and improving the performance of networks.

Bibliography

[1] A.S. Tanenbaum and D.J. Wetherall, “Introduction,” in Computer networks, 5th ed. Prentice Hall, 2011, ch.1, pp.2.

[2] Internet Society, “This is your Internet: Trends and Growth,” Internet society global Internet report 2014.

[3] IEEE Computer Society, “IEEE 802.3,” IEEE Standards Association, http://standards.ieee.org, accessed Jun.20.2014.

[4] IEEE Computer Society, “IEEE 802.11,” IEEE Standards Association, http://standards.ieee.org, accessed Jun.20.2014.

[5] D.E. McDysan and D.L. Spohn, ATM: theory and application, (McGraw-Hill, 1995).

[6] J.Y.L. Boudec, “The asynchronous transfer mode: a tutorial,” Trans. Comp. Networks and ISDN Syst., vol.24, no.4, pp.279–309, May 1992.

[7] ATM Forum, “Broadband forum,” Broadband forum, http://www.broadband-forum.org, accessed Apr.16.2015.

[8] S. Kasera, ATM networks, (McGraw Hill Professional, 2006).

[9] M. Giroux and S. Ganti, Quality of service in ATM networks: state of the art traffic management, (Prentice Hall PTR, Upper Saddle River, 1999).

[10] H.G. Perros, An Introduction to ATM networks, (John Wiley & Sons, 2002).

[11] C.Q. Yang and A.V.S. Reddy, “A taxonomy for congestion control algorithms in packet switching networks,” IEEE Network, vol.9, no.4, pp.34–45, Jul/Aug 1995.

[12] A. Demes, S. Keshav, and S. Sheker, “Analysis and simulation of a fair queuing algorithm,” ACM SIGCOMM Comp. Commun. Review, vol.19, no.4, pp.1–12, Sept 1989.

[13] U. Mukherji, “A schedule-based approach for flow control in data communication networks,” PhD. Thesis, Massachusetts Inst. of Tech., Feb 1986.

[14] L. Zhang, “VirtualClock: A new traffic control algorithm for packet switching networks,” ACM SIGCOMM Comp. Commun. Review, vol.20, no.4, pp.19-29, Sept 1990.

[15] S. Lam and M. Reiser, “Congestion control of store-andâĂŞforward network by input buffer limits analysis,” IEEE Trans. on Commun., vol.27, no.1, pp.127–134, Jan 1979.

[16] S. Gelostani, "Congestion-free communications in high speed packet network," IEEE Trans. on Commun., vol.39, no.12, Dec 1991.

[17] D. Davies, "The control of congestion in packet switch networks," IEEE Trans. on Commun., vol.20, no.3, pp.546–550, Jun 1972.

[18] A. Tanenbaum, Computer Networks, (Prentice-Hall, Englewood Cliffs, NJ, 1981).

[19] N. Yin, S. Li, and T. Stern, "Congestion control for packet voice by selective packet discarding," IEEE Trans. on Commun., vol.38, no.5, pp.674–683, May 1990.

[20] K. Ramakrishnan and R. Jain, "A binary feedback scheme for congestion avoidance in computer networks," ACM Trans. on Computer Syst., vol.8, no.2, pp.158–181, May 1990.

[21] K. Ramakrishnan, R. Jain, and D. Chiu, "A selective binary feedback scheme for general topologies," Networks, Tech. Report, DEC-TR-510, Digital Equipment Corporation, 1991.

[22] J. Robinson, D. Friedman, and M. Steenstrup, "Congestion control in BBN packet switched networks," ACM Computer Commun. Review, vol.20, no.1, pp.76–90, Jan 1990.

[23] Z. Haas, "Adaptive admission congestion control," ACM Computer Commun. Review, vol.21, no.5, pp.58–76, 1991.

[24] O. Rose, "The Q-bit scheme," ACM Computer Commun. Review, vol.22, no. 2, pp.29-42, Apr 1992

[25] C.L. Williamson, "Optimizing file transfer response time using loss-load curve congestion control mechanism," ACM Computer Commun. Review, vol.23, no.4, pp.117–126, Oct 1993.

