Georg-August-Universität
Göttingen
Fakultät für Mathematik
und Informatik

Network Friendly Congestion Control: Framework, Protocol Design and Evaluation

Dissertation
zur Erlangung des
mathematisch-naturwissenschaftlichen Doktorgrades
“Doctor rerum naturalium”
der Georg-August-Universität Göttingen

Vorgelegt von
Mayutan Arumaithurai
Batticaloa, Srilanka

Göttingen 2010

Referent: Prof.Dr.Xiaoming Fu
Korreferent: Dr. K.K.Ramakrishnan

Tag der mündlichen Prüfung: 22 November 2010

Abstract

The increase in bandwidth, the number of internet users and the variety of internet applications, all point towards the need for adaptive and flexible internet protocols. Alternatively, they also emphasize on the necessity for specialized protocols that cater to application requirements. Current Transport protocols such as TCP and UDP are not flexible enough to be used by applications which differ in their performance expectations. UDP, for example, is inelastic in nature and therefore can cause exacerbation of congestion in an already congested network. TCP on the other hand was designed to provide proportional fairness, such that a "flow" is the unit which is allocated resources. This not only results in all flows having an equal share at the bandwidth irrespective of the application requirements, but also give way to a new trend of applications that are able to take advantage of the "flow-oriented" fairness.

Delay-insensitive applications, such as P2P file sharing, generate substantial amounts of traffic and compete with other applications on an equal footing while using TCP. Further, to optimize throughput, such applications open multiple connections. This results in an unfair and potentially poor service for applications having stringent performance objectives. The main part of this thesis proposes NF-TCP, a TCP variant for P2P and similar background delay-insensitive applications. NF-TCP aims to be submissive to delay-sensitive applications under congestion. It is designed to be network friendly based on a fluid flow model for intermediate queues and uses explicit congestion notification (ECN) for early detection of congestion. Moreover, it exploits the measure of the available bandwidth to be able to aggressively utilize spare capacity.

It can be observed that the traditional congestion control protocols comprise of tightly coupled mechanisms, and are designed to address a specific problem or scenario. They either lack the flexibility in meeting varying requirements or utilizing different types of available network support. This thesis proposes a congestion control framework that consists of pluggable components and utilizes it to guide the NF-TCP design. It identifies congestion-detection, flow-control and bandwidth-estimation as the main components of the congestion control protocol, which can be loosely or strictly decoupled from each other under a modular framework.

This thesis implemented NF-TCP on Linux and ns-2. The evaluations of the NF-TCP Linux implementation on ns-2 show that NF-TCP outperforms other network friendly approaches (e.g., LEDBAT, TCP-LP and RAPID). NF-TCP achieves high utilization, fair bandwidth allocation among NF-TCP flows and maintains a small average queue. The evaluations further demonstrate that with NF-TCP, the available bandwidth can be efficiently utilized for supporting both delay-sensitive and insensitive traffic in a wide range of scenarios.

Acknowledgement

It is my pleasure to thank those who made this thesis possible. My deepest gratitude goes to my supervisor Professor Xiaoming Fu, who has guided and supported me throughout my thesis whilst allowing me to work in my own way.

It has been my privilege to work with Mr. Hannes Tschofenig who has been my mentor since my Master thesis and provided me with the opportunity to pursue my research at Nokia Siemens Networks. I appreciate the help and continuous corporation extended by him throughout my professional experience.

I am grateful to Dr. K.K.Ramakrishnan for his constant encouragement and invaluable suggestions, which have significantly improved my work.

I would like to express my deepest gratitude to my dissertation committee members, Prof Hogrefe, Prof Grabowski, Prof Damm and Prof Wörgötter.

I have immensely enjoyed the company of previous and current members of the Computer Networks group, University of Goettingen. I would like to thank Niklas Neumann, Florian Tegler and Ralf Lübban for their valuable reviews and comments. I appreciate the friendship, discussions and the stimulating working environment. I would also like to thank my students Benedickt Schäffler, Rene Rex and Fabian Glaser for their support.

It was my pleasure to have interacted and collaborated with fellow researchers at IETF meetings and other international conferences. I owe a lot to the research community at large, the bloggers, Hacker News, the open source contributors and all others for providing useful information and tools.

I am indebted to my sponsors, Nokia Siemens Networks and the Computer Networks Group, University of Goettingen, without whom this thesis would have been impossible. I thank Maximilian Riegel, Mehmet Ersue, Dirk Weiler and Jonne Soininen for providing me the opportunity to pursue my research at Nokia Siemens Networks. I would also like to thank Annette, Carmen, Heike and Gunnar for meticulously taking care of all the administrative details without hesitation.

A very special thanks to my wonderful wife to be, Sunanda Dasgupta, for her immense support, understanding and patience. I could not have gotten this far without you. Thank you. I would like to extend my gratitude to my parents and teachers for their love, affection and guidance. A special mention of my wonderful family and friends for their constant encouragement and enthusiasm.

Finally, I thank Germany for providing me with such a wide variety of beer and coffee that kept me sane, of course, not forgetting the excellent quality of life and research opportunity.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The Transmission Control Protocol (TCP) [5] is the most widely used transport layer protocol in the Internet. Popular applications such as peer-to-peer (P2P), video, live-streaming and gaming prefer to use TCP due to its inherent congestion control characteristics and reliable delivery semantics. TCP's reliability is achieved by recovering from packet loss through retransmissions. TCP's congestion control ensures that it reacts to congestion that results in packet loss, by halving its sending rate. This ensures that the flows are able to minimize losses and mitigate congestion. These mechanisms enable TCP to adapt to heterogeneous network environments and varying traffic conditions, while also avoiding congestion. Although TCP and its variants have served the Internet for a long period, they unfortunately have certain limitations too (see Section 1.1).

