Department of Communications and Networking

Resource Assignments in Network Data Transport

Gautam Raj Moktan

Aalto University

DOCTORAL DISSERTATIONS

Aalto University publication series
DOCTORAL DISSERTATIONS 115/2020

Resource Assignments in Network Data Transport

Gautam Raj Moktan

A doctoral dissertation completed for the degree of Doctor of Science (Technology) to be defended, with the permission of the Aalto University School of Electrical Engineering, at a public examination held at the lecture hall AS1 of the school on 11 September 2020 at 12.

Aalto University
School of Electrical Engineering
Department of Communications and Networking

Supervising professor
Professor Jukka Manner, Aalto University, FInland

Thesis advisor
Professor Jukka Manner, Aalto University, FInland

Preliminary examiners
Professor Jussi Kangasharju, University of Helsinki, Finland
Professor Anna Brunström, Karlstad University, Sweden

Opponent
Professor Tommi Mikkonen, University of Helsinki, Finland

Aalto University publication series
DOCTORAL DISSERTATIONS 115/2020

ISBN 978-952-60-3982-4 (printed)
ISBN 978-952-60-3983-1 (pdf)
ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)
http://urn.fi/URN:ISBN:978-952-60-3983-1

Unigrafia Oy
Helsinki 2020

Finland

Printed matter
4041-0619

Aalto University

Abstract

Aalto University, P.O. Box 11000, FI-00076 Aalto www.aalto.fi

Author
Gautam Raj Moktan

Name of the doctoral dissertation
Resource Assignments in Network Data Transport

Publisher School of Electrical Engineering

Unit Department of Communications and Networking

Series Aalto University publication series DOCTORAL DISSERTATIONS 115/2020

Field of research Networking technology

Manuscript submitted 9 December 2019 **Date of the defence** 11 September 2020

Permission for public defence granted (date) 12 March 2020 **Language** English

☐ **Monograph** ☒ **Article dissertation** ☐ **Essay dissertation**

Abstract

Communication networks have come a long way since their invention and adoption in the modern society. They have fundamentally changed how society functions and the speed at which activities are conducted. Colossal amount of data is transported through today's networks to convey information between various end points. The Internet is the largest information and communication network shared by billions of end points as a common medium of data transfer today.

The amount of data that can be transported through these networks within a given time period, also known as bandwidth, is the principal resource shared between the different end points. And, as is the case with many other finite resources available to modern society, scarcity of resources has led to the development of various schemes of resource distribution.

The Transmission Control Protocol (TCP) is the dominant mechanism for transporting data between end points in the Internet. It provides various functionalities such as ensuring that the data is transported reliably between the end points and preventing the receiver and the network from being overwhelmed by too much data sent. TCP implements congestion control mechanisms so that multiple flows can share the network capacity.

This thesis analyses the sharing of network resources in communication networks, using a customized TCP variant. Traditionally, TCP treats all flows as equal regardless of their end utility and attempts to provide equal share of the network bandwidth to each flow. This research analyses a scenario without that assumption. Specifically, short and long data transfers are considered unequal. As a manifestation of such concept, the research work developed a TCP congestion control variant which was able to provide different share of network capacity to different flow types. The performance of the protocol was then tested under multiple settings to investigate the notion further and evaluate the implications for deployment at large.

Solutions to the resource assignment problem merit not just technical but also socio-economic analysis. The research explores the development of different resource assignment schemes in society at large and finds analogous schemes in communication networks. By comprehending the various notions of fairness prevalent in the society, several concerns to be addressed while developing resource assignment schemes are identified.

Keywords Resource Assignment, Network Data Transport, TCP, Fairness

ISBN (printed) 978-952-60-3982-4 **ISBN (pdf)** 978-952-60-3983-1

ISSN (printed) 1799-4934 **ISSN (pdf)** 1799-4942

Location of publisher Helsinki **Location of printing** Helsinki **Year** 2020

Pages 132 **urn** http://urn.fi/URN:ISBN:978-952-60-3983-1

Preface

This thesis is the outcome of my research and learning at the Department of Communications and Networking, Aalto University, Helsinki, Finland.

It has been my privilege to have Professor Jukka Manner as my supervisor and I would like to thank him for providing the opportunity to pursue my doctoral research in this area. His guidance, support, patience and encouragement have been very valuable during the whole process. Dr. Sebastian Sonntag inspired me to venture into this area and I owe him tremendous gratitude. Having Dr. Nuutti Varis as my instructor was one of the turning point of this process and I would like to thank him for the insightful reviews of my works. Discussions with him on various topics of the research helped sort out the work and eventually finish it. I also would like to express my gratitude towards my other co-authors; Dr. Lennart Schulte, Dr. Mikko Särelä and Saurabh Pokhrel for their valuable inputs into the research. Collaboration with them was very helpful in shaping this research.

I would like to thank my pre-examiners Professor Anna Brunstrom and Professor Jussi Kangasharju for their valuable feedback. Their comments have helped improve the quality of the thesis. Special thank you goes to Professor Tommi Mikkonen for accepting to examine my thesis as an opponent.

Many thank you to all my colleagues and friends at the department during the research period. Dr. Prajwal Osti, Dr. Pramod Jacob, Arseny Kurnikov, Antti Jaakkola, Dr. Timo Kiravuo, Dr. Le Wang have been there with me through this research process in many ways both inside and outside academia. Timi Lehtola, Victor Nässi and Markus Peuhkuri have been very helpful with various technical support for the research work. I am grateful to the department secretaries for helping me with various travel arrangements during my conference and study trips.

My research was supported by TEKES funded projects such as "Future Internet" (FI-SHOK) program and "Speedifier" Projects. I would also like to thank the Elisa HPY research foundation for their grant Support.

Dr. Prem Raj Adhikari, Dr. Sandeep Tamrakar, Dr. Subash Khanal, Dr. Roshan Chudal, Tekendra Timsina along with all my other friends have also played significant role in making this journey wonderful. I thank you all for the

camaraderie we established over the years.

I am blessed to have the everlasting love, support and blessings of my father Dhan Raj Lama and my mother Jamuna Moktan. My sincerest gratitude to my whole family. And my wife Jyoti, whose patience and love I am lucky to have, I thank you for everything, especially for bringing our beautiful daughter Kelsang to this world, to whom I dedicate this book.

Helsinki, August 5, 2020,

Gautam Raj Moktan

Contents

List of Publications

This thesis consists of an overview and of the following publications which are referred to in the text by their Roman numerals.

I G. R. Moktan, S. Siikavirta, M. Särelä and J. Manner. *Favoring the Short*. In *2012 Proc. of INFOCOM Conference on Computer Communications Workshops*, Orlando, USA, pp. 31-36, Mar. 2012. IEEE.

II G. R. Moktan, S. Pokhrel, L. Schulte, M. Särelä and J. Manner. *Performance Analysis of a Short Flow Favoring TCP*. In *2014 Proc. of International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, New Delhi, pp. 134-142, Sep. 2014. IEEE.

III G. R. Moktan, N. Varis, and J. Manner. *Utilizing Connection Usage Characteristics for Faster Web Data Transport*. *Journal of Computer Networks and Communications*, Vol. 2018, Article ID 4520312, (2018). Hindawi.

IV G. R. Moktan and J. Manner. *Distributive Justice in Network Communications*. *International Journal of Communication Networks and Distributed Systems*, Vol. 23 No. 4, (2019), pp. 499–521. Inderscience.

Author's Contribution

Publication I: "*Favoring the Short*"

This work was mainly done by the author. The idea was formulated with J. Manner and S. Siikavirta. The author developed the simulation testbeds, carried out the experiments and compiled the results. The paper was drafted under the supervision of M. Särelä and J. Manner.

Publication II: "*Performance Analysis of a Short Flow Favoring TCP*"

The author developed the Linux patch with S. Pokhrel. The author designed the experiment and the evaluation methodology. S. Pokhrel conducted the experiment under the author's supervision. The author compiled the results and drafted the paper.

Publication III: "*Utilizing Connection Usage Characteristics for Faster Web Data Transport*"

This was joint work between the authors. The experiment setup was developed by the author with N. Varis. The author conducted the experiment and compiled the results. The paper was drafted by the author with N. Varis under the supervision of J. Manner

Publication IV: "*Distributive Justice in Network Communications*"

This paper was developed by the author with J. Manner as his supervisor.

List of Abbreviations

3G	Third Generation
5G	Fifth Generation
AIMD	Additive Increase Multiplicative Decrease
AQM	Active Queue Management
BBR	Bottleneck Bandwidth and RTT
CDF	Cumulative Distribution Function
CDN	Content Delivery Network
CWND	Congestion Window
DCCP	Datagram Congestion Control Protocol
DelAck	Delayed Acknowledgment
DiffServ	Differentiated Services
DTLS	Datagram TLS
ECN	Explicit Congestion Notification
FIFO	First-In, First-Out
FLD_TCP	Flow-Length Dependent Transmission Control Protocol
HTTP	Hypertext Transfer Protocol
ICT	Information and Communications Technology
IntServ	Integrated Services
IoT	Internet of Things
IP	Internet Protocol
IW	Initial Window
LAN	Local Area Network
LEDBAT	Low Extra Delay Background Transport
LTE	Long Term Evolution
MAC	Medium Access Control

MPTCP	Multipath TCP
MSS	Maximum Segment Size
NFV	Network Functions Virtualization
ns-2	Network Simulator-2
OSI	Open Systems Interconnection
P2P	Peer-to-Peer
QoS	Quality of Service
RED	Random Early Detection
RFC	Request For Comments
RTCP	RTP Control Protocol
RTO	Retransmission Timeout
RTP	Real-Time Transport Protocol
RTT	Round-Trip Time
SACK	Selective Acknowledgment
SCTP	Stream Control Transmission Protocol
SDN	Software-Defined Networking
SRTT	Smoothed Round-Trip Time
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol

1. Introduction

As of February 2019, there are more than 180 million active websites on the Internet [1]. This demonstrates how far the concept of a "Galactic network", introduced in 1962 by J.C.R. Licklider, has come since its inception. While originally intended to share raw data among researchers, the Internet at present day is used for virtually everything from social-networking, e-mails, e-commerce, entertainment, news and telecommunications to many other services. Every day many new forms of services are being developed and launched on it.

Internet is just the name by which we know the largest communication network in human history. As any communication network, its main function is to relay information from the sender to the receiver. The Open Systems Interconnection (OSI) model [2] views communication networks as consisting of several abstraction layers, each capable of providing a set of functionalities, upon which the adjacent layers can independently communicate through a standardized interface. The Internet architecture uses a less strict abstraction, also known as the Transmission Control Protocol/Internet Protocol (TCP/IP) protocol suite, which consists of four loosely defined abstraction layers, namely, the application layer, the transport layer, the internet layer and the link layer [3].

This modular layered design provided many beneficial properties to the TCP/IP stack such as interoperability, independence and flexibility. This resulted in its proliferation as various feature developments could be undertaken at each layer as well as various services could be built independently over it. As the Internet evolved, services such as file transfers, email and the world wide web were built over the application layer. The transport layer protocols were developed to fulfill the data delivery between communicating applications. Addressing/routing problems were handled by the internet layer and various routing technologies were developed for it. Similarly developments in the link and the physical layers allowed efficient ways to move the data into various physical media.

The rapid evolution of services on the world wide web eventually cemented the Internet's fate to be the largest communication network today. Its architecture allows information in various formats to be generated in web servers that is accessible from a client host application such as a browser. The browser fetches information through the network and renders them for the users' view-

ing. The Hypertext Transfer Protocol (HTTP) [4] protocol is used to transfer the information with a client-server model between the browsers and the servers. Underneath HTTP, the transport layer takes care of transporting the information carrying data between the end points. Similarly, other applications also use the same underlying layers for transporting their data which can be of streaming nature or discrete file transfers.

1.1 Network Data Transport

The transport layer provides end-to-end services to communicating applications in areas such as reliability, flow control, congestion control, multiplexing, etc. Different transport protocols exist at this layer that are able to provide certain sets of such services. Protocols are taken into use by the application layer depending on the requirements. In some cases, a transport layer service (such as reliability or congestion control) can be built into the application layer. Hence, such service is not required of the transport layer.

Transport protocols can either work in connection-oriented or connectionless mode. Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) [3] are the most common protocols for each respective modes. TCP can provide reliable and ordered bidirectional data transfer between the endpoints along with flow control and congestion control. UDP is unidirectional and provides no such features but, on the other hand, it is light weight and suitable for real-time services. Stream Control Transmission Protocol (SCTP) [5] is a connection oriented protocol that allows for multi-homing support for data transport although Multipath TCP (MPTCP) [6] also allows TCP to support multi-homing. Reliability and ordering of delivered packets can be optional in SCTP and its congestion control features are like that of TCP. Datagram Congestion Control Protocol (DCCP) [7] on the other hand provides congestion control without reliability and ordered delivery.

Similarly, for various applications, additional services are provided on top of the transport protocol by protocols that can be considered pseudo-transport protocols. Security features can be provided to the data transport by using Transport Layer Security (TLS) [8] or Datagram TLS (DTLS) [9] whereby the data is encrypted during transport. For real-time applications, Real-Time Transport Protocol (RTP) in association with RTP Control Protocol (RTCP) [10] provide support for connection setup, rate control and performance metric reporting among others.

HTTP [4] is a protocol to transfer primarily discrete objects such as files over the Internet and it provides various services to the application that use it. It allows applications to exchange data in different formats and encodings. It also uses lower layers protocols to provide their services to the application layer (such as security with TLS). Using persistent connection (HTTP 1.1 [11] and HTTP 2.0 [12]) it allows multiple objects transfers to be done with the same connection and

avoid connection setup delays. HTTP 2.0 also allows multiplexed transfers of the objects in the same connection. These methods help improve the data transfer speeds for the applications on the world wide web. HTTP 3.0 [13] over QUIC is another alternative to the combination of TCP, TLS and HTTP 2.0. Originally coming out of Google's initiative to improve Internet latency [14], it utilizes UDP instead of TCP on the transport layer which leads to improved connection setup and enables better support for multi-streaming. In a distinct paradigm shift, it pushes for features such as congestion control and error recovery to be implemented in the user space.

Applications on the Internet have at their disposal many of the above transport protocols when determining a suitable protocol for them. Thus, the Internet paths are shared by data traffic using many transport protocols. In the next section, we describe the motivation and research problem addressed in this thesis.

1.2 Motivation and Research Problem

Improved data transport not only provides better user experiences for the services using it, but also helps to cut back on the amount of energy spent moving the bits on todays networks. Quality of Service (QoS) approaches at the Internet Protocol (IP) layer such as Integrated Services (IntServ) [15] and Differentiated Services (DiffServ) [16] have been developed to allow for services to attain various amounts of network resources for their operations.

The objective of this research is to examine the network resource sharing mechanisms at the data transport layer for a more efficient data transport in terms of transfer time. And since congestion control has assumed the task of resource allocation at the transport layer, for technical implementation and analysis, the research is focused on the dominant transport protocol on the Internet, i.e. TCP and specifically on its sender side rate control mechanisms. We describe the motivations for this research in more detail in the next few subsections.

1.2.1 User Ergonomics

Many developments have been undertaken to make web usage more ergonomic and to improve user experience. While better visual designs of websites or the way a user can surf around the contents creates pleasant experiences, user's cognitive ability to perceive network delay and its effects on the derived utility also plays a significant role.

As an example, Amazon found that every 100 ms increase in the page load time would decrease sales by 1%, while for Google, a 500 ms increase in delay of search results display time would reduce revenue by 20% [17]. A survey conducted by Wichita State University shows that the frustration levels of users,

if not *lostness*, does increase as the time taken to download a page becomes longer. They found significant differences in frustration levels between the delays of 1s, 30s and 60s [18]. Another study [19] reveals that the users prefer graphical websites only if the delay is shorter. It implies that the utility of graphics on a website is directly related to the time it takes to load at the users node.

Modern day websites are graphically rich in terms of both content and advertisements. So, they need to focus on minimizing the delay. It has been seen that a users sensitivity to delay is different for transfer of text-only documents and for multimedia graphics [20].

The median size of a web page on desktop is ~1.8 MB, while on mobile it is ~1.6 MB. Median size of images is almost 1 MB and javaScripts have median size of ~400 KB [21]. More than 99% of the Internet traffic flows are short(<100 KB) [22]. Remaining traffic flows like audio, video streaming, Peer-to-Peer (P2P) file transfers, database synchronization, etc are long.

While the time taken to transfer a piece of information from a source to a user, i.e., delay, makes a valuable difference in a service utility, there are also services like audio streaming where disruptions, rather than the delays, bother the users more. Activities like movie downloads and database synchronization can tolerate delays better too. Users have higher tolerance to delay occurring in background applications. Thus, delay tolerance is subject to the nature of utility of the service.

What we can infer from these information is that most of the users' perception of delay is more evident in the shorter flows. Also most flows constituting the Internet traffic are short. Hence, prioritizing the shorter flows to the longer ones while minimizing the transfer delay is valuable to improve Internet's utility for end users.

1.2.2 Energy Savings

The field of Information and Communications Technology (ICT) is very concerned about energy conservation since it is one of the major contributor to the global energy consumption. Datacenters alone in the United States are projected to consume some 73 Billion kWh in 2020 [23]. A significant portion of that energy is spent in data transmission.