[26] P. Misha and H. Kanakia, "A hop by hop rate-based congestion control scheme," ACM Computer Commun. Review, vol.22, no.4, pp.112–123, Oct 1992.

[27] R. Varakulsiripunth, N. Shiratori, and S. Noguchi, "A congestion-control policy on the internetwork gateway," Computer Networks and ISDN Systems, pp.43–58, 1986.

[28] D. Commer and R. Yavatkar, "A rate-based congestion avoidance and control scheme for packet switched networks," Proc. of 10th IEEE Int. Conf. on Distributed Comput. Syst. (ICDCS), pp.390–397, 1990.

[29] A. Mukherjee, L. Landwebber, and T. Faber, "Dynamic time windows: packet admission control with feedback," ACM Computer Commun. Review, vol.22, no.4, pp.124–135, Oct 1992.

[30] V. Jacobson, "Congestion avoidance and control," ACM Computer Commun. Review, vol.25, no.1, pp.157–187, Jan 1995.

[31] R. Jain, "A timeout-based congestion control scheme for window flow controlled networks," IEEE J. on Selected Area in Commun., vol.4, no.7, pp.1162–1167, Oct 1986.

[32] Z. Wang and J. Crowcroft, “A new congestion control scheme: slow start and search [Tri-S],” ACM Computer Commun. Review, vol.21, no.1, pp.32–43, Jan 1991.

[33] K. Park, “Warp control: a dynamically stable congestion protocol and analysis,” ACM Computer Commun. Review, vol.23, no.4, pp.137–147, Oct 1993.

[34] J.F. Kurose and K.W. Ross, Computer networking: a top-down approach, (Peason, 6th edition, 2013)

[35] X.Y. Wang and D. Perkins, “Cross-layer hop-by-hop congestion control in mobile ad hoc networks,” Proc. of IEEE Wireless Commun. and Netw., Las Vegas, USA, Apr 2008.

[36] P. Reena and L. Jacob, “Hop-by-hop versus end-to-end congestion control in wireless multi-hop UWB networks,” Proc. Int. Conf. on Advanced Comput. and Commun., Guwahati, India, pp.255–261, Dec 2007.

[37] P.P. Mishra and H. Kanakia, “A hop by hop rate based congestion control scheme,” Proc. of ACM Sigcomm, New York, USA, pp.112–123, Aug 1992.

[38] L. Tassiulas, “Adaptive back-pressure congestion control based on local information,” IEEE Trans. on Automatic Control, vol.40, no.2, pp.236–250, Feb 1995.

[39] Y. Yi and S. Shakkottai, “Hop-by-hop congestion control over a wireless multi-hop network,” IEEE/ACM Trans. Netw., vol.15, no.1, pp.133–144, Feb 2007.

[40] R. Adams, “Active queue management: A survey,” IEEE Commun. Surveys & Tutorials, vol.15, no.3, pp.1425–1476, Jul 2013.

[41] C.Y. Wan, S.B. Eiseaman, and A.T. Champbell, “CODA: Congestion detection and avoidance,” Proc. ACM Conf. on Embedded Networked Sensor Syst., Los Angeles, USA pp.266–279, Nov 2003.

[42] Q. Pang, V.W.S. Wong, and V.C.M. Leung, “Reliable data transport & congestion control in wireless sensor networks,” Int. J. Sensor Netw., vol.3, no.1, pp.16–24, Jan 2008.

[43] C. Wang, K. Sohraby, B. Li, and W. Tang, “Issue of transport protocols for wireless sensor networks,” Proc. Int. Conf. on Commu., Circuits, and Syst., pp.422–426, May 2005.

[44] C. Wang, K. Shoraby, and B. Li, “SenTCP: A hop-by-hop congestion control protocol for wireless sensor networks,” Proc. of IEEE Computer and Commun. Societies (INFOCOM), Miami, USA, Mar 2005.