Different applications have varying requirements which are summarized in Table 1.1. Based on user expectations and the technologies currently available in the Internet, applications may be broadly classified into:

Delay-sensitive applications: Users have higher expectations and less tolerance for delays caused to delay-sensitive applications such as video conferencing and streaming, and even web-browsing.

Delay-insensitive applications: Users consider applications such as software updates, "download and play" and P2P file sharing to name a few, as lower priority. These applications are therefore relatively delay-insensitive.

TCP treats all of these applications in the same way. Although Quality of Service (QoS) in the network could address the needs of the various applications, it has not been widely deployed. This thesis proposes a "Network Friendly" version of TCP to overcome the limitations (see Section 1.1) of TCP so that it is able to serve the needs of the delay-insensitive applications. This approach is named as Network Friendly TCP (NF-TCP). In the rest of the chapter, the thesis discusses the motivation behind the approach and lists out the key contribution of the thesis.

1.1 Problem statement

Internet traffic is ever increasing and becoming more and more varied and complex. Video, live-streaming and gaming are some of the popular applications which are sensitive to packet drops, delay and jitter. On the other hand, download and play, peer-to-peer applications are tolerant to delay, lost packets and jitter but typically require a large amount of data to be transferred. For example, according to a study from Ipoque [6], P2P accounted for a range between 43% and 70% of the total traffic in large parts of Europe, Middle East, Africa and South America. But, peer-to-peer like applications do not have stringent completion deadlines. Web-traffic on the other hand is an example of an application that has a need to complete in a reasonable time before a user times out, but they transfer relatively smaller amounts of data. In essence, different applications have varying requirements which are summarized in Table 1.1. Figure 1.1 illustrates the various application flows that might be traversing an access router. The access router could be either in a single household or at the Internet Service provider. All of these applications use TCP.

A legitimate question that arises in the design of a network friendly protocol is whether the Transport Layer is the right place for a mechanism that incorporates the policy encouraging delay-insensitive applications to be deferential to other applications in their usage of congested network resources. The primary argument behind this is driven by considerations of time-scale. Placing the responsibility on the user to defer their downloads during congestion periods is difficult because of both the unpredictable nature of congestion (and hence there is no well-defined period of "uncongested network") and its short-term transient nature (relative to human user time scales). Another alternative is to place the responsibility on the application, which requires the application to be then congestion-aware. Despite this being possible, it is difficult to be able to respond to congestion (and the absence of) within small time-scales. Even upon adding several hundreds of milliseconds, a substantial congestion buildup would occur, thereby impacting other traffic not only from the same host but also from others sharing a bottleneck. Moreover, capacity left unused is "lost" if the application is unable to opportunistically exploit available bandwidth within a time scale of a few round-trip times (RTTs). Based on these arguments, delay-insensitive traffic needs to be submissive and it is most suitable when performed as a transport layer protocol.

Limitations of Standard TCP

TCP provides proportional fairness: TCP aims to provide proportional fairness, such that a “flow” is the unit which is allocated resources. This refers to the allocation of resources in a fair manner among all the existing flows, i.e., aiming to ensure similar experience and treatment for every flow. This results in all the flows contributing equally to queue build-up at a congested bottleneck link, thereby causing an increase in delay to all the applications. In Figure 1.1, one can observe that various applications contribute at a single bottleneck. Applications such as P2P can open multiple connections, enabling all the flows to receive a share of the bandwidth. This results in unfair benefit of potentially higher throughput for applications that open multiple connections. Unfortunately, TCP cannot overcome this deficiency.

Standard TCP is not flexible: Standard TCP was designed as a non flexible transport layer protocol and provides similar transport functionality to all applications. This implies that users and applications do not have control over the type of service they desire. Instead, users and applications have to find other means such as manual switch on/off or open multiple connections, to fulfil their needs. An application that uses standard TCP, even though it may wish to be network friendly and be submissive to other traffic when the network is congested, is unable to tune or defer its load to avoid using resources in the network even when there is excess demand.

Inefficient use of spare bandwidth: Inefficiency in utilizing available bandwidth is another disadvantage of standard TCP. As the Bandwidth Delay Product (BDP) increases, performance decreases. This problem is particularly of concern in the case of network friendly flows due to their nature of being submissive during congestion periods.

Effect of ISP charging models

Charging models such as flat-rate charging increase the strain on the networks, since a certain amount of heavy users use up most of the resources. Coupled with the use of TCP, these heavy users try to compete for a fair share of the capacity. Applications that use UDP increase congestion due to the inelastic nature of such flows. One obvious solution to the shortage is that the Internet Service Providers (ISPs) increase their capacities to accommodate the extra traffic. Murat Yuksel et al. argue in [7] that the required extra capacity to service a small increase in traffic is quite significant. Moreover, users are not willing to pay for capacities that would benefit only a small number of players. In light of this, ISPs prefer to make efficient use of their capacity before attempting to increase it. Figure 1.2 shows the actual distribution of traffic in the current network whereas, Figure 1.3 depicts the distribution of traffic that ISPs would prefer.

ISPs and heavy users

ISPs generally try to circumvent the interruption to efficient use of capacity by deploying various charging mechanisms such as volume counting/charging, application based rate limiting, deep packet inspection, blocking/throttling certain applications like P2P, and bottleneck policing. All these underline the fact that according to ISPs, heavy users are receive a disproportionate share of resources. But this raises the following question: Is a user using 5 Mbps in an empty bottleneck link with a bandwidth of 10 Mbps worse than a user using 1 Mbps of the same link during congestion? In the first case, there is no impact on other users whereas, in the second case, all users are affected. Neither the volume of data nor the rate, but the contribution to congestion is what really matters. People using the network during congested periods may be willing to pay a premium for it. Similarly, the ISPs may intend to provide cheaper service to flows that postpone their use of the network to a non congested period. Currently, once a network has been provisioned at a certain size, it does not cost the network operator any more even if the user sends more data without causing congestion [8].This is analogous to offering discounts for using public transport and flights during non peak periods.