Wireless mobile devices are constantly exploring ways for longer battery time. Improvement efforts have been made with low power processors, efficient memory schemes, display techniques and power management units [24]. It has become more crucial with the Internet of Things (IoT) projecting billions of devices connecting to the Internet in the near future.

In case of wireless data transmissions, the energy cost per bit for wireless transmission is more than 1000 times than the energy cost for single computation of that bit [25]. Even when the radio is ON but in "idle-listening" mode, it still consumes the same amount of energy as when it is active [26]. The

switching on and off of the radio links also affect the energy consumed. Thus, by making data transport more efficient we can achieve energy savings in network communications.

1.2.3 Fairness

While more than 99% of the Internet traffic flows are short (<100 KB) [22], most of the bytes in the network(>78%) are contributed by long flows [27]. They both compete for the same network resources while being transported between end points.

For the TCP sessions, all flows irrespective of their lengths, obtain the same flow rate in the network. This is an artefact of the congestion control algorithm employed in TCP. Any other flow whose long term throughput does not exceed the throughput of a conformant TCP is referred to as a TCP friendly flow [28]. TCP's flow rate allocation is max-min fair (meaning that an increase of rate for any flow is at the cost of decrease of some already smaller flow rate). There have been discussions on the practicality of blindly using flow rate as fairness measure [29]. In wireless transmission, a different fairness notion is used, targeted for usage efficiency of the transmission medium. Thus, it seems that there is not a common rationale for choosing one criteria over another across the network stack.

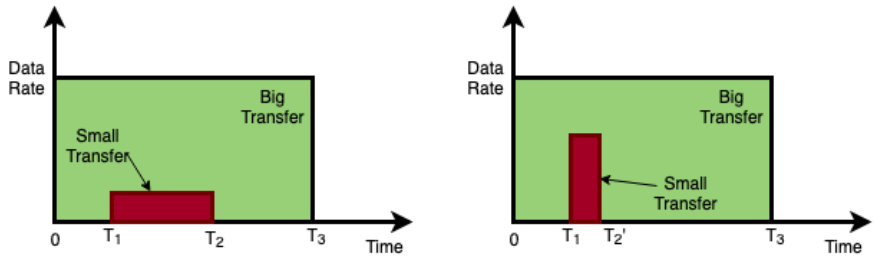


Figure 1.1. Illustration of link sharing

An example would be the case illustrated in Figure 1.1. If the small transfer (short flow), in terms of bytes, were prioritized and allowed greater share of the bandwidth, its transfer time would be significantly reduced without incurring significant (if any) delay to the large transfer. Also, giving service to the job with the shortest-remaining-processing-time (SRPT) is an optimal [30] strategy in minimizing the mean response time. As discussed earlier, a huge proportion of the Internet flows would benefit from such arrangement.

Alongside, another set of development also exist in this area that do not follow the TCP-friendly paradigm for data transport. Some discussions [31] on this front are in fact proposing that the TCP-friendly fairness criteria should be dropped. And some point out that it has no significant philosophical, social or real life basis [29]. TCP auto tuning features [32] allow end hosts to vary their share in resource allocations. Many applications are using UDP or other

proprietary protocols, and many even make use of multiple parallel connections. TCP-friendliness is slowly losing its meaning in today's Internet.

How much resources to assign to which entity and based on what parameters is an important question in many areas where resources are finite. It is even more significant in the case of the Internet, not only due to the plethora of services running on it, but also due to the diversity of its stakeholders, administrative realms and constantly changing market forces. There is no one-size-fits-all design principle in the way network resources are allocated to the applications running on it. Nevertheless, due analysis to the justification of why one notion suits over other should be included in the process of designing and administrating of communication networks. Justice and fairness are social concepts that have been studied in the socio-economic realm for a long time. Understanding how they are conveyed in the networking world is important for anyone trying to develop resource assignment schemes in the context of communication networks.

Thus, this research work embarks with multi-fold motivations. Improving perceived delay would make network services consumption more pleasing and could also provide a positive impact on service provider revenue. A holistic view of the Internet traffic patterns provides potential areas to be leveraged for instrumenting such improvements. There is a real opportunity to improve energy efficiency in the area of network communications. And furthermore, this leads us directly to face the questions of fairness and resource allocation for which we have reference models (with significant history) in the other areas of society. With these motivations, this thesis is primarily focused to the following questions:

- *Can TCP sender side congestion control mechanism be modified for differential treatment of short and long flows ? How much of a difference would such modification make for applications such as file transfers and web browsing?*

- *How does communications network resource assignment work in contrast to other real world resource assignment schemes? What are the important considerations while developing and implementing such schemes to satisfy user needs as well as achieve network utilization efficiency ?*

1.3 Methodology

Resource sharing principles apply to all transport protocols that share the network paths. However, in terms of technical implementation and analysis, the scope of this thesis is focused on TCP and its sender side congestion control mechanism. Namely, the comparison is done amongst our own variant of TCP (Flow-Length Dependent Transmission Control Protocol (FLD_TCP)), Cubic TCP and TCP Reno. In terms of the distributive justice aspect, the socio-economic

analysis and discussions presented in this work can be incorporated to other non-TCP modes of network data transport as well. The methodology adopted is to demonstrate how a resource assignment policy can be implemented to realize differential treatment of short and long flows in data transport. This is then used to segue into the other focus of this research which is to analyze various resource assignment schemes.

One aspect of the research is to analyze the dominant data transport protocol, TCP, in an alternate paradigm that differs from the conventional flow-rate fairness dogma. Thus, the methodology adopted for this analysis was to modify the way TCP transports data, conduct measurement experiments and analyze the results. Namely we change the algorithm for sending the data over the network. Then we test the performance of such modified data transport protocol. The first performance analysis was done in a simulation environment, Network Simulator-2 (ns-2) [33]. This was for the sake of simplicity as well as ability to control the operating parameters of the protocol itself. Compared to standard TCP, we observed data transport with almost twice the speed. Then, we implemented a slight variation of the algorithm (described in section 2.3.2) to a Linux kernel and measured the performance with file transfers. We obtained ~30% improvement in the data transfer speed in comparison to standard TCP. Also to evaluate the real world use cases, the implications for modern web traffic was measured by setting up the algorithm in proxy servers and analyzing performance during the web page loads of hundreds of real world websites. In this scenario, the measured benefits, while still significant, were not to the same levels seen in the more controlled previous evaluation experiments due to the composite nature of modern day web page loads. These incremental development and evaluation (starting from simulation experiments towards actual Linux implementation) allowed for utilizing the learning of each step's evaluation on to the development of the next step. The approach is still limited by the emulated test environment (number and type of networks, devices, websites used in the evaluation). But as a demonstrative example, it serves the purpose of building the argument case with regards to the other question focused by this thesis, the resource assignment issues and challenges in the modern data transport.

The other aspect of the research was to analyze the justifications over which the current communication networks resource sharing mechanisms are built. We explored how different notions of distributive justice and fairness have originated and prevailed in the society at large. We developed a framework of how real world fairness notions translate to the TCP friendliness notions in network data transport and identified the caveats of such notions.

1.4 Contributions of the thesis

This thesis summarizes the work done in analyzing the principles of resource sharing mechanisms in network data transport. The thesis contributions consist

of identifying the multiple end goals of efficient data transport. The research outputs include an algorithm named FLD_TCP to provide for ways to achieving those goals. The new algorithm, described in Publication I, enabled aggressive sending rate as long as the flow length was short and be laid back in case of longer flows.

The thesis provides comprehensive evaluations of the algorithm in simulation environment as well as real world settings. A Linux patch was developed creating sender side modifications to the TCP stack's congestion control mechanism. The algorithm's benefits, as presented in Publication II and Publication III, are quantified in terms of file transfers as well as web page loads over various network conditions.

The thesis also provides a holistic and pragmatic view to the issues of distributive justice in network resources assignment. The philosophical background for the origins of justice and fairness policies are identified. Then the resource assignment mechanisms in practice in many economic activities are contrasted to that of the communication networks. The thesis also demonstrates how cascaded policies can have an unanticipated effect in the aggregated resource assignment and that the policies are not commutative. The thesis prescribes rough guidelines to be considered while making judgments about resource assignment policies and their implementation. This work is presented in Publication IV.

1.5 Structure of the thesis

In this chapter, we described the motivations behind the research topic of the thesis. We begin in chapter 2 with a background on TCP and its congestion control mechanisms. We also describe the technical details of an alternate data transport protocol, FLD_TCP, with regards to network sharing and congestion.

In Chapter 3, we provide the performance evaluation results of the algorithm. Evaluations are in both simulation environment and real world scenarios.

In Chapter 4, we discuss the holistic view of real world resource assignments policies and how they translate to communication networks. We analyze the fairness notions and provide some caveats when used in the context of communication networks.

In Chapter 5, potential improvement areas for FLD_TCP are discussed. A discussion addressing modern day communication networks' stakeholders and its drivers for evolution is presented which reinforces the rationale of the research work.

2. TCP Congestion Control and FLD_TCP

In this chapter, we look into the functionalities of TCP and congestion control. Various developments have resulted into many different variants of TCP. We will go through the prominent ones among them. We also describe few changes to some of the TCP sender side algorithms that resulted into our variant of TCP, Flow-Length Dependent TCP (FLD_TCP).

2.1 TCP

TCP, which specializes on providing the *connection-oriented* mode of stream data connectivity, is the most common protocol at the transport layer. To start a data transfer, TCP connection conducts a handshake mechanism that requires 3 trips of message transfer between the client and the server. A SYN data packet is sent to the server which responds with a SYN-ACK data packet. Upon receiving the SYN-ACK packet, the client sends an ACK packet. At this point both the client and the server have received an acknowledgment of the connection after which data transfer can begin.

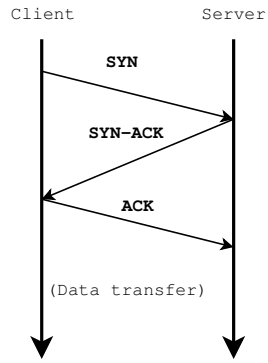


Figure 2.1. TCP 3-way handshake.

During its lifetime, a TCP session provides a multitude of services for the data transfer such as:

- It provides a reliable and an ordered delivery of a stream of bytes between applications: TCP ensures the reliable delivery of data by the use of sequence numbers in packets and acknowledgments. The receiver keeps track of the sequence numbers of incoming packets and maintains a buffer for ordered delivery to the application. Received data packets are acknowledged in a cumulative manner which allows retransmission of the unacknowledged packets from the sender. Hence, the sequence number and the acknowledgment help to counter disorders, fragmentation and packet losses.

- It is optimized for an accurate delivery rather than a timely delivery. Therefore, TCP sometimes induces relatively long delays (in the order of seconds, proportional to overall propagation delay) while waiting for out-of-order messages or retransmissions of lost messages. The 16 bit checksum in the TCP header is used to verify the integrity of received data. Also, there are provisions to use additional checksum using the options field. The checksum ensure the header as well as the payload's integrity.

- It provides flow control, the rate at which data is exchanged to prevent overwhelming the receiver. If the sender is too fast, a slow receiver's buffer will overflow. A sliding window flow control mechanism is used by TCP. The receiver advertises its current window size in the ACKs according to which the sender limits its sending rate. If the advertised window size is 0, then the sender initiates a persist timer and stops sending until the timer expires. Sometimes the window sizes might grow smaller and smaller such that the header becomes larger than the data. This "silly window syndrome" is avoided using Nagle's algorithm to the small packet problem [34] where the sender keeps accumulating small payloads until it is a full packet's worth.

- It controls the network traffic congestion: The networks and paths in the Internet have finite capacity. If a senders transmissions exceed that, the network becomes congested and packets will be lost. So, TCP has mechanisms to infer the network conditions and control its sending rate to avoid overwhelming the network. These congestion control mechanisms also allow for multiple senders to share the network bandwidth among each other.

TCP also has other mechanisms to improve the efficiency of data transmission between the end points. A few prominent ones are introduced below:

Maximum Segment Size (MSS): The Internet paths also have an upper limit for the size of IP datagrams/packets that the physical medium can safely transmit. TCP combats this by the use of MSS that is negotiated during the connection establishment phase. MSS is the maximum size of data TCP can encapsulate in a single segment without considering the TCP and IP headers.

Selective Acknowledgment (SACK): TCP's cumulative ACK isn't efficient in many situations, especially when the packets are lost. Eg. If a packet of a

cumulative ACK space is lost, then the sender has to retransmit all the packets of the current window as all the ACKs from the receiver refers to the packet's sequence number before that of any lost packet. To avoid this, Request For Comments (RFC) 2018 [35] proposed a way to selectively acknowledge the received packets to minimize retransmissions. SACK is used via the options field of the TCP header. SACKs specify the range of packets that was correctly received thus the sender can only resend the lost packets.

Delayed Acknowledgment (DelAck): To better use the links, an optional mechanism of delayed ACK can be used by the receiver so as to avoid sending ACK without the payload/with small payload. The receiver can wait a while as if expecting some more payload from its upper layers to be sent back along with the ACK response.

Window Scaling: Because of the 16 bit window size field in the TCP header, the window size is limited to $65,535(2^{16}-1)$ bytes. However, on links with high bandwidth delay products, this slows down the transmission. In such cases, the window scaling option can be negotiated [36] during the initial handshake to allow the window to increase up to a gigabyte.

Timestamps: TCP timestamps [36] are used to compute the Round-Trip Time (RTT) of transmission paths. The difference in the 4 byte time-stamp value of the sent packet and its echo reply acknowledgment is used to streamline the protocol's behavior in many situations and also in developing newer TCP congestion control algorithms. In situations where the TCP sequence number wraps around, timestamps are used to maintain the ordered delivery of packets.

Out-of-Band Data: When the URG flag is set, the TCP receiver interrupts the stream buffer to send the current packet towards the application. This functionality is benefited by some interactive applications such as terminal emulation e.g. telnet. After processing the urgent data, the TCP resumes back to serve the stream buffer.

Forcing Data: To avoid waiting until the buffer has data greater than the MSS length, as required by certain applications, TCP's PSH flag can force the data delivery even if the send buffer is not filled beyond the MSS.

After completing the data transfer, the connection is terminated by the exchange of FIN and ACK packets from both parties in a maximum of a 4 way handshake fashion. Based on this framework, tremendous volume of data traffic is transported between numerous endpoints on today's Internet at any given time. And the network is shared by many TCP flows concurrently.

2.2 Congestion Control

Network congestion occurs while sending data between endpoints at a rate which is higher than the available network bandwidth. As the network device queues are of limited size, this leads to packet loss and inefficiency in TCP. TCP avoids overwhelming the network by its congestion control mechanism. Congestion

control implies controlling the rate at which a host sends data packets into the network by inferring the network's condition through direct/indirect congestion signals.

2.2.1 Congestion Signals

As data traverse through the network to their destinations, they pass through network nodes such as routers and switches where they are buffered in a queue until the node is free to process the packet and forward it. If the buffers are full, the packets have to be discarded and packet loss occurs. There are many mechanisms to discard the packets at the router buffers depending on the queuing policy implemented. A naive queuing mechanism is a Drop tail/First-In, First-Out (FIFO) queue where all incoming packets, without differentiation, get queued until the buffer is full. Thereafter, incoming packets are discarded. There are Active Queue Management (AQM) mechanisms [37] that act on shortening the queue length, based on some probabilistic rules. Random Early Detection (RED) mechanism, for example, penalizes the flows which have more data in the buffer; it isn't biased towards bursty traffic as much as Drop tail.

Buffers are meant to absorb short-term arrival fluctuations and avoid packet losses. When the links are heavily used, the buffers remain persistently full. If there are too many packets in the queue, processing causes additional delay in the packet's traversal through the network. But in modern networks, as memory became cheaper, the sizes of buffers are getting larger which is causing increased delay overall. This has lead to routers implementing newer AQM methods such as CoDel [38]. Regardless, increasing end-to-end delay can be inferred as signal of network congestion by the end host.

Network nodes can also provide congestion markings on packets, conveying congestion signal to the end hosts. Explicit Congestion Notification (ECN) [39] is one such mechanism. When a router perceives a congestion or a near congestion state, it can mark the ECN bit in the IP header [52] of a packet so that the end host can infer congestion and act on it. ECN can reduce packet drops as the sender can react before the congestion occurs. This is possible if both hosts are ECN capable and the protocol is negotiated during the connection establishment. The network also needs to support it.

The network can also alter its behavior (depending on the algorithm used) towards packet drops, congestion markings and delays to regulate fairness to the flows. The onus then lies on the end hosts to act on those signals to alleviate congestion and to derive the most utility out of the network. TCP congestion control algorithms enable such functionalities.

2.2.2 End-Host Congestion Control

The end host must limit the sending rate of packets into the network to avoid overwhelming network resources which will lead to packet losses and eventual congestion collapse as noted by Van Jacobson [40]. A packet conservation principle was devised to avoid such congestion collapse which stated that in a network operating in a equilibrium state, a new packet is to be inserted into the network only after another packet comes out of the network first.

Most end hosts implement a congestion window to limit their sending rate of packets into the network. It sets the upper limit of data that can be sent for transmission, regardless of the rate at which the application makes the transmit data available. This results into an effective data rate of:

$$Source\,Rate\,(\text{bps}) = W \times MS/RTT \tag{2.1}$$

where,

W = Send Window,
MS = Message Size,
RTT = Round Trip Time

The receiver's advertised window allows TCP to implement flow control and not overwhelm the receiver. The actual window for data transmission (send window) is the minimum of the congestion window and the receiver's advertised window. Thus, in equation (2.1), $W = min(AWND, CWND)$, where $AWND$ = Advertised Window and $CWND$ = Congestion Window.