[45] C. Sergiou, V. Vassiliou, and A. Pitsillides, “Reliable data transmission in event-based sensor networks during overload situation,” Proc. of Int. Conf. Wireless Internet, Austin, Texas, pp.1–8, Oct 2007.

[46] O.B. Akan, and I.F. Akyildiz, “Event-to-sink reliable transport in wireless sensor networks,” IEEE/ACM Trans. on Netw., vol.13, no.5, pp.1003-1016, Oct 2005.

[47] J. Postel, “Transmission control protocol,” RFC793, Sept 1981.

[48] J. Nagle, “Congestion control in IP/TCP internetworks,” RFC896, Jan 1984.

[49] J.K. Sundararajan, D. Shah, M. Medard, S. Jakubczak, M. Mitzenmacher, and J. Barros, “Network coding meets TCP: Theory and implementation,” Proc. of IEEE, vol.99, no.3, pp.490–512, Mar 2011.

[50] S. Vazhkudai, J. Wu, and I. Foster, “Predicting the performance of wide area data transfer,” Proc. of IEEE Int. Parallel and Distributed Processing Symp., Ft. Lauderale, FL, Apr 2002.

[51] S.M.H. Shah, A.U. Rehman, A.N. Khan, and M.A. Shah, “TCP throughput estimation: A new neural networks model,” Proc. Int. Conf. on Emerging Tech., Islamabad, pp.94–98, Nov 2007.

[52] L.A. Grieco and S. Mascolo, “Performance evaluation and comparison of Westwood, New Reno, and Vegas TCP congestion control,” ACM SIGCOMM Comp. Commun. Review, vol.34, no.2, pp.25–38, Apr 2004.

[53] H. Jamal and K. Sultan, “Performance analysis of TCP congestion control algorithm,” Int. J. of Computer and Commun., vol.2, no.1, pp.30–38, 2008.

[54] A.A. Hanbali, E. Altman, and P. Nain, “A survey of TCP over ad hoc network,” IEEE Commun. Surveys and Tutorials, vol.7, no.3, pp.22–36, Jun 2005.

[55] H. Balakrishnan, V.N. Padmanabhan, S. Seshan, and R.H. Katsz, “ A comparison of mechanisms for improving TCP performance over wireless link,” IEEE/ACM Trans. Netw., vol.5, no.6, pp.756–769, Dec 1997.

[56] K.S. Reddy and L.C. Reddy, “A survey on congestion control protocols for high speed network,” Int. J. of Comp. Sci. and Network Security, vol.8, no.7, pp.44–53, Jul 2008.

[57] A. Afanasyev, N. Tilley, P. Reiher, and L. Kleinrock, “Host-to-host congestion control for TCP,” IEEE Commun. Surveys and Tutorials, vol.12, no.3, pp.304–342, May 2010.

[58] Z. Wang and J. Crowcroft, “Eliminating periodic packet losses in 4.3 - Tahoe BSD TCP congestion control,” ACM Comp. Commun. Rev., vol.22, no.2, pp.9–16, Apr 1992.

[59] L.S. Brakmo and L.L. Peterson, “TCP Vegas: end-to-end congestion avoidance on a global Internet,” IEEE J. Sel. Areas in Commun., vol.13, no.8, pp.1465–1480, Aug 2002.

[60] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanov, “TCP selective acknowledgment options,” RFC2018, Oct. 1996.

[61] V. Paxson, “End-to-end Internet packet dynamics,” ACM SIGCOMM, vol.27, no.4, pp.139–152, Oct 1997.

[62] S. Floyd, T. Henderson, and A. Gurtov, “The NewReno modification to TCP’s fast recovery algorithm,” RFC3782, Apr 2004.

[63] S. Floyd, J. Mahdavi, M. Mathis, and M. Podolsky, "An extension to the selective acknowledgement (SACK)," RFC2883, Jul 2000.