Unfortunately, the current protocols are not well suited to use spare bandwidth efficiently, while minimizing the impact caused during congested periods. There is a need for a mechanism that can be used by delay-insensitive applications to seamlessly use spare bandwidth efficiently and be informed so as to enable them to defer usage of the network during congested periods.

1.2 Thesis Contributions

Increasing amount of Internet users along with a wide range of diverse Internet applications are currently motivating researchers to come up with adaptive and flexible Internet solutions. In this thesis, a network-friendly

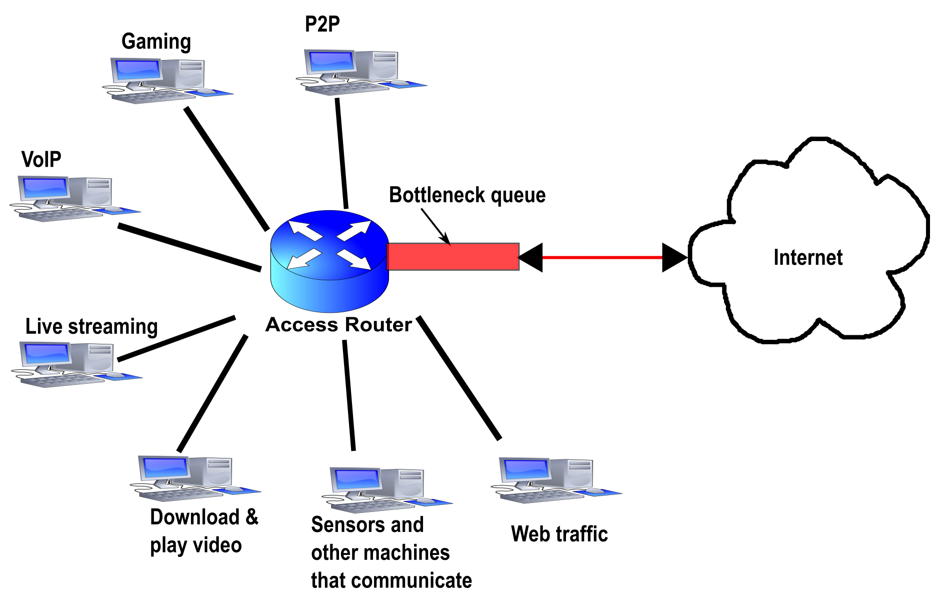


Figure 1.1: Examples of various application flows traversing a common access router.

congestion control protocol (NF-TCP) was developed. It differs from existing approaches in three major ways:

• NF-TCP is a network-friendly, submissive variant of TCP, depending on **network feedback**. It exploits Explicit Congestion Notification (ECN) [9] to reliably identify incipient congestion and have delay-insensitive applications aggressively defer their load from a congested network.

• An aggressive but informed exploitation of available bandwidth such that NF-TCP utilizes spare capacity when the network is uncongested. It exploits information provided by an available **bandwidth measurement component** that is carefully crafted to rapidly obtain the estimate. This is the first work that incorporates a separate bandwidth measurement mechanism into a congestion control framework.

• While NF-TCP exploits the available bandwidth estimation, it is designed to still be **robust** to errors in the estimation.

The key contributions to this thesis include:

Table 1.1: Application requirements

	Examples of High	Examples of low
Volume of data transfer	P2P, download & play	Web-traffic
Tolerance to jitter	Web-traffic	Video, gaming live streaming
Tolerance to delay	P2P	Web-traffic, gaming
Tolerance to packet loss	streaming, video	Web-traffic, gaming, sensors

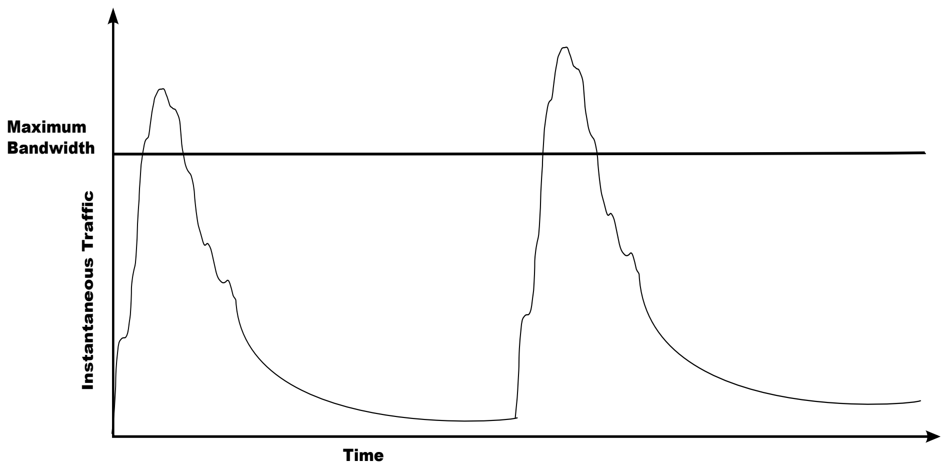


Figure 1.2: Current distribution of traffic in the ISP networks, where peaks resulting in congestion during certain periods are prominent