There are various algorithms in use at the end host for the sake of congestion control. They optimize for data transport efficiency around the Congestion Window (CWND) parameter. The congestion window varies during the data transmission depending on various factors. At the beginning of a connection, the window is set to some default value. Earlier, this Initial Window (IW) size used to be at most 4K bytes [41]. But with modern hosts these days, the IW size is around 10 segments (~15KB) [42].

Slow-Start. Once the connection is established, the sender starts sending out the packets allowed by the CWND and upon receiving each acknowledgment it increases the CWND by one. This essentially doubles the CWND at each round trip time. This is called the TCP *slow-start* phase of the connection, which lasts until the CWND crosses a threshold value (*ssthreshold*) or a loss occurs.

Packet losses during a TCP connection is inferred from the expiration of a timer. If there is no ACK for a sent packet within a timeout, known as Retransmission Timeout (RTO), the sender assumes that the packet is lost and resends the corresponding packet. The connection can again enter this slow-start phase if the retransmission timer expires at any time during the connection. The objective is to reach a higher window as soon as possible to utilize the available bandwidth.

Congestion Avoidance. After the CWND crosses the slow-start threshold size, the data rate is inferred to be closer to the available bandwidth and the subsequent increase in congestion window is more conservative. Additive Increase Multiplicative Decrease (AIMD) algorithm is adopted in most cases where:

if $No\ Congestion$ **then**
  $CWND(i) = CWND(i-1) + u$
else if $Congestion$ **then**
  $CWND(i) = d \times CWND(i-1)$; where, $d < 1$
end if

The values of u and d are varied and adapted to different scenario. Their optimization has led to the development of various TCP variants described in Section 2.4.

Fast Retransmit and Recovery. To make the connection recover faster from a packet loss, corresponding packet is resent not only upon expiration of RTO, but also upon the receipt of three duplicate ACK packets. This is called the Fast-Retransmit algorithm.

While RTO expiration causes TCP to go into slow-start phase and start scaling again from the Initial CWND size, Fast-Recovery algorithm allows for maintaining a higher CWND size (thus high link utilization). It does so by only dropping the CWND to the *ssthreshold* (half of the CWND) upon receipt of three duplicate ACKs. After retransmitting the missing segment, it increases the CWND by additional three segments to accommodate three segments leaving the network into the receiver's buffer as inferred from the three duplicate ACKs. If more duplicate ACKs arrive, the CWND is increased by one as well as a new packet is sent. When an ACK arrives that acknowledges the lost segment in question, the CWND is deflated back to *ssthreshold* and the congestion avoidance resumes.

RTT Variance Estimation. The Round Trip Time between the end hosts is used by TCP for various purposes, RTO estimation being one of them. RTO packet headers include a timestamp option which is used to calculate the per packet RTT. Over multiple packets, network noise induces variations in the RTT values that need to be filtered out using a low pass filter. The Smoothed Round-Trip Time (SRTT) for use is computed as follows:

$$SRTT = \alpha \times SRTT + (1-\alpha) \times M$$

where, M = most recent RTT measurement, and α = 0.8~0.9. General TCP implementations calculate RTO as: $RTO = SRTT + 4 \times var(RTT)$. Many other algorithms including Kalman filtering [43] have been developed for better estimating the RTT.

With these congestion control mechanisms from both network as well as end hosts working in conjunction, flow rates of TCP flows are controlled. Multiple flows sharing the network compete for the access of the same available bandwidth. Each flow tries to get equal share of network bandwidth at any given moment, irrespective of history, future, or the cost created to other flows.

In the next section, we describe the design of a new congestion control algorithm whose behavior varies according to the number of packets sent in a TCP connection flow. We call this the Flow-length Dependent Congestion Control and the details are published in Publication I.

2.3 FLD_TCP

We have discussed that most of the traffic flows in the Internet are short and only a small proportion of the connections are long. We also established how flows infer network conditions and incrementally attempt to gain a larger share of the network bandwidth. In this resource competition scenario, the short flows are disadvantaged as long flows have more time to infer the network conditions and react accordingly. Short flows also cannot possibly utilize large available bandwidth due to congestion window scaling from a smaller initial value. We also argued that prioritizing the completion time of short flows could lead to various benefits to user experience as well as energy savings.

With the objective of accelerating the transport of short flows (flows with less bytes/packets to transport), we propose some changes into the TCP sender side algorithm. After a TCP connection is setup, with each RTT the connection optimizes to get larger share of bandwidth from the network. Since short flows have less bytes to transport, they may finish all their data transmission before TCP reaches to an optimal state due to slow-start. To level the playing field, one can boost the sending rate from the very beginning, i.e., have a larger Initial Window. This has been incorporated in modern TCP versions [42].

We alter the congestion control algorithm in such a way that it is very aggressive in the initial round trips of a TCP flow, and becomes less aggressive as the flow gets longer. This causes the short flows to achieve more of the network bandwidth, allowing shorter flows to push more data into the network in the same time. We use a threshold flow-length, after which we step down the aggressiveness. To make TCP perform a go-fast-finish-early behavior, we propose a new TCP, Flow-Length Dependent TCP (FLD_TCP) by altering two mechanisms of the traditional TCP.

The objective of building this TCP variant is to demonstrate how a resource assignment policy can be implemented to realize flow-length varying aggressiveness for data transport which allows us to segue towards analysis of various resource assignment schemes. Thus, due to the direction, scope and time requirements of the research work, optimization of FLD_TCP itself has had limited treatment in this research. For example, FLD_TCP developed in this work does not implement safeguards against the overshoot of sending rate during slow-start. However, such algorithms can be added to refine FLD_TCP without affecting the intended primary goal of the protocol which is to favor short flows. We discuss some of these features, which could constitute future work in this area, in the concluding chapter.

2.3.1 Slow Start Increment Increase

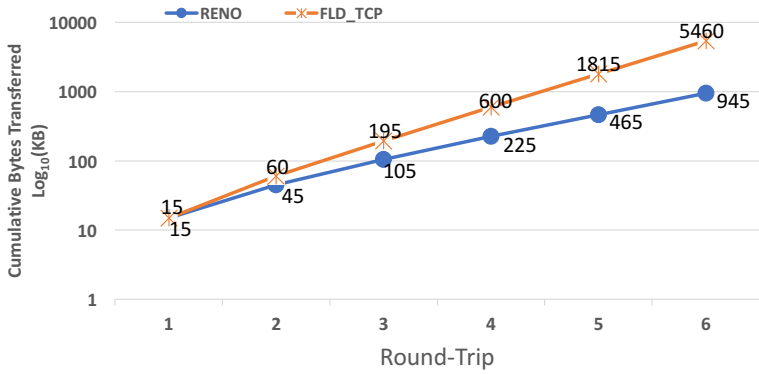


Figure 2.2. Two different Slow-Start Increments

In traditional TCP Slow-Start, the congestion window is increased by 1 segment for every ACK received which effectively doubles the CWND per RTT. In FLD_TCP, the slow start increment factor is changed from 2x to 3x as shown in Figure 2.2. Thus, the CWND proceeds towards saturation much quicker and aid the data transfer acceleration.

2.3.2 Congestion Avoidance

The congestion avoidance phase of FLD_TCP uses two different algorithms depending on the amount of data that has been transmitted. As long as the flow-length (in bytes) is smaller than a threshold, an aggressive algorithm known as Relentless TCP is used. Relentless TCP [44] does not follow AIMD principle but rather the window decrease is the amount of packets lost only. This makes the relentless TCP more aggressive than the standard TCP.

We want to boost up the transfer of flows that are smaller than the threshold flow length but set back the aggressiveness if the transfer size is larger. So post-threshold, we make the algorithm laid-back. To compensate for the aggressiveness at beginning of the connection, we tested by changing the multiplicative decrease part such that the congestion window reduces to one third rather than halve it. We found no significant benefit towards our objective from it. Thus in our Linux implementation, we implemented the post threshold behavior to be TCP Reno-SACK with the standard rate-halving. Accelerating transport for short flows implies added utility to the web-surfers and the transaction oriented web sessions by shifting the load to the longer flows of background transport and the likes.

2.3.3 Threshold Selection

The threshold value is set to accommodate what constitutes a short flow in the current web traffic. We identified in the previous chapter that a median size of a web page on desktop is around 1.8 MB, while on mobile it is around 1.6 MB. Median size of images is almost 1 MB and javaScripts have median size of ~400 KB [21]. We have also seen in Publication III, p. 4 that almost 99% of web connections transport less than 500kB of data. A threshold of 2MB was selected for the purpose of this research. Thus, FLD_TCP is aggressive when the flow length is less than 2MB, and TCP friendly if the flow length is greater than that.

2.4 Related Works

A plethora of TCP variants have been developed over the years to cater for different networks and service profiles. *TCP Tahoe* [45], one of the preliminary implementations of TCP, incorporated the slow-start, fast retransmit and congestion avoidance algorithm for congestion control while *TCP Reno* added the fast recovery part and SACK was added in a later TCP version. *TCP Vegas* was congestion control mechanism whose congestion engine was triggered more by RTT variations than by packet drops. It was less aggressive to Reno and also network rerouting caused problems to its RTT based operations.

To be TCP-friendly as well as scalable in high-bandwidth long distance networks with long RTT values, *TCP BIC* [46] incorporated a binary search increase algorithm into its congestion control. When combined with the additive increase algorithm, it allowed for the initial increase of congestion window to be linear and the increase near link saturation to be logarithmic. It also allowed for aggressive bandwidth probing upon no loss conditions thus improving link utilization. *CUBIC TCP* [47] is a more TCP-friendly version of BIC where a cubic function controls the growth rate of the congestion window depending on the time elapsed since the last loss.

Similarly, *TCP Compound* maintains loss triggered conventional congestion window (CWND), and a delay triggered window (DWND) to correspondingly achieve TCP scalability and RTT fairness in the high-bandwidth long distance networks. Also, *Bottleneck Bandwidth and RTT (BBR) TCP* [48] has been recently developed that uses recent measurements of the network's delivery rate and round-trip time and builds a model to be used for congestion control. It attempts to operate at the point of onset of queuing at the path bottleneck.

In contrast, our proposed TCP variant, FLD_TCP is a set of modifications to the traditional sender side algorithms, with the flow-length based differing algorithm behavior as the distinguishing feature. The goal of the algorithm is to favor short flows, for which we introduced more aggressive behaviors during the initial period of the TCP connection. Bigger initial window size as in newer TCP variants [42] and an increased slow-start increment factor combined with

relentless congestion avoidance behavior enabled us to create a TCP variant exhibiting such behavior.

After a TCP connection is established, it incrementally attempts to gain a larger share of the network bandwidth. During this probing phase the link utilization is not efficient and also is a cause of unfairness between short and long flows. Some approaches attempt on solving the issue by starting the data flow with a data rate that is as close to the available bandwidth as possible. Quick-Start [49] proposes a mechanism where during handshake the receiver can relay a desired sending rate back to the sender. But this method has dependency on the routers along the path and they should either agree or modify the sending rate value as it gets to the sender. The sender then can start data transmission on the agreed sending rate from the very beginning. Jumpstart [50] removes the dependency of Quick-Start on the IP layer by proposing that the sender start sending at an arbitrary rate but pace out the packets allowed by flow control over the RTT inferred from the three-way handshake. Not only does Jumpstart incur higher packet drop rates than slow-start but it also prolongs flow completion times as compared to slow-start when multiple flows are sharing the network.

Halfback [51] is another more recent approach which borrows the pacing idea of Jumpstart but also adds upper bound to the data transmitted as minimum of the flow control window size, flow size, and a Pacing Threshold. After the pacing phase, it proactively retransmits the flow's packets in reverse order at the same rate that it receives ACKs from the previous phase, in addition to the normal packet retransmission. This increases the network utilization by preemptive retransmissions in Reverse-Ordered Proactive Retransmission phase of ~50% of the packets in pacing phase. Performance wise, Halfback appears to improve flow completion times as compared to Jumpstart.

Overshooting of the sending rate during slow-start causes wasteful packet losses in TCP and being more aggressive in the slow-start phase further exacerbates it (as shown in our evaluation chapter). Network buffers along the TCP path and their corresponding queuing mechanisms also play a significant role in this [1]. Regardless, various TCP methods allow for switching from slow-start to congestion avoidance mode safely without causing heavy packet losses. Limited slow-start [52] mitigates this by capping the exponential growth up to a *max_ssthresh* parameter value. Beyond that size of the congestion window, the exponential growth is replaced with a less than linear window increase. Cubic Hystart [53] utilizes ACK trains and RTT delay measurements to infer whether the current congestion window size has reached the available capacity of the forward path. This allows the connection to exit slow start earlier, before losses start to occur. Hystart++ [54] utilizes a combination of both Hystart and Limited slow-start to increase congestion window further until the first

[1] Mechanics of bufferbloat in network nodes have been an actively researched issue. This research attempts to limit itself on the sender side mechanisms for technical implementation and evaluation

packet loss occurs. TCP Pacing [55, 56] allows for shaping the TCP flow and avoid bursts thus mitigating the overshooting issue. An approach of probing for multiple feasible flow rates is used in TCP RAPID [57] to adapt to available bandwidth faster and avoid packet losses. Paced chirping [58] builds on that method of inferring better estimate of the available bandwidth by using trains of packets called 'chirps', thereby allowing connections to avoid overshoot.

Various agencies, networks and service providers are also professing/deploying their own flavors of TCP protocols. Some of those are more aggressive than traditional TCP such as Relentless TCP [44] and FastTCP® [59]. While others are less aggressive than traditional TCP such as TCP-Nice [60], TCP-LP [61] and Low Extra Delay Background Transport (LEDBAT) [62].

2.5 Summary

In this chapter, we provided background on the various elements of TCP, the dominant transport protocol in the Internet. We also described various network congestion signals and how end host congestion control operate over them. There exist many flavors of TCP tailored for optimization of various objectives. Also, the TCP-friendliness notion has not been universally implemented in practice.

We also described FLD_TCP, which includes changes in the slow-start and congestion avoidance algorithm of TCP. This TCP flavor devised for our research purpose has the aggressiveness as a function of the flow length. In the next chapter, we describe the evaluation of FLD_TCP in different scenarios.

3. Evaluation

In this chapter, we go through the prominent results obtained during the comprehensive evaluation of FLD_TCP. The evaluation of FLD_TCP is done in three different scenarios. First we implemented the algorithm in the NS-2 network simulation environment as a proof of concept, the results of which are published in Publication I. We then implemented the algorithm in Linux machines and conducted a controlled performance test against Reno and Cubic flows, in lab conditions. In Publication II, we reported on various metrics (goodput, flow completion time, retransmissions) observed of such flows while they transmit data over Third Generation (3G), WiFi and Local Area Network (LAN) environments. In Publication III, we evaluated the effects of FLD_TCP on real web browsing experience. We tested the performance of the protocol while transporting web pages of the top 100 sites from Alexa top sites list [63]. Below we provide a summary of results.

3.1 Simulation

With FLD_TCP implementation in NS-2 [33], we created a simple dumbbell topology (with four hosts labeled A-D and two routers labeled A and B) as shown in Figure 3.1 and set up competing flows across the hosts.

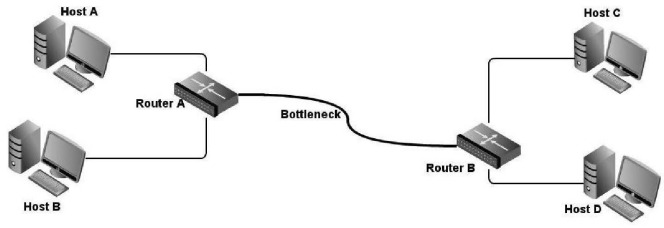


Figure 3.1. NS-2 testbed.

Figure 3.2 shows the results when we have two TCP Reno flows competing versus when we have a FLD_TCP flow competing against TCP Reno flow. We see that the two competing Reno flows achieve comparable bandwidths. FLD_TCP

on the other hand, obtains a high bandwidth in the beginning in line with our implementation of higher initial window, higher slow-start increment factor and the one-third multiplicative decrease algorithm. We also see the congestion window of FLD_TCP quickly surpass that of Reno and then ramp down later. In the same figure, at around 18 seconds we also witness the RENO flow suffering from a retransmission timeout causing a period of convergence between the two flows' congestion windows.