[64] S. Mascolo, C. Casetti, M. Gerla, M. Y. Sanadidi, and R. Wang, "TCP Westwood: Bandwidth estimation for enhanced transport over wireless links," Proc. of ACM MOBICOM, Rome, Italy, pp.287–297, Jul 2001.

[65] C.P. Fu and S.C. Liew, "TCP Veno: TCP enhancement for transmission over wireless access networks," IEEE J. Sel. Areas Commun., vol.21, no.2, pp.216–228, Feb 2003.

[66] R. Wang, M. Valla, M.Y. Sanadidi, B. Ng, and M. Gerla, "Efficiency/friendliness tradeoffs in TCP Westwood," Proc. of 7th IEEE Symp. on Computer and Commun., Taormina, Italy, pp.304–311, Jul 2002.

[67] A. Venkataramani, R. Kokku, and M. Dahlin, "TCP Nice: a mechanism for background transfers," Operating Syst. Rev., vol.36, pp.329–344, Dec 2002.

[68] A. Kuzmanovic and E.W. Knightly, "TCP-LP: Low-priority service via end-point congestion control," IEEE/ACM Trans. Netw., vol.14, no.4, pp.739–752, Aug 2006.

[69] V. Tsaoussidis and C. Zhang, "TCP-Real: Receiver-oriented congestion control," J. of Computer Netw., vol.40, no.4, pp.477–497, Nov 2002.

[70] F. Wang and Y. Zhang, "Improving TCP performance over mobile adhoc networks with out-of-order detection and response," Proc. of ACM Int. Symp. Mobile Ad Hoc Network and Computer, New York, USA, pp.217–225, Jun 2002.

[71] G. Yang, R. Wang, M. Sanadidi, and M. Gerla, "TCPW with bulk repeat in next generation wireless networks," Proc. IEEE Int. Conf. on Commun., Anchorage, USA, vol.1, pp.674–678, May 2003.

[72] T. Kelly, "Scalable TCP: improving performance in highspeed wide area networks," ACM SIGCOMM Comp. Commun. Rev., vol.32, no.2, pp.83–91, Apr 2003.

[73] S. Floyd, "HighSpeed TCP for large congestion windows," RFC3649, Dec 2003.

[74] C. Jin, D. Wei, S. Low, J. Bunn, H. Choe, J. Doyle, H. Newman, S. Ravot, S. Singh, and F. Paganini, "FAST TCP: from theory to experiments," IEEE Netw., vol.19, no.1, pp.4–11, Jan 2005.

[75] M. Handley, S. Floyd, J. Padhya, and J. Widmer, "TCP friendly rate control (TFRC): Protocol specification," RFC5348, Sept 2003

[76] L. Xu, K. Harfoush, and I. Rhee, "Binary increase congestion control (BIC) for fast long-distance networks," Proc. of IEEE Computer and Commun. Societies (INFOCOM), Hong Kong, vol.4, pp.2514–2524, Mar 2004.

[77] C. Caini and R. Firrincieli, "TCP Hybla: A TCP enhancement for heterogeneous networks," Int. J. Satellite Commun. and Netw., vol.22, pp.547–566, Sept 2004.

[78] K. Srijith, L. Jacob, and A. Ananda, "TCP Vegas-A: Improving the performance of TCP Vegas," J. Computer Commun., vol.28, no.4, pp.429–440, Mar 2005.

[79] S. Biaz and N.H. Vaidya, "Differentiated service: A new direction for distinguishing congestion losses from wireless losses," Tech. Rep., Auburn University, pp.1–15, Feb 2003.

[80] J. Sing and B. Soh, "TCP New Vegas: improving the performance of TCP Vegas over high latency links," Proc. IEEE Netw. Comp. and Applications, Cambridge, MA, pp.73–80, Jul 2005.

[81] R. King, R. Baraniuk, and R. Riedi, "TCP-Africa: An adaptive and fair rapid increase rule for scalable TCP," Proc. of IEEE Computer and Commun. Societies (INFOCOM), Miami, USA, vol.3, pp.1838–1848, Mar 2005.