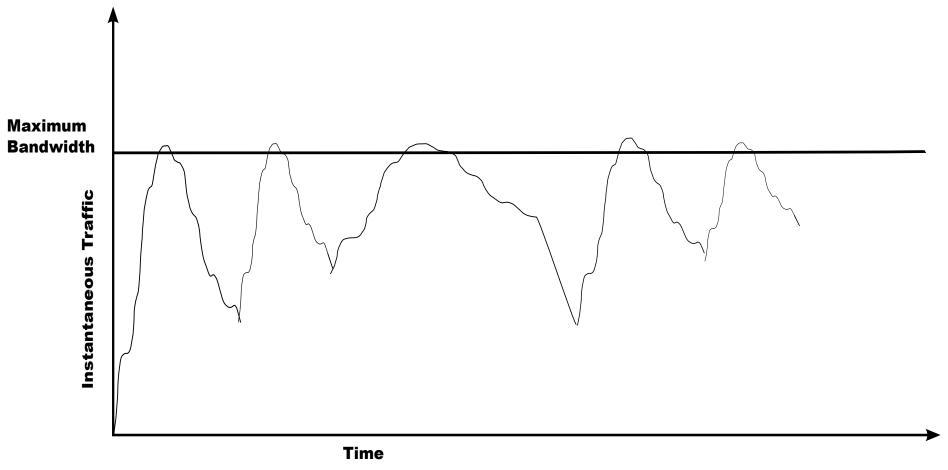


Figure 1.3: An ideal distribution of traffic in the ISP networks, where applications are able to uniformly distribute their usage of the network capacity thereby reducing congestion periods in the network

- The thesis promotes a modular and flexible congestion control framework that comprises of decoupled congestion control mechanisms including congestion detection, flow control and bandwidth estimation. This facilitates the understanding and design of new components of a congestion control protocol, while allowing the reuse of existing available mechanisms and protocol elements.

- The thesis proposes Network Friendly TCP (NF-TCP) as an enhancement to TCP to allow delay-insensitive applications to be reliably submissive to delay-sensitive applications during periods of congestion. This has been implemented on Linux and results from a testbed setup on the ns-2 simulator [10] have been provided.

- The thesis proposes a novel ECN enhanced bandwidth estimation mechanism (ProECN) to guide NF-TCP to aggressively utilize the bandwidth during non-congestion periods. This enables NF-TCP to avoid under-utilization of available bandwidth.

1.3 Thesis overview

Chapter 2 highlights and discusses the design goals of the NF-TCP approach. Moreover, it reviews the existing proposals and details their shortcomings, thereby reducing their chance to be a network friendly protocol. Chapter 3 outlines the congestion control framework and NF-TCP in described in detail in Chapter 4. In Chapter 5, evaluation results of a comparative study of NF-TCP, LEDBAT, RAPID, TCP-LP and standard TCP Reno is provided. Chapter 6 details upon the deployment scheme and tries to answer some of the frequently asked questions. Finally, a summary of the dissertation has been presented in Chapter 7.

Chapter 2

Design Goals and Related Work

In this chapter, the thesis explains the design goals of NF-TCP and explore the state of the art approaches that could be used by delay-insensitive applications.

2.1 Design Goals of a Network Friendly Congestion Control Protocol

The main goal of this work is to develop a network friendly protocol that addresses the challenges described above. The solution is built based on the following requirements:

Requirement-I: Be submissive to standard TCP when encountering network congestion

This is to ensure that packets of delay-insensitive applications do not occupy the buffers, which can impact existing or newly arriving TCP flows. Buffer occupancy could result in an increase in latency or drop rate for delay-sensitive applications. This also enables standard TCP flows to utilize the available capacity.

During congestion periods where the queue is building up, one gets the following condition for NF-TCP using the rate deterministic model ($T_N \propto \frac{a}{\sqrt{p}}$) described in [11]:

$$\frac{a_N}{\sqrt{p_N}} \to 0 \tag{2.1}$$

where T_N is the throughput of an NF-TCP flow, p_N is the loss rate of the NF-TCP flow and a_N is the rate of increase of the NF-TCP flow. Therefore, to meet the conditions of the equation, either NF-TCP's increase factor (a_N) should be close to zero or the loss rate of NF-TCP (p_N) should be very high.

Requirement-II: Ability to detect network congestion early

Early detection of congestion provides NF-TCP ample time to become submissive and yield its share of the capacity thereby satisfying Requirement-I.

Requirement-III: Ability to saturate available bandwidth as fast as possible in the absence of other TCP flows

An NF-TCP flow must be capable of aggressively capturing available bandwidth during non-congestion periods without having a negative impact on co-existing TCP flows. Moreover, in the presence of other NF-TCP flows, the bandwidth should be equally shared among all flows.

Requirement-IV: Ability to operate in very short timescales

As mentioned in Requirement-I, II and III, an NF-TCP flow must be capable of aggressively capturing available bandwidth during non-congestion periods and being submissive during congestion periods. It is therefore essential for it to operate in as short a timescale as possible. The choice of a transport layer solution ensures that NF-TCP is able to operate in timescales of a few RTTs.

Comparison of a NF-TCP flow to a standard TCP flow

It is undesirable for delay-insensitive traffic to contribute to queuing because of the elastic nature of its demand and the inherent ability of such applications to defer until the network is relieved of congestion. Figure 2.2 shows the friendly nature of standard TCP as compared to what is expected of a network friendly TCP shown in Figure 2.1. One can observe in Figure 2.1 that the network friendly flow is able to be submissive in the presence of standard TCP flow during congestion instead of competing for an equal share. Figure 2.1 illustrates that standard TCP is able to utilize the full bandwidth whereas in Figure 2.2, standard TCP is able to utilize only half the capacity.