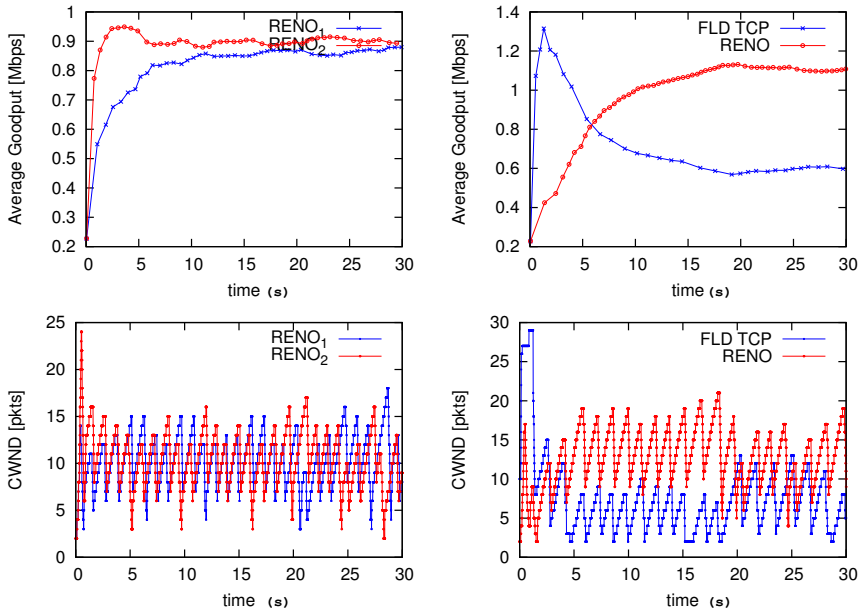


Figure 3.2. Simulation results: Link capacity: 2 Mbps, Link Delay: 50 ms

In another test, we set 3 background flows each of 15 MB of file transfers starting at 0, 2 and 4 s and introduced 4 bursts of 512 KB, 1 MB, 1.5 MB and 2 MB traffic at 10, 15, 20 and 25 s respectively. Figure 3.3 shows the observations for Reno and 3.4 for FLD_TCP. The transfer times of each of the flows are also seen in Table 3.1 and 3.2. The short flows of FLD_TCP have roughly half the transfer time than that of the corresponding Reno bursts. Also the effect on the completion time of background flows are minimal.

Also from the chart of the congestion window and losses, we see that loss rate is more for FLD_TCP leading to more instantaneous delay as well. Controlling the overshooting of the congestion window should be be able to minimize the loss rate.

Thus, the simulation experiment showed promising results for FLD_TCP. Publication I contains more details on the evaluation.

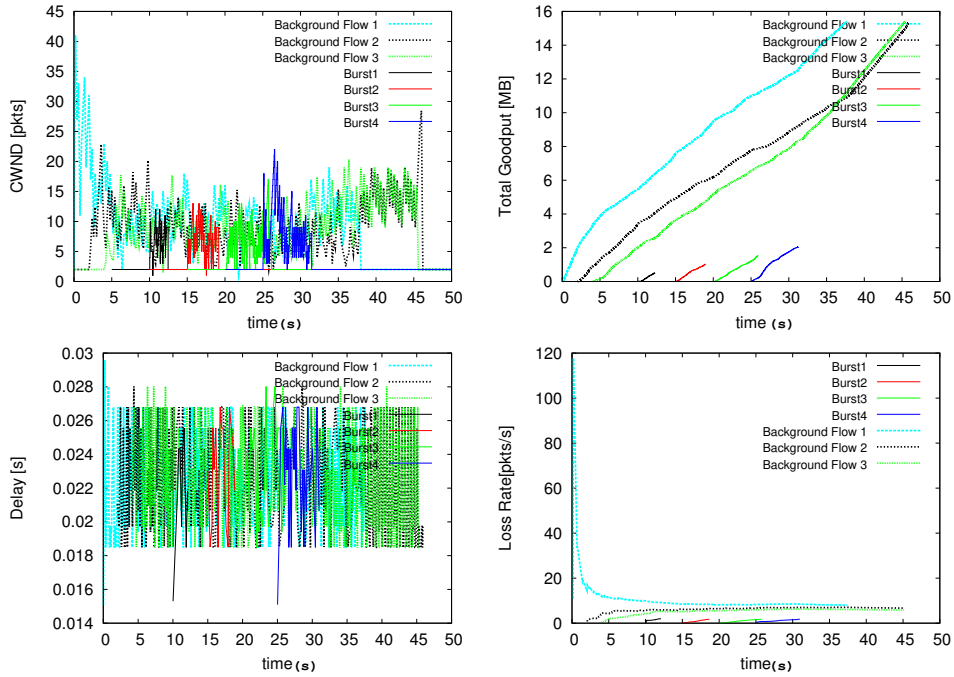


Figure 3.3. 4 Reno Short Bursts in 3 background flows

Table 3.1. Bursts Transfer Times

	FLD_TCP Transfer Time	Reno Transfer Time	Ratio (Reno/FLD_TCP)
Burst (512KB)	0.6 s	1.7 s	2.833
Burst (1MB)	1.5 s	2.4 s	1.6
Burst (1.5MB)	2 s	4.6 s	2.3
Burst (2MB)	2.8 s	6.2 s	2.2

3.2 Controlled File Transfers

After the simulation evaluation of the concept, the protocol was implemented in Linux kernel. Its evaluation was then done with controlled file transfers on real network settings.

Linux Implementation

The FLD_TCP implementation was done in Linux-2.0.78 kernel. The patch and fld_tcp.c module is available at [64]. In this Linux version the Initial window is 2 but FLD_TCP overrides it to 10 when initialized. The *turbo* and *agr* flags indicate if the flow length is less than 2MB.

```
static void tcp_fld_init (...)
{
```

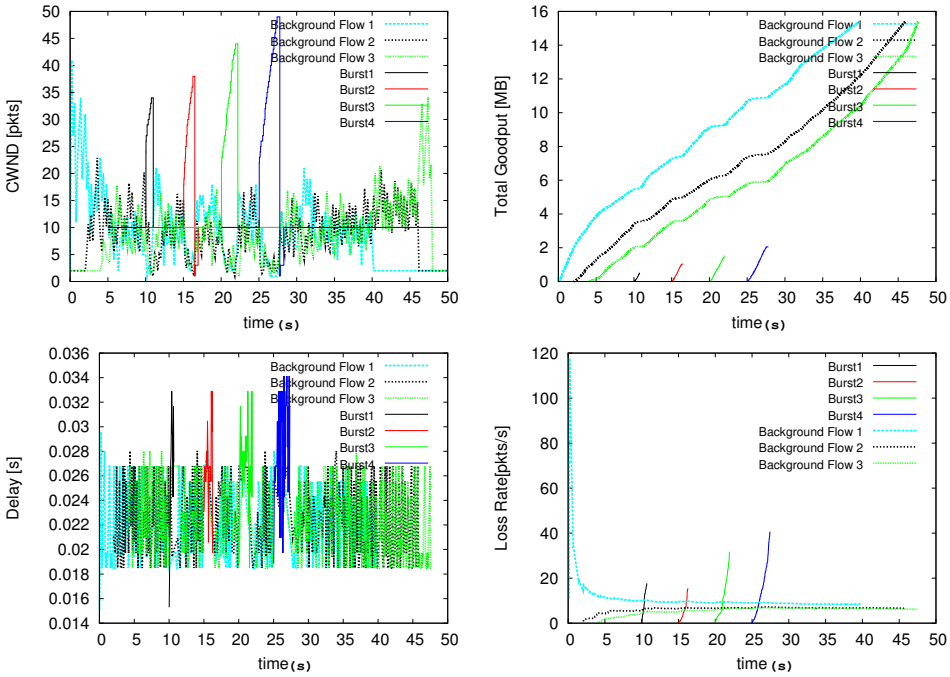


Figure 3.4. 4 FLD_TCP Short Bursts in 3 background flows

Table 3.2. Background Flows Transfer Times

	With FLD_TCP bursts	With Reno bursts	Ratio (FLD_TCP/Reno)
Background Flow 1	39.3 s	39.3 s	1.0155
Background Flow 2	43.9 s	41.6 s	1.0555
Background Flow 3	43.7 s	42 s	1.0405

```
...
tp->turbo=1;
tp->snd_cwnd = 10 ;
fld->agr = 1 ;
...
}
```

When the flow length of the connection crosses 2MB, the algorithm is same as traditional TCP. There are few differences when the flow length is within 2MB. During slow start (i.e. CWND is less than the slow start threshold), if the flow length is less than 2MB, then the CWND is increased by 2 segments for each ACK received. On the other hand, if it is above 2MB then it is increased by 1 segment per ACK.

```
static void tcp_fld_cong_avoid(...)
{
...
```

```
if (tp->snd_cwnd <= tp->snd_ssthresh
   && fld->agr
   && tp->snd_ssthresh != 0){
        tp->snd_cwnd += 2;
} else if (tp->snd_cwnd <= tp->snd_ssthresh){
        tp->snd_cwnd += 1;
} else
...
}
```

Similarly, during the fast recovery phases, if the total bytes sent is less than 2MB, the CWND is set to be minimum of the CWND and packets in flight plus one instead of halving the rate. This is still in accordance with the packet conservation principle. Within the 2MB flow length, the slow-start threshold is set as CWND minus lost packets unlike CWND/2 after crossing 2MB.

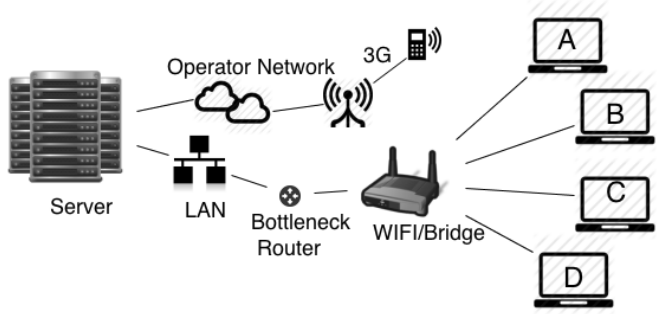


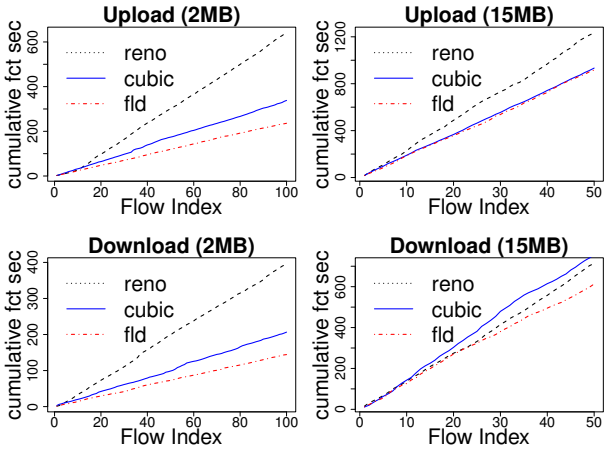
Figure 3.5. File transfer test setup.

Testbed and Results

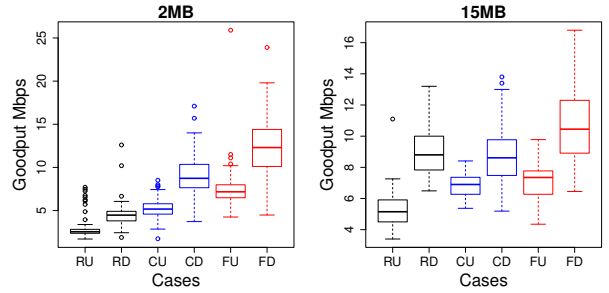
With these modifications, the new TCP flavor is evaluated with a setup shown in Figure 3.5. The modified kernel was compiled in the host machines including a Nokia N900 mobile phone for 3G test. In 3G, the network itself has natural characteristics of background traffic, delay and bandwidth bottlenecks. For WiFi and LAN tests, a middle-box was used to introduce those characteristics using *netem* [65] and background traffic was created when needed. Comparative performance tests of FLD_TCP, Reno and Cubic flows were done in 3G, LAN and Wifi environments and various metrics were observed. Comprehensive results are in Publication II. We highlight the Wifi case below:

Figure 3.6 shows the results when 100 short flows (2MB) and 50 long flows (15MB) are uploaded and downloaded sequentially between server and clients. Ten continuous Reno SACK flows share the same bottleneck (100 Mbps) creating background traffic and end-to-end delay of 50 ms was created using netem in the bottleneck router.

The cumulative flow completion times (fct) shown in Figure 3.6a, demonstrate



(a) Cumulative flow completion times



(b) Connection goodput

Figure 3.6. Wifi Upload and Downloads

the gain obtained from using FLD_TCP. In WiFi upload and download cases, cumulative fct of 100 short flows of 2MB size show that flows on FLD TCP is 63% faster than Reno flows and 30% faster than Cubic short flows. For the 15MB upload, FLD_TCP and Cubic finished flows at similar time while Reno had quite high cumulative f.c.t. Similarly the goodput charts in 3.6b show that FLD_TCP upload (FU) and FLD_TCP download (FD) achieve higher goodput than corresponding Reno (RU, RD) and Cubic (CU, CD) flows, especially in the 2MB case.

In order to see how the bandwidth is shared by different TCP variants, all four hosts make 25 connections each with the server simultaneously. After every run of the flows, the TCP variant is changed for one client-server pair (i.e. at first all clients (A -> D) have Reno enabled. For second run, client A has FLD_TCP and (B -> D) have Reno, and so on).

To enable the same in download case, two servers have two different TCP variants enabled at a time. This allows us to observe the change in average bandwidth share of flows at the bottleneck when 25% flows change their TCP algorithm in every step. The averages are made between flows of the same

client-server pair since they use the same TCP variant at any given step of the experiment. Figure 3.7 shows the bandwidth attained by each variant in each run of the experiment.

For example, in the "Upload 2MB" case, when all the flows are Reno (“RRRR”), average goodput of each flow is around 1 Mbps. When one client has FLD_TCP enabled (“FRRR”), the average goodput of the FLD_TCP flow is quite high in this case. In “FFRR” and “FFFR” cases also, FLD_TCP flows get higher goodput than the Reno flows. Contention of FLD TCP with Cubic flows is similar but the bandwidth share increase is lesser. FLD_TCP also achieves higher goodput for short flows in download case. For long flows, FLD_TCP does not attain more goodput than others as intended by design. However, we also see that in the 2MB cases, Reno and Cubic flows achieve lesser and lesser bandwidths as the proportion of FLD_TCP flows increases. Actually, the achieved goodput in 2MB ’FFFF’ cases appear slightly worse than ’RRRR’ and ’CCCC’ cases. This limitation of FLD_TCP is unlikely to manifest in the real networks where ’FFFF’ situation would be a mix of short and long flows whereby short flow would benefit.

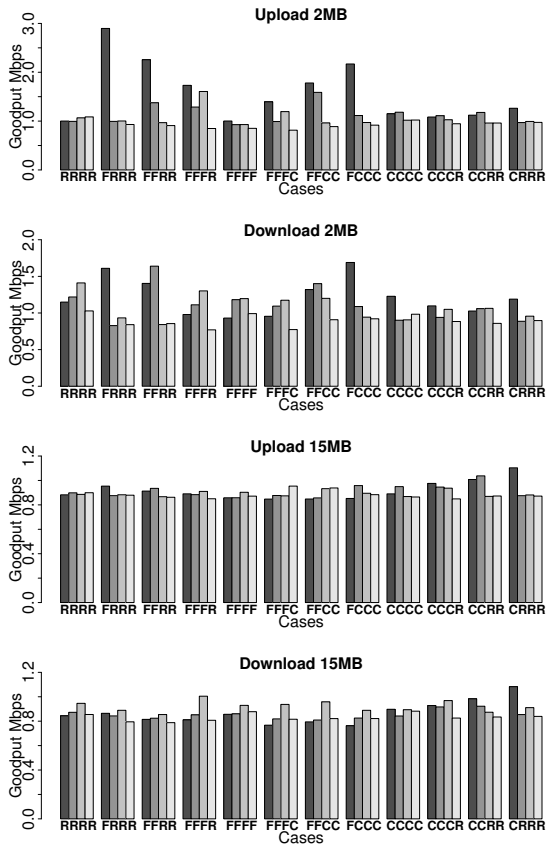


Figure 3.7. Bandwidth sharing in Wifi

After the simulation and static files evaluations, the protocol is next tested with real web pages to see the impact it has on modern web page loading on modern browsers.

3.3 Web Page Loads

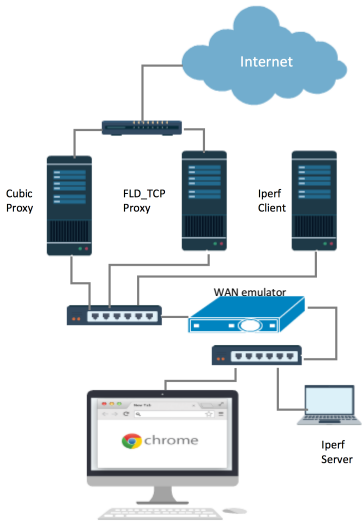


Figure 3.8. Evaluation Setup

In order to measure the benefits of FLD_TCP on the downloads of real web pages into a user's browser, we implemented the transport protocol in a Squid proxy server (to emulate the content servers) and measured the performance while accessing the Alexa top 100 sites via different network scenarios (using a WAN emulator and iPerf [66] background traffic) on a Chrome browser. Page load times were obtained from the Chrome browser's *loadEventFired* event. The comparison is done against Cubic as it is the dominant TCP on most web servers. The evaluation setup is presented in Figure 3.8. We also analyzed the connection-level patterns of modern day websites. There are idle times during the page load, during which there is no ongoing data transfer from the network to the browser. Thus, we calculate the *page net transfer time* as the difference between the page load duration and the idle duration during the page load. The results below show significant improvement of page net transfer times through the use of FLD_TCP in both HTTP/1.x and HTTP/2.0 cases.