[82] K. Tan, J. Song, Q. Zhang, and M. Sridharan, "A compound TCP approach for high-speed and long distance networks," Proc. of IEEE Computer and Commun. Societies (INFOCOM), Barcelona, Spain, pp.1–12, Apr 2006.

[83] Y. Shu, W. Ge, N. Jiang, Y. Kang, and J. Luo, "Mobile host centric transport protocol for EAST experiment," Proc. of 15th IEEE-NPSS Real Time Conf., Betavia, pp.1–8, Apr 2007.

[84] S. Liu, T. Basar, and R. Srikant, "TCP-Illinois: A loss and delay-based congestion control algorithm for high-speed networks," J. Performance Evaluation, vol.65, no.6-7, pp.417–440, Jun 2008.

[85] K. Kaneko, T. Fujikawa, Z. Su, and J. Katto, "TCP-Fusion: A hybrid congestion control algorithm for high-speed networks," Proc. PFLDnet, Los Angeles, USA, pp.31–36, Feb 2007.

[86] I. Rhee and L. Xu, "CUBIC: A new TCP-friendly high-speed TCP variant," ACM SIGOPS Operating Syst. Rev., vol.42, no.5, pp.64–74, Jul 2008.

[87] G. Marfia, C. Palazzi, G. Pau, M. Gerla, M. Sanadidi, and M. Roccetti, "TCP Libra: Deriavation, analysis, and comparison with other RTT-fair TCPs," J. of Computer Netw., vol.54, no.14, pp.2327–2344, Oct 2010.

[88] J. Wang, J. Wen, J. Zhang, and Y. Han, "TCP-FIT: An improved TCP congestion control algorithm and its performance," Proc. of IEEE Computer and Commun. Societies (INFOCOM), Shanghai, China, pp.2894–2902, Apr 2011.

[89] H. Wu, Z. Feng, and Y. Zhang, "ICTCP: Incast congestion control for TCP in data-center networks," IEEE/ACM Trans. on Netw., vol.21, no.2, pp.345–358, Apr 2013.

[90] Z.T. Alisa, and S.R. Qasim, "A fuzzy based TCP congestion control for wired networks," Int. J. of Comp. Appl., vol.89, no.4, pp.36–42, Mar 2014.

[91] H.Y. Hsieh, K.H. Kim, Y. Zhu, and R. Sivakumar, "A receiver centric transport protocol for mobile hosts with heterogeneous wireless interfaces," Proc. of ACM MobiCom, New York, USA, pp.1–15, Sept 2003.

[92] R. Gupta, M. Chen, S. McCanne, and J. Walrand, "A receiver-driven transport protocol for the web," Telecommun. Syst., vol.21, no.2–4, pp.213-230, Aug 2002.

[93] P. Mehra, C. Vleeschouwer and A. Zakhor, "Receiver-driven bandwidth sharing for TCP and its application to video streaming," IEEE Trans. on Multimedia, vol.7, no.4, pp.740–752, Aug 2005.

[94] N.T. Spring, M. Chesire, M. Berryman, V. Sahasranaman, T. Anderson, and B. Bershad, "Receiver based management of low bandwidth access links," Proc. of IEEE Computer and Commun. Societies (INFOCOM), Tel-Aviv, Isael, vol.1, pp.245–254, Mar 2000.

[95] K. Shi, Y. Shu, O. Yang, and J. Luo, "Receiver assistant congestion control in high speed and lossy networks," Proc. of 16th IEEE-NPSS Real Time Conf., Beijing, China, pp.91–95, May 2009.

[96] H. Sheikhi, M. Dashti, and M. Dehghan, "Congestion detection for video traffic in wireless sensor networks," Proc. Int. Conf. on Consumer Electron., Commun., and Netw., Xian Ning, China, pp.1127–1131, Apr 2011

[97] W. Graham, "Signal coding and processing," 2nd ed., Cambridge University Press, pp.34, 1994.

[98] M. Mahoney, "Data compression explained," online book, Dell Inc., 2013.