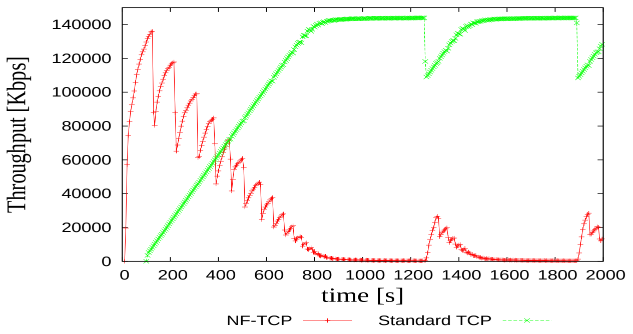


Figure 2.1: Required behaviour of a network friendly flow

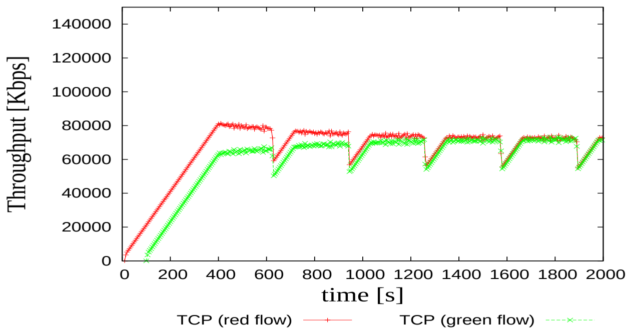


Figure 2.2: TCP Fairness

2.2 State of the Art in Network Friendly Approaches

The IETF Application-Layer Transport Optimization (ALTO) [12] protocol relies on the support of dedicated servers to assist in the selection of peers, to avoid the overloading of any specific part of the network. Application layer solutions such as ALTO [12] that help in peer selection could in fact compliment transport layer solutions. Recently, to relax the dependence on dedicated servers and to react to instantaneous congestion levels in the networks, the IETF LEDBAT working group has been formed to develop a network friendly protocol [2], which is primarily an end-host delay-based protocol. Other delay-based network friendly mechanisms include TCP-LP [13] and RAPID [14].

In this section, the work briefly describes standard TCP and other candidate approaches for a network friendly protocol. The properties of each of the approach are then compared against the requirements of a network friendly approach listed out in Section 2.1.

2.2.1 Standard TCP

The Transmission Control Protocol (TCP) [5] is the most widely used transport layer protocol and the thesis uses the term standard TCP to denote the same. TCP establishes a connection between sender and receiver. Each segment, which is sent over the network is acknowledged by the receiver to make the data transfer reliable. The time taken from sending a packet to receiving an acknowledgement (ACK) for this packet is called RTT (Round Trip Time). Among other abilities TCP provides Congestion Control [15] by using four algorithms: *slow start*, *congestion avoidance*, *fast retransmit* and *fast recovery* which are defined in [5]. These algorithms make extensive use of two variables: *congestion window* (`cwnd`), which limits the amount of data a sender can inject into the network before receiving an acknowledgement and the receiver's *advertised window* (`rwnd`), which limits the amount of outstanding data to a value which the TCP receiver can handle. The congestion window is limited by the *advertised window* so as to not overwhelm the receiver. To determine whether slow start or congestion avoidance should be used, another state variable, the *slow start threshold* (`ssthresh`) is used. Initially `ssthresh` is set arbitrarily high and is then reduced in response to congestion as described later on in this chapter.

TCP Slow Start

During the slow start phase, for every successful transmission and acknowledgement of a TCP segment, the congestion window is doubled resulting in an exponential growth. This exponential growth is continued till the congestion window reaches the SS-threshold value or till a packet loss is observed (See Figure 2.3). When a TCP sender is starting to transmit data, the available bandwidth of the network is unknown. Therefore, it slowly needs to probe for the capacity in order to refrain from introducing congestion in the network. It does so by using the slow start algorithm. Slow start is used when `cwnd` < `ssthresh`, while congestion avoidance is used otherwise.

At the beginning of a connection TCP starts with a `cwnd` of one segment. This means that it is only able to send one segment and has to wait for the acknowledgement from the receiver, before sending a second segment.

During slow start the sender increases the `cwnd` by at most one segment size per acknowledgement received, leading to effectively doubling the congestion window each RTT. The slow start phase is terminated when the `cwnd` exceeds `ssthresh` or when congestion is observed. The setting of the `cwnd` on a received acknowledgement is then given by:

$$\text{ACK} : cwnd \leftarrow cwnd + min(N, SMSS) \tag{2.2}$$

where `SMSS` is the maximum segment size and `N` is the number of unacknowledged bytes acknowledged by the receiver.

TCP Congestion avoidance

TCP enters the congestion avoidance phase on exiting the slow start phase. In this phase, the `cwnd` is incremented roughly by one `SMSS` per RTT. One way to achieve this behavior is to count the number of acknowledged bytes for new data. When the number of bytes counted reaches `cwnd`, it is incremented by one `SMSS`. Another way to increase the congestion window on every received ACK is given by:

$$\text{ACK} : cwnd \leftarrow cwnd + SMSS * SMSS/cwnd \tag{2.3}$$

Both congestion avoidance formulas lead to an almost linear increase of the `cwnd`.

When TCP switches to congestion avoidance, the `ssthresh` variable is reset on detecting a segment loss event and given by:

$$\text{SEGMENT\_LOSS} : ssthresh \leftarrow max(FlightSize/2, 2 * SMSS) \tag{2.4}$$

where `FlightSize` is the amount of sent but not yet acknowledged data in the network. When there is no acknowledgement received for a long time and the connection times out, the `cwnd` is reset to one `SMSS`.