We analyzed the connection sizes distribution of the home pages of those Alexa top 100 websites. The distribution was found to be similar to the connection distribution of webpages of more than a million web pages maintained by *httparchive (https://httparchive.org/)* as shown in Figure 3.9. Almost 85% of connections transport less than 50 kB of data and 99% of connections transport less than 500 kB of data.

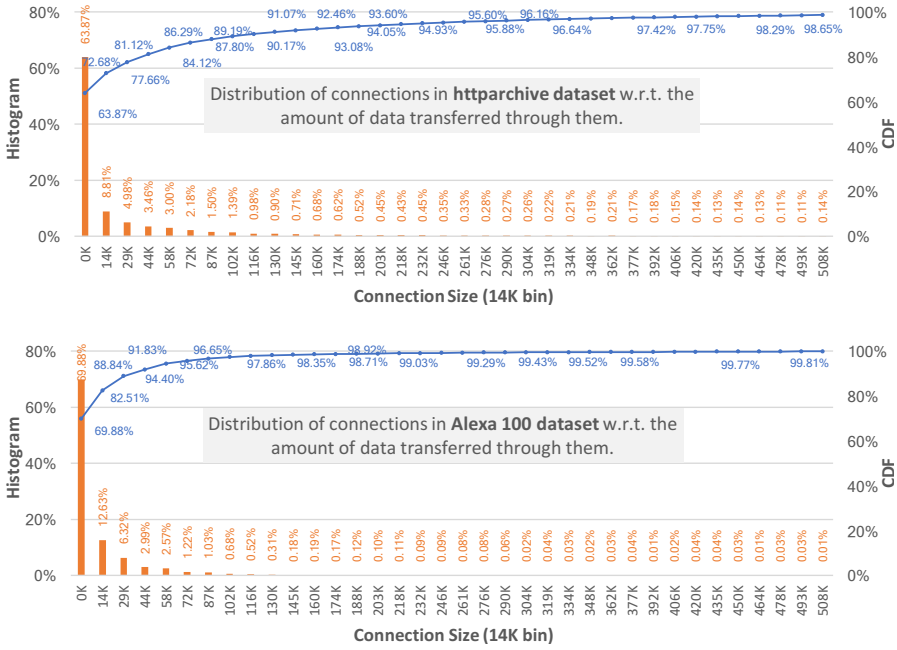
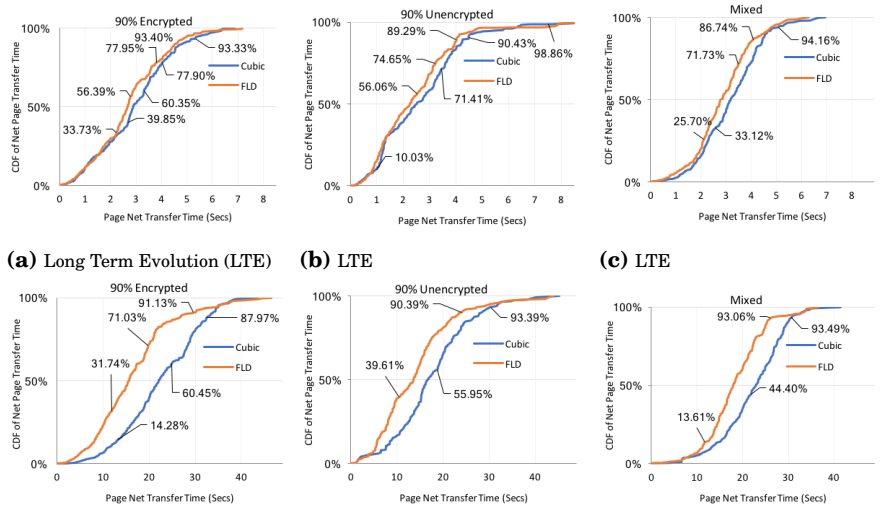


Figure 3.9. Comparison of connection sizes distributions in homepages of Alexa 100 sites with the larger dataset of http://httparchive.org.

Modern day TCP connections are increasingly getting encrypted. It was observed that 37.6% of web pages have more than 90% of the TCP connections encrypted while 33.4% of web pages had more than 90% unencrypted connections



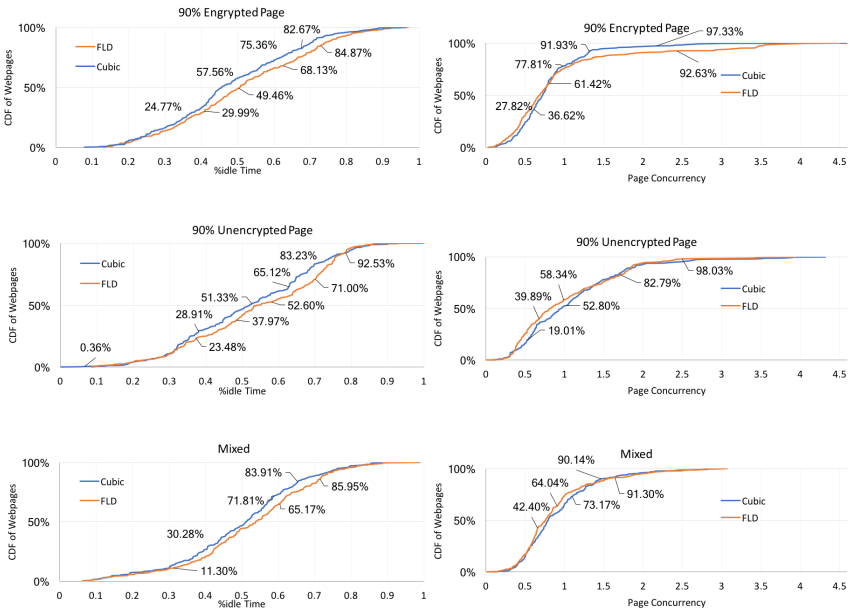
(a) Long Term Evolution (LTE) **(b)** LTE **(c)** LTE

(d) 3G **(e)** 3G **(f)** 3G

Figure 3.10. Page Net Transfer Time obtained for different page types for Cubic and FLD_TCP

serving the pages. However, 29.1% pages were served by both encrypted and unencrypted TCP connections. The analysis is later segmented into these different page types. Also analysis is done for effect on HTTP/2 connections which allows for multiplexing.

Figure 3.10 shows the effect on page net transfer times of the three different page types on different network characteristics upon using the two different transport protocols, Cubic and FLD_TCP. Each point represents an individual run of fetching one of the 100 web pages and each page is fetched 11 times. We see that FLD_TCP reduces the net page transfer time for all page types (encrypted, unencrypted, and mixed, where we define an encrypted page as one where more than 90% of the connections of the page are encrypted and so on). The reduction in page net transfer time is more in 3G network setup than LTE, meaning that FLD_TCP provides more benefit in low bandwidth situations.



(a) Distribution of idle time inside pages for different page types.

(b) Effective connection concurrency across pages for different page types.

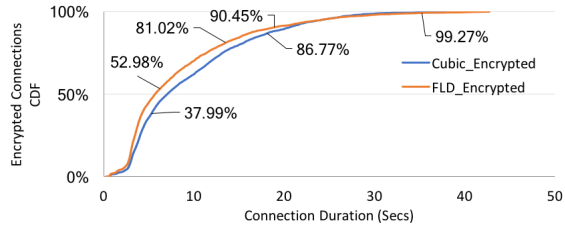
Figure 3.11. Distribution of idle time and connection concurrency

There is a lot of idle period during a page load when there is no active fetching of web objects from the server. The median number of pages have around 50% of page load time as idle from network data reception perspective. Figure 3.11(a) shows the Cumulative Distribution Function (CDF) of such idle time durations during the page load on LTE indexed against the total page load duration. We observed that FLD_TCP increases the idle time proportion in receiving objects for the page as FLD_TCP finishes object transfers faster than Cubic.

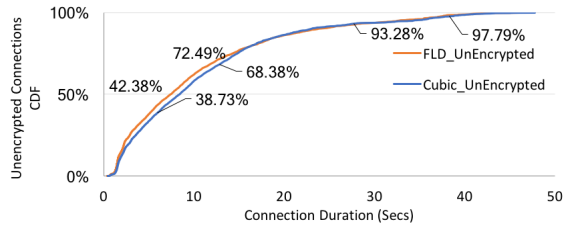
We also analyzed a page level effective concurrency metric for each page load

that denotes the effective number of connections seen throughout the page load duration. This was done by tracking the number of active connections (objects being received through them) and their cumulative duration. A concurrency level of 1.5 would imply that without any idle time, the page could be served the same amount of data through 1.5 connections. Figure 3.11(b) shows that FLD_TCP decreases the effective concurrency of the page compared to Cubic, implying that it would require fewer TCP connections to serve a page, leading to an improved connection efficiency. While this metric serves to amplify the gains of FLD_TCP by removing the idle time from the analysis, it also shows the potential benefits in a situations with multiplexed data transfers.

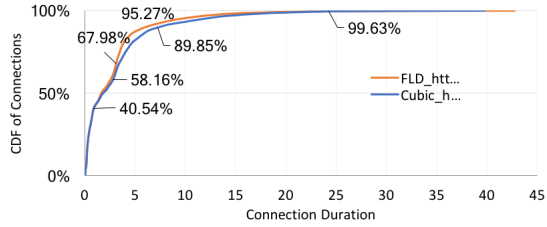
We also analyzed the measurements of three different connection types. Figure 3.12 suggests that FLD_TCP slightly decreases the connection durations of encrypted, unencrypted as well as HTTP/2 connections.



(a) Across encrypted connections.



(b) Across unencrypted connections.



(c) Across HTTP/2 connections.

Figure 3.12. Distribution of connection durations

To see the window increase function of TCP in action, we analyzed the maximum size of web objects transferred in each round trip time of the connection considering only the first object of the connection. The LTE setup used for this purpose had link delay of 13 ms leading to an RTT of at least 26 ms. Figure 3.13 shows the bins and corresponding maximum data received with Cubic and FLD_TCP. We see that in the first RTT of the connection, both FLD_TCP

and Cubic receive data of a maximum 14 KB size in accordance with the 10 segments initial window size. In the second RTT, Cubic's window size doubles while FLD_TCP's window size triples, which allows larger web objects transfer to also finish within the second RTT. Similarly, in each of the succeeding RTTs, FLD_TCP is able to finish the transfer of larger web objects than Cubic.

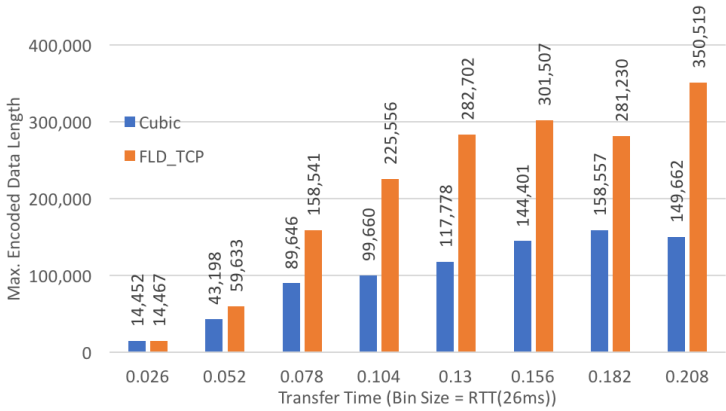


Figure 3.13. Maximum size of web objects transferred in each RTT bin

From this real world evaluation of FLD_TCP, the benefits of using FLD_TCP at the sender side of TCP connections are clear. On the objects level, RTTs are saved, hence web objects are fetched faster. On connection level, the connection durations of the TCP connections become shorter for both encrypted (including HTTP/2) and unencrypted connections. These accumulate at the page level, making the web pages load faster thus improving user experience on the web. Further results of these evaluation experiments can be found in Publication I, Publication II and Publication III.

3.4 Conclusion

We took the idea of changing the TCP sender side behavior to favor short flows and implemented as well as evaluated the new algorithm via simulations, laboratory setups as well as with public websites. We found FLD_TCP to work well and provide intended benefits to short flows in comparison to the traditional Reno TCP as well as the dominant Cubic TCP.

Even a marginal improvement of web latency significantly affects revenue for service providers. This is primarily from user experience improvement perspective that is geared towards user retention to those Internet services. From energy saving perspective as well faster transmissions and completions of web connections make sense since the energy cost per bit for wireless transmission is 1000 times more than the energy cost for single computation of that bit in modern wireless devices [25].

Thus, the idea of using a go-fast-finish-early transport mechanism such as

FLD_TCP certainly has its merits. However, the question of fairness is also brought forward by such idea of prioritizing short flows. The next chapter delves into the application of fairness aspects of network data transport derived from the real world notions.

4. Distributive Justice

In this chapter, we look into the origins and development of fairness notions and how they translate into network operation. We then analyze how differing notions applied in cascade can result into totally different outcomes. This is followed by a discussion into the basic guidelines for implementing distributive justice in network communications.

4.1 Introduction

Fairness of resource assignment is often questioned, analyzed and justified in areas where the distributed resource is of economic value and in limited supply. There is a constant struggle between personal gain and social welfare resulting from different kinds of resource distribution policies. Optimization of these competing aspects and finding a stable equilibrium is a non-trivial problem. In communication systems there are many commodities, such as bandwidth, delay, and other performance measures, which are of economic value that require resource assignment policies. The outcome is an interplay between users, applications, network, network operators and the society through regulators. The questions of net-neutrality, network utility maximization, quality of service, quality of experience are examples of the competing objectives desired by these stakeholders of communication networks.

Considering the hierarchy of networking needs as shown in Figure 4.1, as proposed by Alvaro Retana [67], one can see that the network resources are spread across all of the layers. After providing connectivity with acceptable level of performance and some usable functionality, concerns on the justice in allocation of network resources shall arise in the priority layer which stems from the basic resources distribution. This layer is analogous to the esteem layer of Maslow's hierarchy of needs [68] where one seeks respect and individual recognition. Justice is sought and questions of fairness are put forth at the priority layer. Thus, during the network evolution, such needs may need to be addressed duly. Considerations of such desires should be included in the design process of the resource assignment policies themselves.

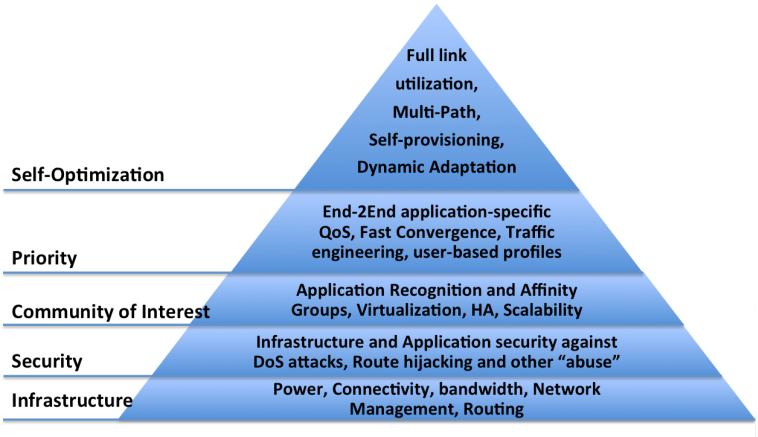


Figure 4.1. Retana's Hierarchy of Networking Needs

Some services value delay more than bandwidth, some value in-sequence delivery, some value accuracy while others value something else. It is a challenge for the network architects, designers and operators to identify and provide network performance to satisfy all those needs. Due to the modular characteristic of the communication network's stack, its different layers have their own design considerations. Also due to differences in the administrative realms of the networks themselves, policies that guide the assignment of network resource usage vary. Thus this diversity of service, stakeholders, administrative realms and constantly changing market forces makes it a complex and yet very crucial to carefully design the principles of assigning resources in network communications. Tussles in cyberspace [69] result from inadequate design and self-interests of the users.

4.2 Origin and Development

It has been demonstrated that even capuchin monkeys reject unequal pay [70]. Also as societies are exposed to more market integrations they are more prone to seek fairness in transactions [71]. Thus, fairness appears to have evolutionary roots and is further engrained in humans by the social construct as the perception of distribution is established.

Along the modern society's formation, philosophers have taken a keen interest in understanding the human need for fairness in the distribution of goods. The principles of distributive justice are normative principles designed to guide the assignment of the benefits and burdens of economic activity. Its scope varies on what is subject to distribution, the basis of the distribution and the nature of the subjects of the distribution [72]. Various distribution policies are regarded as just by different schools of thought. For *strict egalitarianism*, every person should

have the same level of material goods and services no matter what. For the *difference principle* [73], inequalities in distribution are allowed only when they are open to all, under conditions of fair equality of opportunity and distributions should prioritize greatest benefit of the least advantaged members of society. This led to Rawls' MAXI-MIN approach, i.e. maximize the opportunities and minimize the inequalities among all. *Welfare-based principles* prefer maximizing the overall level of wellbeing, which results in policies favoring the majority. *Desert-based principles* focus on the entitlement to resource-share based on individual's toil. And *libertarian* principles do not require any distributive pattern to be termed just. Real world policies are derived from these principles according to the justification of the people and the law of the land.

Different instruments have been developed to implement the fairness notions in the economic markets. *Market price*, *command*, *majority rule*, *contest*, *first-come-first-served*, *lottery*, *personal characteristics* and even *force* [74] have been used to decide the entitlement and priority order for resource distribution. In cases where the individuals have an awry sense of entitlement, the distribution mechanism (which is intended for maximizing the collective welfare) can result into negative situations (Tragedy of the commons) [75].