[99] D.A. Lelewer and D.S. Hirschberg, "Data compression," ACM Comput. Surv., vol.19, no.3, pp.261–296, Sept 1987.

[100] J.H. Pujar and L.M. Kadlaskar, "A new lossless method of image compression and decompression using huffman coding techniques," J. of Theoretical and Applied Info. Tech., vol.15, no.1, pp.18–23, May 2010.

[101] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," IEEE Trans. on Info. Theory, vol.23, no.3, pp.337–343, May 1977.

[102] M.K. Tank, "Implementation of Limpel-Ziv algorithm for lossless compression using VHDL," Proc. of 1st Int. Conf. on Contours of Comput. Tech., Thinkquest, Berlin, pp.275–283, 2010.

[103] D. Salomon, Data compression: the complete reference, (4th edition, Springer, 2007)

[104] S. Ferilli, Automatic digital document processing and management: problems, algorithms, and techniques, (Springer, 2011)

[105] M. Nelson and J.L. Gailly, The data compression book, (2nd edition, M & T Books, 1996)

[106] K. Sayood, Introduction to data compression, (4th edition, Elsevier, 2012)

[107] M. Banikazemi, "LZB: Data compression with bounded references," Data Compression Conf., pp.436, Mar 2009.

[108] J. Lansky and M. Zemlicka, "Compression of small text files using syllables," Proc. of Data Compression, Utah, pp.458, 2006.

[109] V. Jacobson, “Compressing TCP/IP header for low-speed serial links,” RFC1144, Feb 1990.

[110] S. Casner and V. Jacobson, “ Compressing IP/UDP/RTP header for low-speed serial links,” RFC2508, Feb 1999.

[111] M. Degermark, B. Nordgren, and S. Pink, “IP header compression,” RFC2507, Feb 1999.

[112] L.E. Jonsson, “RObust Header Compression (ROHC) terminology and channel mapping examples,” RFC3759, Apr 2004.

[113] J. Padhye, V. Firoru, D. Towsley, and J. Kurose, “Modeling TCP throughput: a simple model and its empirical validation,” ACM SIGCOMM Comp. Commun. Review, vol.28, no.4, pp.303–314, Oct 1998.

[114] S.M. Ross, “Expectation of a random variable,” in Introduction to probability models, 9th ed. Academic Press, ch.2, sec.4, pp.38, 2007.

[115] R.W. Hamming, “Random variables, mean and the expected value,” in The art of probability for scientists and engineers, Addison-Wesley, ch.2, sec.5, pp.64, 1991.

[116] H. Hisamatsu, H. Ohsaki, and M. Murata, “Modeling a heterogeneous network with TCP connections using fluid flow approximation and queuing theory,” Proc. of Int. Society for Optical Eng., Jan 2003.

[117] T. Poka-anon and S. Pattaramalai, “Throughput efficiency by employing a relay node at different distances in wimax cell,” Int. J. of Comput., Commun. & Instrum. Eng. (IJCCIE), vol.1, no.1, pp.25–28, 2014.

[118] Juniper Network, (2006) Analyzing the molecular sequence reduction (MSR) technology, Juniper Network Incorporated, 1194 North Mathilda Avenue, Sunnyvale, USA, from http://forums.juniper.net/jnet/attachments/jnet/wx/19/1/

[119] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, “Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification,” RFC2205, Sept 1997

[120] Sandvine Intelligent Broadband Networks,(2014) Global Internet phenomena report, Sandvine Incorporated ULC, Ontario, Canada, from https://www.sandvine.com/trends/global-internet-phenomena/

[121] A. Beirami and F. Fekri, “Results on the redundancy of universal compression for finite-length sequences,” Proc. of IEEE Int. Symp. Info. Theory, pp.1504–1508, Aug 2011.

[122] S.T. Klein and D. Shapira, “Improved variable to fixed length codes,” Lecture Notes in Comp. Sci., vol.5280, pp.39-50, 2009.