In the Congestion Avoidance algorithm a retransmission timer expiring or the reception of duplicate ACKs can implicitly signal the sender that a network congestion situation is occurring. The sender immediately sets its transmission window to one half of the current window size (the minimum of the congestion window and the receivers advertised window size), but to at least two segments. If congestion was indicated by a timeout, the congestion window is reset to one segment, which automatically puts the sender into Slow Start mode. If congestion was indicated by duplicate ACKs, the Fast Retransmit and Fast Recovery algorithms are invoked (see below). As data is received during Congestion Avoidance, the congestion window is increased. However, Slow Start is only used up to the halfway point where congestion originally occurred. This halfway point was recorded earlier as the new transmission window. After this halfway point, the congestion window is increased by one segment for all segments in the transmission window that are acknowledged. This mechanism will force the sender to more slowly grow its transmission rate, as it will approach the point where congestion had previously been detected (See Figure 2.3).

TCP Fast Retransmit and Fast Recovery

If three or more duplicate ACKs are received, it is a strong indication that at least one segment has been lost. The sender does not even wait for a retransmission timer to expire before retransmitting the segment (as indicated by the position of the duplicate ACK in the byte stream). This process is called the Fast Retransmit algorithm and immediately following Fast Retransmit is the Fast Recovery algorithm.

Since the Fast Retransmit algorithm [16] is used when duplicate ACKs are being received, the TCP sender has implicit knowledge that there is data still flowing to the receiver. The reason is because duplicate ACKs can only be generated when a segment is received. This is a strong indication of the fact that serious network congestion may not exist and that the lost segment was a rare event. So instead of reducing the flow of data abruptly by going all the way into Slow Start, the sender only enters Congestion Avoidance mode. Rather than start at a window of one segment as in Slow Start mode, the sender resumes transmission with a larger window, incrementing as if in Congestion Avoidance mode. This allows for higher throughput under the condition of only moderate congestion.

Features of standard TCP against the requirements of a network friendly transport protocol listed in Section 2.1

Table 2.1 compares the features of the standard TCP protocol to that of the requirements of a network friendly protocol listed out in Section 2.1. As mentioned earlier, TCP is not designed to be a network friendly approach and is therefore not submissive. Moreover, TCP detects congestion via packet losses or an ECN marking to indicate the same. TCP is not equiped to use spare bandwidth aggressivley too.

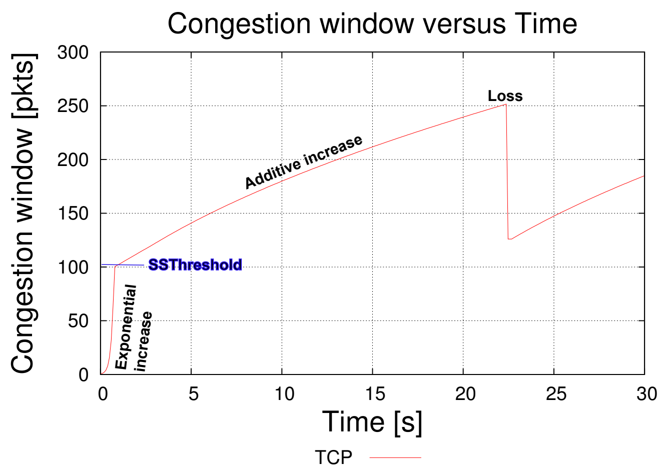


Figure 2.3: TCP congestion control overview.

Table 2.1: Comparison of Standard TCP to the requirements of a network friendly transport protocol listed in Section 2.1

	Yes/no	Detail
Submissive	No	
Detect Congestion earlier	No	
Saturate bandwidth as fast as possible	No	
Ability to operate in shorter timescales of a few RTTs	Yes	

2.2.2 Application-Layer Transport Optimization (ALTO)

A significant part of the Internet traffic today is generated by peer-to-peer (P2P) applications used for file sharing, real-time communications, and live media streaming. P2P applications transfer a large volume of data and in most cases download from multiple sources. In P2P systems, the files stored are normally redundant in nature. Therefore nodes will have to choose from a selection of peers. Currently, peers make this selection based on some adhoc values and do not base their decision on the underlying topology, cost to the Internet Service Provider (ISP) among others. Given the emergence of P2P and other delay-insensitive traffic, recent developments have attempted to provide means to support such applications. The IETF Application-Layer Transport Optimization (ALTO) [12][17][18] protocol relies on the support of

Table 2.2: Comparison of ALTO to the requirements of a network friendly transport protocol listed in Section 2.1

	Yes/no	Detail
Submissive	No	
Detect Congestion earlier	No	Longer timescale
Saturate bandwidth as fast as possible	No	Not in the required timescale
Ability to operate in shorter timescales of a few RTTs	No	Decisions are made at the beginning of a transfer

dedicated servers to assist in the selection of peers, to avoid the overloading of any specific part of the network. ALTO helps applications to perform better-than-random initial peer selection. ALTO services may take different approaches at balancing factors such as maximum bandwidth, minimum cross-domain traffic, lowest cost to the user. Application layer solutions such as ALTO [12] that help in peer selection could in fact compliment the NF-TCP.

Table 2.2 compares the features of the ALTO protocol to that of the requirements of a network friendly protocol listed out in Section 2.1. Since ALTO is an application layer solution, one can note that it is not able to operate in the timescales required. As mentioned earlier, ALTO can compliment NF-TCP in the initial phase of peer selection by providing a better-than-random peer selection.

2.2.3 TCP westwood

TCP Westwood [19] (or TCPW for short) differs from Reno in the fact that it adjusts the congestion window after a loss detection by setting it to the measured rate currently experienced by the connection, rather than using the conventional multiplicative decrease scheme (i.e., divide the current window by half). The control scheme of TCPW converges to “fair share” at steady state under uniform path conditions.