Optimality of distribution is also used to compare different schemes. For example, while Pareto optimality implies that one participant cannot benefit without causing loss to the other, Kaldor-Hicks allows for optimality in cases where the loser of a contention can be compensated in some manner so that no one is worse off [76].

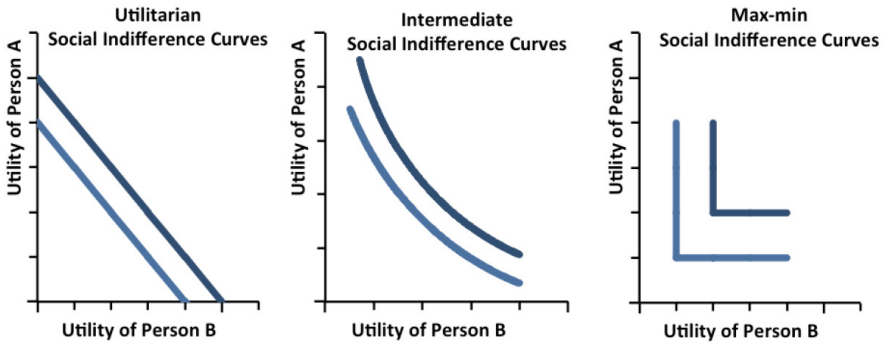


Figure 4.2. Different Social Indifference Curves

Another way to compare the different schemes is through translation of the social welfare functions into social indifference curves as shown in Figure 4.2 where the two curves in each sub figure denote different total capacity. Different schemes accept allocations according to the resulting utilities on the corresponding participants. For the pure utilitarian, the sum of utilities needs to be the same (maximum possible in the system) while for the Maxi-Min approach the utility for one user can only be increased without decreasing the utility for the user, which has less utility. The intermediate indifference curves (in the middle) justify the increase in inequality by making increase in utility for the

one gaining much greater than the utility decrease for the other.

4.3 Network Communications

Communication networks have various resources that are limited and shared among the communicating entities. Thus, as other economic goods, their distribution is guided by various schemes either by design or by providence. Bandwidth, buffer space, processor cycles, battery life, frequency spectrum are some examples of such resources. We find different assignment instruments described earlier to be used in different aspects of network communications as well. *Market price* differentiate different data packages to the users. Network administrators use *command* mechanism to delegate the resources. *Majority rule* is used to select cluster head in wireless sensor networks. FIFO queues of router showcase the first-come-first-served mechanism while random access in Medium Access Control (MAC) protocols are using the *lottery* method in a sense. Similarly, QoS provisioning utilize the *personal characteristics* based assignment instrument.

At each layer of the communication stack, we find various resources that are distributed to participating entities. At the processor level, time scheduling is done to allow multiple individuals and programs to share a common computing system. At the link layer, different access methods control how a host can gain medium access without colliding with another host's transmission. At the IP layer, assigning different paths to different connections allow for load balancing and other objectives. At the transport layer, flow control, congestion control and multiplexing create controls for resource sharing. Consequently, services at the application layer set their entitlements and specific requirements for justifying different forms of resource assignments.

Communication networks span across heterogeneous administrative boundaries. Thus resource assignment policies are affected by other factors such as institutional and national policies, service level agreements, financial and security considerations, etc. These can often give different result than intended to the passing traffic because of the cascaded policies.

4.3.1 Fairness notions and measures

Along with the development of resource assignment policies in communication networks, various works on their implication on fairness have been done [77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87]. They range from identifying the different realms of resource assignment to network utility maximization. Some attempt to identify fairness as a quality attribute of the network while others segment fairness into specific metrics of performance, interference, allocation. These have led to quantifying fairness with the aid of various statistical measures [88, 89, 90, 91, 92] which can then be used to contrast different policies. Jain's fairness index [88] has been used most often to measure and compare the fairness of resource

assignments. For n users and X_i resource assigned to each, it is calculated as:

$$F(x) = \frac{(\sum_{i=1}^{n} X_i)^2}{n.\sum_{i=1}^{n} X_i^2}$$

In the case of communications networking environment, there are a few prominent notions of fairness upon which resource assignment policies are based. *Max-Min* fairness [93] maximizes the minimum data rate any flow achieves and then to maximize the second lowest data rate a flow achieves and so on. The fair queuing algorithm and the round robin (for equally sized packets under equal channel conditions) achieve max-min fairness. *Maximum Throughput Resource Management* prioritizes capacity allocation to the least expensive (e.g. best signal to noise ratio) flows. While this can starve the expensive flows, it makes sense in wireless scenarios where the difference in cost of flows are significant. *Proportional Fairness* [94] avoids the starvation of expensive flows while trying to maximize the network throughput. It provides a minimum service to all flows and after a minimum service level, the expensive flows obtain a lower service quality. Mathematically, if C is the capacity to be distributed among n participants with x_i share assigned to user i, then proportional fairness ensures:

$$x_i \geq 0, \quad \sum_{i=1}^{n} x_i \leq C,$$

And for any other allocation with x_i^* share assigned to each user,

$$\sum_{i=1}^{n} \frac{x_i^* - x_i}{x_i} \leq 0.$$

The α - *fairness* scheme is a generalization off the proportional fairness notion and can provide different notions based of different values of α. It also approximates the max-min fairness for $\alpha \to \infty$. There is also an utility function that satisfies α - fairness as:

$$U_i(x_i) = p_i \frac{x_i^{1-\alpha}}{1-\alpha} \tag{4.1}$$

where, x_i = share assigned to user i.

Weighting (p_i in equation 4.1) of assignment allow control mechanisms to prioritize the resource assignments in these fairness notions. Also the utility is subjective and varying values of α allows for testing various fairness notions with corresponding utility functions. We analyzed multiple notions of the α - *fairness* with differing weights and compare the resulting outcome using Jain's Index, the results of which are summarized in the next section. Also an analysis is done in Publication IV when differing notions are applied in a cascaded fashion.

4.3.2 Prioritization under various fairness notions

In order to analyze the effect of different weights, two users were distributed 100 units of resources. While keeping weights(p_i) for *User1* constant at 1 (p_1=1), weight for *User2* was varied from 1 to 20, (p_2=[1,...,20]). Using CVXPY solver [95], the resulting resource assignments x_i of the two users (specifically x_1 and x_2) under different α-fairness notions (0.1,) was optimized with the objective of maximizing the sum of the utilities. The resulting distribution was obtained as in Figure 4.3.

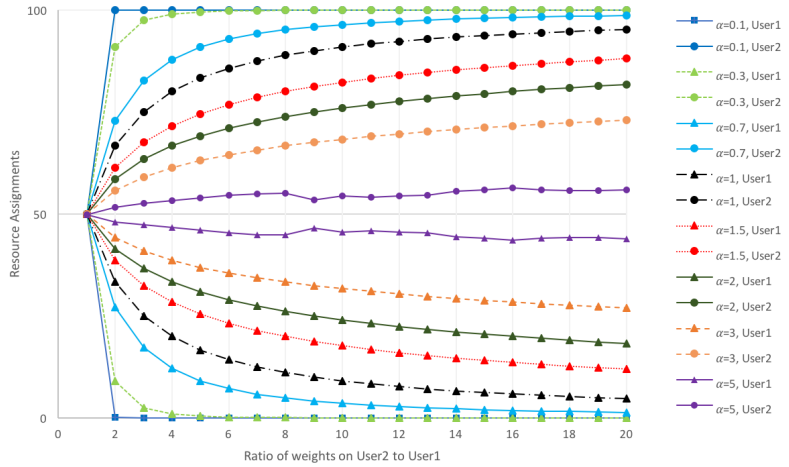


Figure 4.3. Different α-values react differently to weights variation

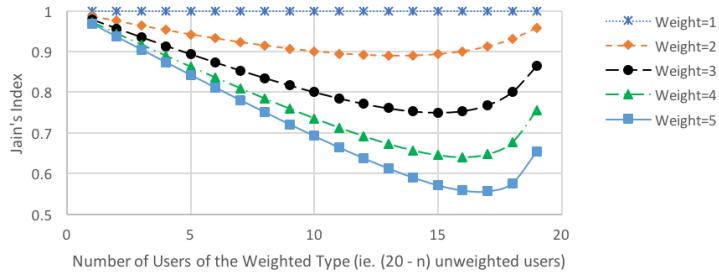
Under the same weights i.e. 1, all values of α assign equal resource (50 units) to each user. As the ratio of weight to User 2 increases, more resources get assigned User 2. For smaller values of α, the weights control the resource assignment very strongly, while for bigger values of α the weights have less impact. This shows that prioritization can perform differently under various fairness notions.

4.3.3 Fairness and prioritization under different utility functions

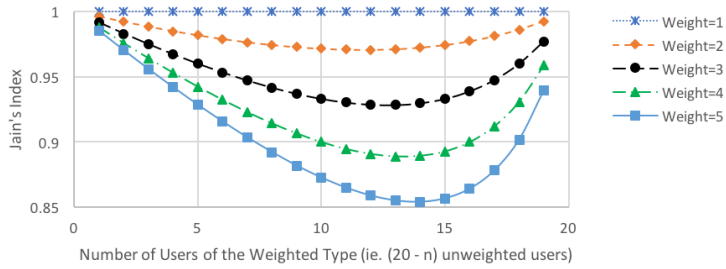
An analysis follows where we look into two different fairness notions scenarios with varying weights. In many cases different utilities are derived from different flows that exist in varying proportions in the network. In order to assess the impact of applying weights or prioritizing some flows, analysis was done with a scenario where 2000 units of resource are to be assigned among 20 users/flows which are of two distinct types (for example short vs long flows).

Two fairness notions were chosen. α=1, following the proportional fairness notion with the corresponding log utility function and α=2, a minimum potential delay fairness scheme which turns the weighted α-fairness utility function into: $U_i(x_i) = -p_i \frac{1}{x_i}$. If the flow utility is related to the delay of the flow (which is

proportional to the flow completion time) and the resource assignment problem is to minimize the delay for those flows, prioritizing such flows implies that they are assigned higher weights hence get more resources leading to lesser delay than other flows. We ran the optimization with different weights applied to different proportions of each flow type under the two different utility functions. We calculated the Jain's index of the resulting assignment as shown in Figure 4.4.



(a) Under $\alpha = 1$



(b) Under $\alpha = 2$

Figure 4.4. Jain's index for different proportions of weighted flow types under different fairness notions

We observe that for the same weight (i.e. no discrimination), the resulting assignment has obvious Jain's index of 1 for all proportions in the mix. As the weights to prioritized flows increases, the position of least value of Jain's index begins to skew towards the right as we increase the proportions of weighted flow type. i.e. worst inequality case is dependent in the weight factor as well. On top of that, observing the difference between Figure 4.4 (a) and (b), the severity of inequality under α=1 is seen to be more prominent (extreme being 0.55 for weight = 5) than under α=2 (extreme being 0.85 for weight = 5). These dependence of fairness on prioritization intensity and chosen utility function make the resource assignment problem more challenging.

Considering the prioritization of short flows using the case α=2, Figure 4.4 (b) shows possibility for one to assign appropriate weights to implement such short flows favoring policies with the knowledge of proportions of user types and

tolerable levels of Jain's index values.

4.3.4 Cascading of fairness notions

Along the communication path and network stack, resource assignment policies are applied which can stem from different fairness notions. For example two nodes (X and Y) can have two different policies ($\alpha 1$ and $\alpha 2$) as in Figure 4.5. This cascading effect can end up with a different resulting allocation than originally envisioned. We analyze such a scenario with node X implementing $\alpha 1$ =1 and node Y implementing $\alpha 2$ = 2, that respectively translate to corresponding utility functions as $U_i = p_i \log(x_i)$ and $U_i = -p_i \frac{1}{x_i}$.

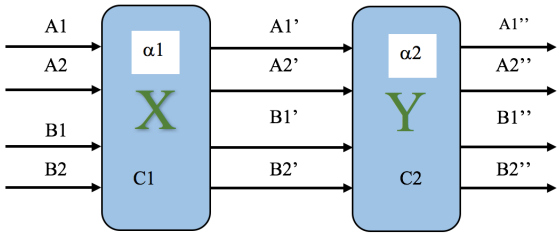


Figure 4.5. Cascade of fairness policies

20 participants, 10 of each type (A1,..., A10, B1,..., B10) are to be assigned maximum of C1=200 and C2=100 resource units at node X and Y respectively. Also type B is given double the weights than type A participant. Assignment vectors are calculated using the CVXPY solver and corresponding Jain's indices are also calculated and presented in Table 4.1. For example, let us take the case shown in the first cell; the case of $\alpha 1$=0.2 and $\alpha 2$=0.2. The 200 resources are assigned among the 20 users (10 of each weight type) at node X according to $\alpha 1$=0.2, and at node Y according to $\alpha 2$=0.2. Each node optimizes its allocation. The difference at node Y is that there is an additional constraint i.e. the maximum resource assigned is capped to the assignment resulting from node X for each participant. Thus as a result, each user of type A gets 0.307 and each user of type B gets 9.692 units of resource and the resulting Jain's index of this assignment is 0.532.

Table 4.1 (highlighted cells) show that for $\alpha 1$=2 and $\alpha 2$=1, the Jain's index is 0.9 and when the alpha values (fairness notion) in the nodes are switched (i.e. $\alpha 1$=1 and $\alpha 2$=2), we get the Jain's index as 0.971. In general we see that $J_{ij} \neq J_{ji}$. The inequality of final assignment when we switch the placement of the policies only reveals that distributive policies (i.e. fairness notions) do not hold commutative property.

Additionally, we see that $\alpha 2$ is mostly dominant in determining the resulting assignment. In particular, for all values of $\alpha 1 > 1$, the resulting assignment depends only on the value of $\alpha 2$. Since node Y has less capacity, it becomes the bottleneck and the policy applied there i.e. $\alpha 2$ is dominant in determining how much resource is assigned to each traffic flow. But for values of $\alpha 1 < 1$,

		$\alpha 1$						
		0.2	0.5	1	1.5	2	5	10
$\alpha 2$	0.2	(0.307,9.692) 0.532	(0.615,9.385) 0.565	(0.615,9.385) 0.565	(0.615,9.385) 0.565	(0.615,9.385) 0.565	(0.615,9.385) 0.565	(0.615,9.385) 0.565
	0.5	(0.307,9.692) 0.532	(2.009,7.99) 0.737	(3.34,6.66) 0.901	(3.869,6.13) 0.951	(4.019,5.981) 0.963	(4.019,5.981) 0.963	(4.019,5.981) 0.963
	1	(0.307,9.692) 0.532	(2.009,7.99) 0.737	(3.34,6.66) 0.901	(3.869,6.13) 0.951	(4.146,5.854) 0.972	(4.655,5.344) 0.995	(4.827,5.172) 0.999
	1.5	(0.307,9.692) 0.532	(2.009,7.99) 0.737	(3.34,6.66) 0.901	(3.869,6.13) 0.951	(4.146,5.854) 0.972	(4.655,5.344) 0.995	(4.827,5.172) 0.999
	2	(0.307,9.692) 0.532	(2.009,7.99) 0.737	(3.34,6.66) 0.901	(3.869,6.13) 0.951	(4.146,5.854) 0.972	(4.655,5.344) 0.995	(4.827,5.172) 0.999
	5	(0.307,9.692) 0.532	(2.009,7.99) 0.737	(3.34,6.66) 0.901	(3.869,6.13) 0.951	(4.146,5.854) 0.972	(4.655,5.344) 0.995	(4.827,5.172) 0.999
	10	(0.307,9.692) 0.532	(2.009,7.99) 0.737	(3.34,6.66) 0.901	(3.869,6.13) 0.951	(4.146,5.854) 0.972	(4.655,5.344) 0.995	(4.827,5.172) 0.999

Table 4.1. Cascade matrix of different α values and corresponding assignments and Jain's indices

this is not true. For $\alpha 1$ =0.2, the inequality persists even for higher values of $\alpha 2$. This is due to the fact that node X's policy has heavily biased towards weighted flows already. Thus, how much each node's policy determines the net resulting assignment is an interplay between the different values of $\alpha 1$ and $\alpha 2$, the prioritization intensity and, of course, the maximum available resources at each node.

This has significant impact and adds further complexities in creating network policies. Due to the cascading effect, the fairness notions upon which different resource assignment policies are built at different network nodes as well as different layers of the networking stack can result into a composite notion. Such composite notions need not be the desired notion for a given application or service. Furthermore, such resulting composite notion might not be the desired notion from socio-economic or socio-political perspective on that administrative realm. The fact that these notions are non-commutative adds another angle while addressing this issue of the end-to-end bidirectional network traffic.

Cascading manifests in communication networks in several places. Traffic flows through various shared links and administrative boundaries which implement different notions. Wireless links on first/last hop of the network path can have proportional fairness while backbone router queues can have max-min fairness notions. A flow passing through such an end-to end path can then be subjected to a composite notion which is not clearly one or the other. Uplink and downlink traffic are prioritized differently by ISPs which can further affect the resulting assignment. To ensure that the intended results are obtained in resource assignment processes, it is important to be aware of the cascading effects.