[123] M. Powell, “The canterbury corpus,” *http : //corpus.canterbury.ac.nz/descriptions /#cantrbry*, access on Jun 2014.

[124] Silesian University of Technology, “Silesia compression corpus,” $http : //sun.aei.polsl.pl/\ sdeor/index.php?page = silesia$, access on Jun 2014.

[125] L.S. Tan, S.P. Lau, and C.E. Tan, “Quality of service enhancement via compression technique for congested low bandwidth network," Proc. of Int. Conf. on Commun. (MICC), Sabah, Malaysia, pp.71–76, 2011.

[126] V. Antonenko, “Implementation of LZW compression algorithm in C," $https : //code.google.com/p/clzw/source/browse/src/lzw - enc.c?name = 693$ $72a470e\&r = ace0a939d395d42ac6c16c003fe3c0f5caefda10$, access on Aug 2014

Publications

Journal

[1] Lee Chin Kho, Xavier Defago, Yasuo Tan, and Yuto Lim, "A survey of congestion detection and avoidance for TCP congestion control," J. of Theoretical and Applied Info. Tech. (JATIT), vol.75, no.1, May 2015.

[2] Lee Chin Kho, Yasuo Tan, and Yuto Lim, "Efficient congestion management framework using compression techniques," IEICE Trans. on Commun. (To be submitted)

[3] Tran Phuong Thao, Lee Chin Kho, Azman Osman Lim, and Kazumasa Omote, "SWC-POR: Slepian-wolf coding-based POR scheme for cloud storage," IEICE Trans. on Fundamentals of Electron., Commun. and Computer Sci. (Submitted)

International Conferences

[4] Lee Chin Kho, Xavier Defago, Azman Osman Lim, and Yasuo Tan, "A taxonomy of congestion control techniques for TCP in wired and wireless networks," IEEE Symp. on Wireless Tech. and Appl. (ISWTA), Kuching, Malaysia, pp.147-152, Sept. 2013.

[5] Lee Chin Kho, Sze Song Ngu, Yasuo Tan, and Azman Osman Lim, "Applications of data compression technique in congested network," Int. Conf. on Wireless Commun., Netw. and Appl. (WCNA), Shenzhen, China, Dec. 2014.

[6] Lee Chin Kho, Yasuo Tan, and Yuto Lim, "Joint LZW and lossless dictionary-based bit packing compression techniques for congested network," The 2^{nd} Int. Conf. on Computer, Commun., and Control Tech. (I4CT), Kuching, Malaysia, Apr. 2015.

[7] Tran Phuong Thao, Lee Chin Kho, and Azman Osman Lim, "SW-POR: A novel POR scheme using slepian-wolf coding for cloud storage," The 11^{th} IEEE Int. Conf. on Autonomic and Trusted Comput. (ATC), Dec. 2014.

Domestic Conferences

[8] Lee Chin Kho, Yasuo Tan, and Azman Osman Lim, "Study of compression techniques for congestion control in wireless networks," The 67^{th} IPSJ SIG Technical Report, Tokyo, Japan, Sept. 2013.

[9] Lee Chin Kho, Yasuo Tan, and Azman Osman Lim, “Adaptive congestion control scheme using network coding and compression techniques for wireless networks,” IEICE Society Conf., Fukuoka, Japan, Sept. 2013.

[10] Lee Chin Kho, Yasuo Tan, and Azman Osman Lim, “Feedback-based congestion control gateway router in home M2M networks,” Malaysia-Japan Academic Scholar Seminar, Tokyo, Japan, Nov. 2012.

[11] Lee Chin Kho, Yasuo Tan, and Azman Osman Lim, “Feedback-based congestion control gateway router in Home M2M networks,” IEICE society conf., Toyama, Japan, Sept. 2012.

[12] Lee Chin Kho, Yasuo Tan, and Azman Osman Lim, “Does the network coding technique work well for congestion control in wireless sensor networks?” IEICE Technical Report on Ubiquitous and Sensor Networks (USN), Aichi, Japan, vol.112, no.31, pp.7-12, May 2012.