In TCP Westwood the sender continuously computes the connection Bandwidth Estimate (BWE) which is defined as the share of bottleneck bandwidth used by the connection. Thus, BWE is equal to the rate at which data is delivered to the TCP receiver. The estimate is based on the rate at which ACKs are received and on their payload. After a packet loss indication, (ie, reception of 3 duplicate ACKs, or timeout expiration), the sender resets the congestion window and the slow start threshold based on

Table 2.3: Comparison of TCP Westwood to the requirements of a network friendly transport protocol listed in Section 2.1

	Yes/no	Detail
Submissive	No	
Detect Congestion earlier	No	BWE does it to a certain extent
Saturate bandwidth as fast as possible	Yes	Uses BWE
Ability to operate in shorter timescales of a few RTTs	Yes	

BWE.

Table 2.3 compares the features of the TCPW protocol to that of the requirements of a network friendly protocol listed out in Section 2.1. Since TCPW was not designed to be a network friendly approach, it is not able to be submissive. Moreover, on detecting congestion, it neither gives up its share of the bandwidth nor reduce the congestion window by half. It reduces the sending rate to a rate based on the BWE.

2.2.4 TCP Vegas

TCP Vegas [20][21] was one of the first TCP protocols to identify congestion by means other than a packet loss or duplicate ACKs. In the Congestion Avoidance phase, TCP Vegas calculates the expected throughput as congestion-window-size / BaseRTT, where congestion-window-size is the current congestion window and BaseRTT is the minimum of all measured RTTs. If the actual measured throughput is smaller than the expected throughput (minus a threshold beta), this is taken as a sign of congestion, causing the protocol to linearly decrease its rate.

Table 2.4 compares the features of the TCP Vegas protocol to that of the requirements of a network friendly protocol listed out in Section 2.1. Since TCP Vegas was not designed to be a network friendly approach, it is not designed to be completely submissive. It is designed to have a smaller sending rate than TCP when they share a bottleneck bandwidth.

2.2.5 TCP Nice

TCP Nice [22] is similar to TCP Vegas and improves upon it. TCP Vegas, on detecting congestion decreases its rate linearly, whereas TCP Nice reduces its congestion window by half, similar to that of TCP. To avoid being too

Table 2.4: Comparison of TCP Vegas to the requirements of a network friendly transport protocol listed in Section 2.1

	Yes/no	Detail
Submissive	No	
Detect Congestion earlier	Yes	Not reliable
Saturate bandwidth as fast as possible	No	
Ability to operate in shorter timescales of a few RTTs	Yes	

Table 2.5: Comparison of TCP Nice to the requirements of a network friendly transport protocol listed in Section 2.1

	Yes/no	Detail
Submissive	Partially	
Detect Congestion earlier	Yes	Not reliable
Saturate bandwidth as fast as possible	No	
Ability to operate in shorter timescales of a few RTTs	Yes	

conservative, TCP Nice performs such a decrease only on noticing a number of delay-based incipient congestion that is above a certain threshold per RTT. Else it uses TCP Vegas linear decrease and reverts to standard TCP's decrease on experiencing a packet loss.

Table 2.5 compares the features of the TCP Nice protocol to that of the requirements of a network friendly protocol listed out in Section 2.1. TCP Nice unlike TCP Vegas is partially submissive, especially on noticing enough number of delay-based incipient congestion signals.

2.2.6 TCP-LP: A distributed algorithm for low priority data transfer

The goal of TCP Low Priority (TCP-LP [13]) is to provide a low priority transport protocol in the presence of TCP. It provides a low priority service instead of the best effort service provided by standard TCP. It achieves this by measuring the one-way delay.

Table 2.6: Comparison of TCP-LP to the requirements of a network friendly transport protocol listed in Section 2.1

	Yes/no	Detail
Submissive	Yes	
Detect Congestion earlier	Yes	Not reliable
Saturate bandwidth as fast as possible	No	
Ability to operate in shorter timescales of a few RTTs	Yes	

One-way delay: It is defined as the transmission delay of a packet that is sent from a source to the destination. This is different from an round trip time (RTT), and is a better indicator of congestion in the forward path. An RTT might include transmission delay caused by the ACK having to traverse a congested link. The issue with measuring one-way delay is that the source and the destination clocks might have a slight skew. In order to negate this effect, it is important to use the measured one-way delay as a reference delay.

TCP-LP source attempts to measure one-way delay to identify the onset of congestion and therefore does not need any network support. It aims to use spare bandwidth whenever available and therefore tries to be transparent to both TCP and UDP traffic. LEDBAT [2][23], which is currently being pursued by the IETF, is similar to TCP-LP, and is explained more elaborately, in the next section.

Table 2.6 compares the features of the TCP-LP protocol to that of the requirements of a network friendly protocol listed out in Section 2.1. TCP-LP is submissive and detects congestion early by using one-way delay measurements. The disadvantages of a delay-based congestion detection mechanism are elaborated later in Section 2.3.

2.2.7 LEDBAT: Low Extra Delay Background Transport

The IETF LEDBAT working group is developing a Low Extra Delay Background Transport (LEDBAT) [2] layer solution as an alternative experimental congestion control algorithm that could be used by delay-insensitive applications. Their goal is to standardize a network congestion control algorithm, that is able to saturate the bottleneck, while maintaining low delay and not competing to standard TCP traffic. This allows applications that send large amounts of data, particularly upstream on home connections, such as peer-to-peer application, to operate without destroying the user experience in interactive applications. LEDBAT enables a networking application to minimize the extra delay it induces in the bottleneck while saturating the bottleneck. It thus implements an end-to-end version of scavenger service. This mechanism not only allows to keep delay across a bottleneck low, but also yields quickly in the presence of competing traffic with loss-based congestion control. Beyond its utility for P2P, LEDBAT enables and rather makes it easier for other advanced networking applications to get out of the way of interactive applications.