4.4 Summary and Design Guidelines

One can easily observe that given the numerous methods of assignment, notions of fairness, measures of evaluating such assignment, realms of implementation, multi-party involvement and the subjective nature of utility of resources to a user, it is a complex task to design policies of assigning resources. Many factors need to be considered to provide a satisfactory level of service, without incurring envy among the participants. Nevertheless, just distribution is required and fairness is called upon often to scrutinize the resource assignment policies embodied by the prioritization and distribution mechanisms/ instruments.

While satisfying user needs lie on one side, efficiency is sought by the network operators for full utilization of their infrastructures. A suitable trade-off of these two is sought to maximize overall social welfare usually by authority bodies. Decoupling the preferences of these three parties help in getting a clearer picture of their actions, while a holistic view has to be kept simultaneously to monitor the stability and equilibrium when these forces interact.

We attempt to provide pointers for network architects, designers and admin-

istrators while considering the distributive justice provided by communication infrastructures. These are the considerations for design of resource allocation schemes as is prescribed in Publication IV.

Primary needs first: Basic connectivity, security, robustness, etc. are most crucial aspects for the network. For ordinary users, it is difficult to perceive the assignment policy without specific evaluation tools. But distributive justice will eventually be called upon after those needs are met, thus considering resource allocation policies and the fairness measures alongside will help build foresight.

Entities and utility: Participants of the assignment procedure do matter. Applications have varying needs and network regimes have varying constraints. Private and public networks operate with their own agreed values that guide the assignment policy while they cumulatively affect the end-to-end service outcome. Peering relationships between networks affect the allocation framework significantly. Equal is not always fair. Different entities have different resource requirements and the derived utilities differ depending on many factors. Thus, policies will have to consider such requirements.

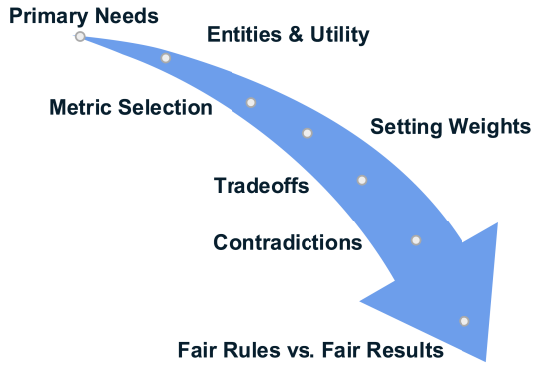


Figure 4.6. Resource Assignment Scheme Design Process

Selection of Metric: Proper metric selection is significant in assigning resources. Network parameters provide varying utilities to participants. Fairness of assignment is associated with each parameter. For example, TCP friendliness concerns with providing flow-rate fairness. Briscoe's argument [29] on fairness mechanisms derived from the cost of a users action to others leads to a different sort of metric formulation. Taking bytes instead of packets to design congestion management schemes have been proposed [96]. Thus different metrics can be used while justifying fairness of distribution and they will have consequent impact on the service resulting from the policy.

Setting appropriate weights: We highlighted the significance of weights to prioritize participants of resource assignment policy. Weights can lead to unequal assignment of resources under fairness notions. Appropriateness of weights can be a different problem of its own depending on the context of resource assignment. It is best if suitable parameters can be selected that act as weights automatically for the participating individuals.

Trade-off exists: One can build policies that allow for starvation of expensive flows in order to achieve spectral or some other resource's efficiency. Such unrestrained optimization of can lead to extreme unfairness. Also attempting a thorough inspection of all the attributes of the participating entities require heavy computation. Thus juggling between fairness, efficient utilization and overall welfare to obtain a balanced and stable trade-off is key in designing resource allocation schemes.

Contradicting policies: One should be aware of contradicting policies and the resulting situation. Random access provides certain kind of distribution policy while priority-based access provides a different kind. If they are cascaded in the different layers or through different middle-boxes it morphs into quite different overall policy.

Fair Rules Vs. Fair Results: Two contrasting perceptions come into play in this regards: 'It is not fair if the result is not fair' vs. 'It is not fair if the rules are not fair'. Not everyone is fair minded and/or with similar capabilities and/or at the same starting level, thus fair rules might not lead to fair results. While on the other hand, ensuring fair results will require more effort and authoritative policies. It helps to establish first and foremost whether the network policy is driven by 'fair rules' or 'fair result'.

Following these steps to design and administer communication networks will help improve the distributive justice provided by the network. The objective of an optimal resource assignment in communication networks is a moving target, but one can come closer to satisfying the fairness needs with due consideration to the different aspects, involved parameters and informed contextual judgment.

5. Summary & Discussions

The research presented in this doctoral thesis focuses on analyzing the principles of resource allocation mechanisms in network data transport. It challenges the flow-length agnostic bandwidth allocation and demonstrates how an alternative data transport protocol would perform. The results of this research are reported in Publication I - Publication IV

The motivations for a short flow favoring data transport are presented in Chapter 1. The benefits of improving user experience and energy efficiency are described. Also the need for clearly understood fairness notions is highlighted and how short flows can be prioritized under some notions are described.

The design and implementation of a short flow favoring data transport algorithm (FLD_TCP) is presented in Chapter 2. Slow start and congestion avoidance algorithms of TCP were modified to favor the transport of short flows. The evaluation results of the new TCP is described in Chapter 3. The simulations test showed promising results which led to actual Linux implementation and evaluation of real file transfers. Additionally the performance improvement to web page loading in modern browsers was reported using page loads of top websites via a FLD_TCP equipped proxy.

Chapter 4 delves into the holistic and pragmatic view to the issues of distributive justice in network resources. Discussions on the origins and development of fairness notions in the society at large was tied to the resource assignment schemes in communication networks. Using the α-fairness paradigm, the effect of prioritization in assigned resource amounts under different fairness notions as well as under different proportions of participants were analyzed. The non-commutative property of resource assignment schemes when they occur in cascades (which is often the case in communication networks) was demonstrated. By bringing out such issues of distributive justice in network communications a set of guidelines was prescribed to help network architects, designers and administrator to implement appropriate systems.

5.1 On Improving FLD_TCP

As stated in the FLD_TCP design section in Chapter 2, FLD_TCP developed in this work does not implement safeguards against the overshoot of sending rate during slow-start. Algorithms developed via various works [55, 56, 57, 58, 53] can refine FLD_TCP's performance and help avoid the wasteful retransmissions during slow-start overshoot.

In our implementation, we set the flow length threshold value to be 2 MB. The threshold value was derived from empirical data regarding Internet flow size distributions. One significant improvement for FLD_TCP would be to replace the threshold value with something more scalable such as context based dynamic values. In such a scenario, the context could perhaps be obtained from other layers. As an example, QUIC has been shown to allow for such context inference such as bandwidth estimates from client side [97].

While ours was only a sender-side algorithm in its implementation, the underlying principle of the algorithm is in fact not tied to TCP only. It can be implemented in other transport layer protocols and even be exported to other network layers. For example, in a QUIC-like operation model, such flow length dependent treatment can be implemented into the stream multiplexing logic. And, if the congestion state is shared across multiple streams within a connection, it could provide nested policies, affecting connection-level as well as stream-level prioritization. One could even further expand such flow-length dependent treatment of flows at different network nodes or even as a distributed algorithm.

5.2 On Network Data Transport

The speed by which communication networks are evolving is phenomenal. The Internet data transport landscape has seen many changes throughout the duration of this research and the trend is likely to continue. This means that the traffic interactions in the Internet will grow more complex as various stakeholders of the networks tussle to morph the hourglass view [98] of the Internet to their will. In many cases, the original requirements for the communications networks infrastructures are insufficient and the original principles upon which the network protocols were built are no longer valid. The End-to-End principle [99] does not completely address today's federated networks of giant cloud providers and their data centers and Content Delivery Network (CDN)s along with paradigm shifting technologies such as Edge computing [100], Software-Defined Networking (SDN) [101] and Network Functions Virtualization (NFV) [102].

This opens discussions on the new ways of thinking data transport on the Internet. Many developments are happening that directly or indirectly affect how services get assigned network resources for their data transport. The prolif-

eration of content delivery networks have brought content closer to the end user reducing the latency. On top of the traditional CDN providers such as Akamai and Limelight, we now are also seeing content providers themselves (such as Netflix) take initiatives to push this paradigm [103]. With technologies such as edge computing and cloudlets that enable the Fifth Generation (5G) networks, the network path between the end hosts of data transport is getting shorter and also restricted within a more local administrative boundary. This implies that the cascades of resource allocation nodes would be more manageable.

To use an aggressive data transport protocol for all flows in the Internet would not be recommendable as it would lead to "the tragedy of the commons" via congestion collapse. Regardless, with a selective prioritization we can still gain the benefits of improved user experience and potential savings in energy consumption of wireless devices with a stable behavior. FLD_TCP shows the potential to enable such improvements.

QoS mechanisms allow for selective prioritization but the basis and justification of such prioritization needs rigorous deliberation by the stakeholders. The issues of Net Neutrality have demonstrated that out of the many challenges in this area, technical challenge is perhaps the easiest one. Certain types of preferential treatment of data flows in the Internet are deemed acceptable by the society, such as those for traffic optimization and service requirements.

On the other hand, questions arise if data transfers via Multipath TCP and P2P are indeed workarounds to obtain more than fair share of the network resources. Even when HTTP pipelining or Multiplexing is used, the advantage from sharing the connection setup cost among piggybacking data-transfer provides an advantage compared to other flows that do not have such enabled. As the Internet evolves, answers from a distributive justice perspective are sought. Solutions and their implementations will be milestones to look forward to.

We are also witnessing some developments in decoupling of congestion control from transport layer so that they can be optionally and quite easily plugged in via the upper layers. This implies congestion control implementations are moving from operating system to browsers or similar applications. Any application developer can implement congestion control algorithms of his/her choosing (or embodying their notion of resource allocation fairness) and plug into their applications. Examples would be Google's development of congestion control mechanisms for Real-Time Communication in browsers and mobile applications (WebRTC) [104] as well as for QUIC [105]. For big players in the ecosystem that have dominant market penetration of both server and client side applications, this makes in-house build and release of such new congestion control algorithms (incorporating the will of said player) faster and easier. With no safeguards from the operating system, the resource allocation policies baked into the network traffic flows in such a scenario will be significantly more heterogeneous and the tussles among them more dynamic. Such an ecosystem in the communication networks definitely warrants scrutiny from various interest groups, from policymakers and governance to service providers and network providers.

Decision makers as well as the society at large need to decide on their selection of fairness notions and the resulting development and implementation of resource assignment schemes at different areas of communication networks. And for them, the research world needs to provide better tools and analysis so that sound and actionable principles can be based upon. The results and some techniques used here can be applied, perhaps in the design of future networks, to provide better services to the user and to efficiently utilize the resources available to the network.

References

[1] Netcraft. *February 2019 Web Server Survey*. Available at: https://news.netcraft.com/archives/2019/02/28/february-2019-web-server-survey.html. [Accessed: Mar. 26, 2019].

[2] Recommendation ITU-T. *X. 200: Information Technology–Open Systems Interconnection–Basic Reference Model: The Basic Model*. International Telecommunication Union. 1994.

[3] R. Braden. *Requirements for Internet Hosts - Communication Layers*. RFC 1122. IETF, Oct. 1989. URL: http://www.rfc-editor.org/rfc/rfc1122.txt.

[4] R. Fielding and J. Reschke. *Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing*. RFC 7230. IETF, June 2014. URL: http://www.rfc-editor.org/rfc/rfc7230.txt.

[5] R. Stewart. *Stream Control Transmission Protocol*. RFC 4960. IETF, Sept. 2007. URL: http://www.rfc-editor.org/rfc/rfc4960.txt.

[6] A. Ford et al. *TCP Extensions for Multipath Operation with Multiple Addresses*. RFC 6824. IETF, Jan. 2013. URL: http://www.rfc-editor.org/rfc/rfc6824.txt.

[7] S. Floyd, E. Kohler, and J. Padhye. *Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 3: TCP-Friendly Rate Control (TFRC)*. RFC 4342. IETF, Mar. 2006. URL: http://www.rfc-editor.org/rfc/rfc4342.txt.

[8] T. Dierks and E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.2*. RFC 5246. IETF, Aug. 2008. URL: http://www.rfc-editor.org/rfc/rfc5246.txt.

[9] E. Rescorla and N. Modadugu. *Datagram Transport Layer Security Version 1.2*. RFC 6347. IETF, Jan. 2012. URL: http://www.rfc-editor.org/rfc/rfc6347.txt.

[10] H. Schulzrinne et al. *RTP: A Transport Protocol for Real-Time Applications*. RFC 3550. IETF, July 2003. URL: http://www.rfc-editor.org/rfc/rfc3550.txt.

[11] R. Fielding and J. Reschke. *Hypertext Transfer Protocol (HTTP/1.1): Authentication*. RFC 7235. IETF, June 2014. URL: http://www.rfc-editor.org/rfc/rfc7235.txt.

[12] M. Belshe, R. Peon, and M. Thomson. *Hypertext Transfer Protocol Version 2 (HTTP/2)*. RFC 7540. IETF, May 2015. URL: http://www.rfc-editor.org/rfc/rfc7540.txt.

[13] M. Bishop. *Hypertext Transfer Protocol Version 3 (HTTP/3)*. Internet-Draft, (Work in Progress), draft-ietf-quic-http-19. IETF, Mar. 2019. URL: http://www.ietf.org/internet-drafts/draft-ietf-quic-http.

[14] J. Roskind. *Multiplexed Stream Transport over UDP*. Available at: https://docs.google.com/document/d/1RNHkx_VvKWyWg6Lr8SZ-saqsQx7rFV-ev2jRFUoVD34. QUIC, edited Dec. 2013. [Accessed: Dec. 8, 2019].

[15] J. Wroclawski. *The Use of RSVP with IETF Integrated Services*. RFC 2210. IETF, Sept. 1997. URL: http://www.rfc-editor.org/rfc/rfc2210.txt.

[16] K. Nichols et al. *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*. RFC 2474. IETF, Dec. 1998. URL: http://www.rfc-editor.org/rfc/rfc2474.txt.

[17] R. Kohavi and R. Longbotham. *Online Experiments: Lessons Learned*. In: *Computer* Vol. 40.No. 9 (2007), pp. 103–105. IEEE.

[18] P. R. Selvidge, B. S. Chaparro, and G. T. Bender. *The World Wide Wait: Effects of Delays on User Performance*. In: *International Journal of Industrial Ergonomics* Vol. 29.No. 1 (2002), pp. 15–20. Elsevier.

[19] J. A. Jacko, A. Sears, and M. S. Borella. *The Effect of Network Delay and Media on User Perceptions of Web Resources*. In: *Behaviour & Information Technology* Vol. 19.No. 6 (2000), pp. 427–439. Taylor & Francis.

[20] V. Aggarwal et al. *Characterizing Fairness for 3G Wireless Networks*. In: *2011 Proc. of Workshop on Local Metropolitan Area Networks (LANMAN)*. 2011, pp. 1–6. IEEE.

[21] Httparchive. *Page Weight*. Available at: https://httparchive.org/reports/page-weight. [Accessed: Mar. 26, 2019].

[22] N. Brownlee and K.C. Claffy. *Internet Stream Size Distributions*. In: *SIGMETRICS Performance Evaluation Review* Vol. 30.No. 1 (2002), pp. 282–283. ACM.

[23] A. Shehabi et al. *United States Data Center Energy Usage Report*. Tech. rep. Lawrence Berkeley National Laboratory, June 2016.

[24] N. Sklavos and K. Touliou. *Power Consumption in Wireless Networks: Techniques & Optimizations*. In: *2007 Proc. of The International Conference on "Computer as a Tool"*. 2007, pp. 2154–2157. IEEE.

[25] K. C. Barr and K. Asanović. *Energy-Aware Lossless Data Compression*. In: *Transactions on Computer Systems (TOCS)* Vol. 24.No. 3 (2006), pp. 250–291. ACM.

[26] S. Goel and T. Imielinski. *Etiquette Protocol for Ultra Low Power Operation in Sensor Networks*. Tech. rep. DCS-TR-552DCS-TR-552, CS Dept., Rutgers University, 2004.

[27] K. Lan and J. Heidemann. *On the Correlation of Internet Flow Characteristics*. Tech. rep. ISI-TR-574, USC/Information Sciences Institute, 2003.

[28] S. Floyd and K. Fall. *Promoting the Use of End-to-End Congestion Control in the Internet*. In: *Transactions on Networking* Vol. 7.No. 4 (1999), pp. 458–472. IEEE/ACM.

[29] B. Briscoe. *Flow Rate Fairness: Dismantling a Religion*. In: *SIGCOMM Computer Communication Review* Vol. 37.No. 2 (2007), pp. 63–74. ACM.

[30] D. R. Smith. *A New Proof of the Optimality of the Shortest Remaining Processing Time Discipline*. In: *Operations Research* Vol. 26.No. 1 (1978), pp. 197–199. INFORMS.

[31] M. Mathis. *Heresy following TCP: Train-wreck*. http://oakham.cs.ucl.ac.uk/pipermail/iccrg/2008-April/thread.html. Note sent to the ICCRG mailing list. [Accessed: Mar. 26, 2019].

[32] J. Semke, J. Mahdavi, and M. Mathis. *Automatic TCP Buffer Tuning*. In: *SIGCOMM Computer Communication Review* Vol. 28.No. 4 (Oct. 1998), pp. 315–323. ACM.

[33] D. X. Wei and P. Cao. *NS-2 TCP-Linux: an NS-2 TCP Implementation with Congestion Control Algorithms from Linux*. In: *2006 Proc. of workshop on ns-2: the IP network simulator*. 2006, p. 9. ACM.

[34] J. Nagle. *Congestion Control in IP/TCP Internetworks*. RFC 896. IETF, Jan. 1984. URL: https://www.rfc-editor.org/rfc/rfc896.txt.

[35] M. Mathis et al. *TCP Selective Acknowledgment Options*. RFC 2018. IETF, Oct. 1996. URL: https://www.rfc-editor.org/rfc/rfc2018.txt.

[36] D. Borman et al. *TCP Extensions for High Performance*. RFC 7323. IETF, Sept. 2014. URL: https://www.rfc-editor.org/rfc/rfc7323.txt.

[37] F. Baker and G. Fairhurst. *IETF Recommendations Regarding Active Queue Management*. RFC 7567. IETF, July 2015. URL: https://www.rfc-editor.org/rfc/rfc7567.txt.

[38] K. Nichols and V. Jacobson. *Controlling Queue Delay*. In: *Queue* Vol. 10.No. 5 (May 2012), pp. 20–34. ACM.

[39] K. Ramakrishnan, S. Floyd, and D. Black. *The Addition of Explicit Congestion Notification (ECN) to IP*. RFC 3168. IETF, Sept. 2001. URL: http://www.rfc-editor.org/rfc/rfc3168.txt.

[40] V. Jacobson. *Congestion Avoidance and Control*. In: *SIGCOMM Computer Communication Review* Vol. 18.No. 4 (Aug. 1988), 314–329. ACM.

[41] M. Allman, S. Floyd, and C. Partridge. *Increasing TCP's Initial Window*. RFC 3390. IETF, Oct. 2002. URL: https://www.rfc-editor.org/rfc/rfc3390.txt.

[42] J. Chu et al. *Increasing TCP's Initial Window*. RFC 6928. IETF, Apr. 2013. URL: https://www.rfc-editor.org/rfc/rfc6928.txt.

[43] K. Jacobsson et al. *Round-Trip Time Estimation in Communication Networks using Adaptive Kalman Filtering*. 2004 Proc. of Reglermöte. 2004.

[44] M. Mathis. *Relentless Congestion Control*. Internet-Draft, (Work in Progress), draft-mathis-iccrg-relentless-tcp-00. IETF, Mar. 2009. URL: https://tools.ietf.org/html/draft-mathis-iccrg-relentless-tcp-00.

[45] S. J. Leffler, M. J. Karels, and M. K. McKusick. *The Design and Implementation of the 4.3 BSD UNIX Operating System*. Reading, MA: Addison-Wesley, 1989.

[46] L. Xu, K. Harfoush, and I. Rhee. *Binary Increase Congestion Control (BIC) for Fast Long-Distance Networks*. In: *2004 Proc. of INFOCOM Conference on Computer Communications Workshops*. Vol. 4. Mar. 2004, pp. 2514–2524. IEEE.

[47] S. Ha, I. Rhee, and L. Xu. *CUBIC: A New TCP-Friendly High-Speed TCP Variant*. In: *SIGOPS Operating Systems Review* Vol. 42.No. 5 (2008), pp. 64–74. ACM.

[48] N. Cardwell et al. *BBR: Congestion-Based Congestion Control*. In: *Queue* Vol. 14.No. 5 (Oct. 2016), 20–53. ACM.

[49] S. Floyd et al. *Quick-Start for TCP and IP*. RFC 4782. IETF, Jan. 2007. URL: https://tools.ietf.org/rfc/rfc4782.txt.

[50] D. Liu et al. *Congestion Control Without a Startup Phase*. In: *2007 Proc. of International Workshop on Protocols for Fast Long-Distance Networks*. Feb. 2007. PFLDnet.

[51] Q. Li, M. Dong, and P. Godfrey. *Halfback: Running short flows quickly and safely*. In: *2015 Proc. of Conference on Emerging Networking Experiments and Technologies*. 2015, p. 22. ACM.

[52] S. Floyd. *Limited Slow-Start for TCP with Large Congestion Windows*. RFC 3742. IETF, Mar. 2004. URL: https://tools.ietf.org/rfc/rfc3742.txt.

[53] S. Ha and I. Rhee. *Taming the Elephants : New TCP Slow Start*. In: *Computer Networks: The International Journal of Computer and Telecommunications Networking* Vol. 55.No. 9 (2011), pp. 1–19. Elsevier.

[54] P. Balasubramanian, Y. Huang, and M. Olson. *HyStart++: Modified Slow Start for TCP*. Internet-Draft, (Work in Progress), draft-balasubramanian-tcpm-hystartplusplus-01. IETF, July 2019. URL: http://www.ietf.org/internet-drafts/draft-balasubramanian-tcpm-hystartplusplus-01.txt.

[55] A. Aggarwal, S. Savage, and T. Anderson. *Understanding the performance of TCP pacing*. In: *2000 Proc. of INFOCOM Conference on Computer Communications*. Vol. 3. 2000, pp. 1157–1165. IEEE.

[56] D. Wei et al. *TCP Pacing Revisited*. In: *2006 Proc. of INFOCOM Conference on Computer Communications*. Vol. 2. 2006, p. 3. IEEE.

[57] V. Konda and J. Kaur. *RAPID: Shrinking the Congestion-Control Timescale*. In: *2009 Proc. of INFOCOM Conference on Computer Communications*. 2009, pp. 1–9. IEEE.

[58] J. Misund and B. Briscoe. *Paced Chirping: Rapid Flow Start with very Low Queuing Delay*. In: *2019 Proc. of INFOCOM Conference on Computer Communications Workshops*. Apr. 2019, pp. 798–804. IEEE.

[59] C. Jin, D. X. Wei, and S. H. Low. *FAST TCP: Motivation, Architecture, Algorithms, Performance*. In: *2004 Proc. of INFOCOM Conference on Computer Communications*. Vol. 4. 2004, pp. 2490–2501. IEEE.

[60] A. Venkataramani, R. Kokku, and M. Dahlin. *TCP Nice: A Mechanism for Background Transfers*. In: *SIGOPS Operating Systems Review* Vol. 36.No. SI (2002), pp. 329–343. ACM.

[61] A. Kuzmanovic and E. W. Knightly. *TCP-LP: A Distributed Algorithm for Low Priority Data Transfer*. In: *2003 Proc. of INFOCOM Conference on Computer Communications*. Vol. 3. 2003, pp. 1691–1701. IEEE.

[62] S. Shalunov et al. *Low Extra Delay Background Transport (LEDBAT)*. RFC 6817. IETF, Dec. 2012. URL: http://www.rfc-editor.org/rfc/rfc6817.txt.

[63] Alexa. *The Top 500 Sites on the Web*. Available at: http://www.alexa.com/topsites. [Accessed Mar. 26 2019].

[64] S. Pokhrel and G.R. Moktan. *FLD_TCP*. https://github.com/grmoktan/FLDTCP [Accessed: Mar. 16, 2020]. 2014.

[65] S. Hemminger. *Network Emulation with NetEm*. In: *2005 Proc. of Linux Conference Australia*. 2005, pp. 18–23. linux.conf.au.

[66] A. Tirumala et al. *iPerf - The Ultimate Speed Test Tool for TCP, UDP and SCTP*. 2005. URL: https://iperf.fr/. [Accessed: Mar. 16, 2020].

[67] A. Retana. *Network evolution: the Hierarchy of Networking Needs.* Blog, Internet Archive. June 20 2011. URL: http://h30507.www3.hp.com/t5/HP-Networking/Network-evolution-the-Hierarchy-of-Networking-Needs/ba-p/94169. [Accessed: Mar. 26, 2019].

[68] A. H. Maslow. *A Theory of Human Motivation.* In: *Psychological review* Vol. 50.No. 4 (1943), p. 370. American Psychological Association.

[69] D. Clark et al. *Tussle in Cyberspace: Defining Tomorrow's Internet*. In: *SIGCOMM Computer Communication Review*. Vol. 32. 4. 2002, pp. 347–356. ACM.

[70] S. F. Brosnan and F. BM. De Waal. *Monkeys Reject Unequal Pay*. In: *Nature* Vol. 425.No. 6955 (2003), pp. 297–299. Nature Publishing Group.

[71] J. Henrich et al. *Markets, Religion, Community Size, and the Evolution of Fairness and Punishment*. In: *science* Vol. 327.No. 5972 (2010), pp. 1480–1484. American Association for the Advancement of Science.

[72] J. Lamont and C. Favor. *Distributive Justice*. In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. 2012. URL: http://plato.stanford.edu/archives/win2012/entries/justice-distributive/.

[73] J. Rawls. *A Theory of Justice*. Harvard University Press, 2009.

[74] M. Parkin, M. Powell, and K. Matthews. *Economics (7th edn)*. Pearson Education Limited, England, 2008, pp. 102–103.

[75] G. Hardin. *The Tragedy of the Commons*. In: *Journal of Natural Resources Policy Research* Vol. 1.No. 3 (2009), pp. 243–253. Taylor & Francis.

[76] J. L. Coleman. *Efficiency, Utility, and Wealth Maximization*. In: *Hofstra Law Review* Vol. 8.No. 3 (1979), p. 509. URL: https://scholarlycommons.law.hofstra.edu/hlr/vol8/iss3/3/.

[77] J. Wong, J. Sauve, and J. Field. *A Study of Fairness in Packet-Switching Networks*. In: *Transactions on Communications* Vol. 30.No. 2 (Feb. 1982), pp. 346–353. IEEE.

[78] P. K. Varshney and D. Surajit. *Fairness in Computer Networks: A Survey*. In: *IETE Journal of Research* Vol. 36.No. 5-6 (1990), pp. 440–445. Taylor & Francis.

[79] H. Shi et al. *Fairness in Wireless Networks: Issues, Measures and Challenges*. In: *Communications Surveys & Tutorials* Vol. 16.No. 1 (2014), pp. 5–24. IEEE.

[80] M. Gerla, H.W. Chan, and J.R.B. De Marca. *Fairness in Computer Networks*. In: *1985 Proc. of International Conference on Communications*. 1985, pp. 1384–1389. IEEE.

[81] J. Jaffe. *Bottleneck Flow Control*. In: *Transactions on Communications* Vol. 29.No. 7 (1981), pp. 954–962. IEEE.

[82] C. Douligeris and L.N. Kumar. *Fairness Issues in the Networking Environment*. In: *Computer Communications* Vol. 18.No. 4 (1995), pp. 288–299. Elsevier.

[83] R. Denda, A. Banchs, and W. Effelsberg. *The Fairness Challenge in Computer Networks*. In: *Quality of Future Internet Services*. 2000, pp. 208–220. Springer.

[84] R. Hwang and C. Chi. *Fairness in QoS guaranteed networks*. In: *2003 Proc. of International Conference on Communications* Vol. 1 (2003), pp. 218–222. IEEE.

[85] A. Kumar and J. Kleinberg. *Fairness Measures for Resource Allocation*. In: *Proc. 41st Annual Symposium on Foundations of Computer Science*. 2000, pp. 75–85. IEEE.

[86] P. Nilsson. *Fairness in Communication and Computer Network Design*. PhD thesis. Lund University, 2006.

[87] D.M. Chiu and A.S.W. Tam. *Fairness of Traffic Controls for Inelastic Flows in the Internet*. In: *Computer Networks* Vol. 51.No. 11 (2007), pp. 2938–2957. Elsevier.

[88] R. Jain, D.M. Chiu, and W. R. Hawe. *A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer Systems*. Vol. 38. Eastern Research Laboratory, Digital Equipment Corporation Hudson, MA, 1984.

[89] Z. Chen and C. Zhang. *A New Measurement for Network Sharing Fairness*. In: *Computers & Mathematics with Applications* Vol. 50.No. 5-6 (Sept. 2005), pp. 803–808. Elsevier.

[90] Chen, Z. and Zhang, C. *Network Sharing Fairness Function with Consideration of Priority Levels*. In: *Computers & Mathematics with Applications* Vol. 60.No. 9 (2010), pp. 2548–2555. Elsevier.

[91] S. Floyd. *Metrics for the Evaluation of Congestion Control Mechanisms*. RFC 5166. IETF, Mar. 2008. URL: http://www.rfc-editor.org/rfc/rfc5166.txt.

[92] T. Lan et al. *An Axiomatic Theory of Fairness in Network Resource Allocation*. In: *2010 Proc. of INFOCOM Conference on Computer Communications*. San Diego, California, USA, 2010, 1343–1351. IEEE.

[93] A. Demers, S. Keshav, and S. Shenker. *Analysis and Simulation of a Fair Queueing Algorithm*. In: *SIGCOMM Computer Communication Review*. Vol. 19. 4. 1989, pp. 1–12. ACM.

[94] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan. *Rate Control for Communication Networks: Shadow Prices, Proportional Fairness and Stability*. In: *Journal of the Operational Research Society* Vol. 49.No. 3 (Apr. 1998), pp. 237–252. Taylor & Francis.

[95] S. Diamond and S. Boyd. *CVXPY: A Python-Embedded Modeling Language for Convex Optimization*. In: *Journal of Machine Learning Research* Vol. 17.No. 83 (2016), pp. 1–5. Microtome Publishing.

[96] B. Briscoe and J. Manner. *Byte and Packet Congestion Notification*. RFC 7141. IETF, Feb. 2014. URL: http://www.rfc-editor.org/rfc/rfc7141.txt.

[97] J. Rüth et al. *Blitz-starting QUIC Connections*. In: *arXiv e-prints* (May 2019). URL: https://arxiv.org/abs/1905.03144v1.

[98] S. Akhshabi and C. Dovrolis. *The Evolution of Layered Protocol Stacks Leads to an Hourglass-Shaped Architecture*. In: *SIGCOMM Computer Communication Review* Vol. 41.No. 4 (2011), pp. 206–217. ACM.

[99] J. H. Saltzer, D. P. Reed, and D. D. Clark. *End-to-End Arguments in System Design*. In: *Transactions on Computer Systems* Vol. 2.No. 4 (Nov. 1984), 277–288. ACM.

[100] W. Shi et al. *Edge Computing: Vision and Challenges*. In: *Internet of Things Journal* Vol. 3.No. 5 (2016), pp. 637–646. IEEE.

[101] D. Kreutz et al. *Software-Defined Networking: A Comprehensive Survey*. In: *Proceedings of the IEEE* Vol. 103.No. 1 (2015), pp. 14–76. IEEE.

[102] B. Han et al. *Network Function Virtualization: Challenges and Opportunities for Innovations*. In: *IEEE Communications Magazine* Vol. 53.No. 2 (2015), pp. 90–97. IEEE.

[103] T. Böttger et al. *Open Connect Everywhere: A Glimpse at the Internet Ecosystem Through the Lens of the Netflix CDN*. In: *SIGCOMM Computer Communication Review* Vol. 48.No. 1 (2018), pp. 28–34. ACM.

[104] S. Holmer et al. *A Google Congestion Control Algorithm for Real-Time Communication*. Internet-Draft, (Work in Progress), draft-alvestrand-rmcat-congestion-03. IETF, June 2015. URL: https://tools.ietf.org/html/draft-alvestrand-rmcat-congestion.

[105] J. Iyengar and I. Swett. *QUIC Loss Detection and Congestion Control*. Internet-Draft, (Work in Progress), draft-ietf-quic-recovery-18. IETF, Jan. 2019. URL: http://www.ietf.org/internet-drafts/draft-ietf-quic-recovery.

Errata

Publication I

- The indices of Table I and II should read Burst1(512KB), Burst2(1MB), Burst3(1.5MB) and Burst4(2MB)

Publication III

The second author's name has been published as Nutti Varis while it should have been Nuutti Varis.

Publication IV

The figure titled "Different social indifference curves" should include two curves per subfigure as below:

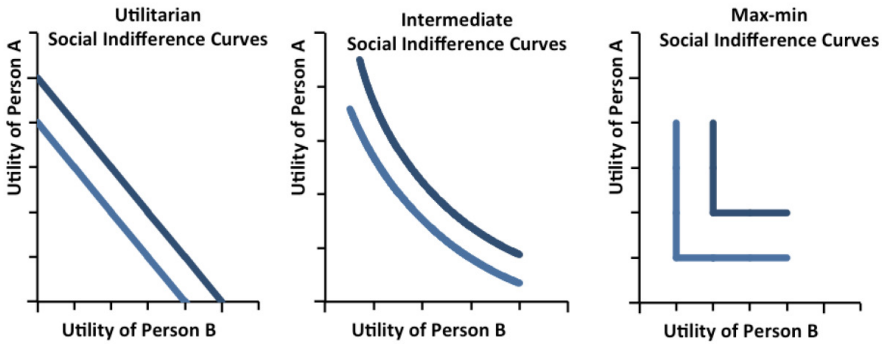


Figure 5.1. Different Social Indifference Curves

Aalto-DD 115/2020

ISBN 978-952-60-3982-4 (printed)
ISBN 978-952-60-3983-1 (pdf)
ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)

Aalto University
School of Electrical Engineering
Department of Communications and Networking
www.aalto.fi

BUSINESS + ECONOMY

ART + DESIGN + ARCHITECTURE

SCIENCE + TECHNOLOGY

CROSSOVER

DOCTORAL DISSERTATIONS