The goal of the LEDBAT algorithm is to minimize the queuing delay. However it cannot be zero, because that would mean that the bottleneck link is not fully utilized. The goal is to keep it as close to a target value and not to have zero queuing delay. The LEDBAT algorithm measures the one way delay between sender and receiver and keeps the minimum as a base delay. The queuing delay then is set as the difference between the measured delay and the base delay. If the queuing delay is higher than the target value, LEDBAT reduces and if it is lower, it increases its *congestion_window*. The base delay is updated from time to time to be able to react to route changes. Listing 2.1 shows the pseudocode of the linear controller that is used by LEDBAT to set the congestion window on each acknowledgement.

```
on acknowledgement:
  delay = acknowledgement.delay
  update_base_delay(delay)
  update_current_delay(delay)
  queuing_delay = current_delay() - base_delay()
  off_target = TARGET - queuing_delay + random_input()
  cwnd += GAIN * off_target / cwnd
  # flight_size() is the amount of currently not acked data.
  max_allowed_cwnd = ALLOWED_INCREASE + TETHER*flight_size()
  cwnd = min(cwnd, max_allowed_cwnd)
```

Listing 2.1: Linear controller of LEDBAT [2]

In Listing 2.1, $TARGET$ is the target for the queuing delay and $current\_delay$ is the delay just measured. Because LEDBAT aims to be friendly to TCP, the $GAIN$ parameter is set so that the maximum ramp up speed is not faster than for TCP in congestion avoidance. The pseudocode presented is a simplification and ignores noise filtering and base expiration. The more precise pseudocode that takes these factors into account is shown in Listing 2.2 and must be followed.

In Listing 2.2, $NOISE\_FILTER$ maintains a list of the current delays

```
on initialization:
    set NOISE_FILTER delays used by current_delay() to +infinity
    set BASE_HISTORY delays used by base_delay() to +infinity
    last_rollover = -infinity # More than a minute in the past.
```

Listing 2.2: Precise linear controller of LEDBAT [2]

measured and *BASE_HISTORY* maintains a list of the measured base delays.

LEDBAT maintains a history of the current delays observed, with the older values being flushed out as shown in Listing 2.3.

```
    update_current_delay(delay)
      # Maintain a list of NOISE_FILTER last delays observed.
      forget the earliest of NOISE_FILTER current_delays
      add delay to the end of current_delays
```

Listing 2.3: List of current delays observed in LEDBAT [2]

To account for changes in route, the *Base_delay* is reset after a certain time period (currently every 25ms) as shown in Listing 2.4.

```
    update_base_delay(delay)
      # Maintain BASE_HISTORY min delays. Each represents a minute
      if round_to_minute(now) != round_to_minute(last_rollover)
        last_rollover = now
        forget the earliest of base delays
        add delay to the end of base_delays
      else
        last of base_delays = min(last of base_delays, delay)

    base_delay()
      min(the BASE_HISTORY min delays stored by update_base_delay)
```

Listing 2.4: Base_Delay resetting in LEDBAT [2]

Table 2.7 lists the variables used by the LEDBAT algorithm and their recommended values along with an advocated range. LEDBAT proposes the use of a consistent *TARGET* value. If flows using different *TARGET* values share a bottleneck, flows with higher values will tend to get a disproportionately large share of the bottleneck. This is the reason why *TARGET* is the

Table 2.7: Recommended LEDBAT parameters [2]

	Condition	Value
TARGET	MUST	100ms
GAIN	SHOULD	same or upto 2 times that of max ramp up of TCP
BASE_HISTORY	SHOULD	10 (greater than 2 and less than 20)
NOISE_FILTER	SHOULD	1 (atleast 1 and less than cwnd/2)
ALLOWED_INCREASE	SHOULD	atleast 1 packet and not more than 3 packets
TETHER	SHOULD	1.5 (must be greater than 1)
RANDOMNESS_AMOUNT	SHOULD	0 (between 0 and 0.1, inclusive)

Table 2.8: Comparison of LEDBAT to the requirements of a network friendly transport protocol listed in Section 2.1

	Yes/no	Detail
Submissive	Yes	
Detect Congestion earlier	Yes	Not reliable
Saturate bandwidth as fast as possible	No	
Ability to operate in shorter timescales of a few RTTs	Yes	

only parameter that is specified with no flexibility in the implementation. The *RANDOMNESS_AMOUNT* is used to introduce a random addition to the *TARGET* to avoid synchronization and is currently set to zero based on experience.

Table 2.8 compares the features of the LEDBAT protocol to that of the requirements of a network friendly protocol listed out in Section 2.1. LEDBAT, similar to TCP-LP, is submissive and detects congestion early by using one-way delay measurements. The disadvantages of a delay-based congestion detection mechanism are elaborated later in Section 2.3.

2.2.8 RAPID: Shrinking the Congestion-control Timescale

RAPID [14], unlike other congestion control protocols is a multi-rate based protocol based on the principles of pathChirp. RAPID sends packets with varying intergap and uses this intergap to identify available bandwidth. In the next subsection, pathChirp is explained in detail followed by a description of RAPID.

pathChirp: A tool to estimate available bandwidth

One of the main problems when talking about congestion control is to estimate the available bandwidth on a network path. PathChirp [3] uses a probe mechanism to solve this problem. It relies on the concept of self-induced congestion and introduces a chirp probing train, that consists of exponentially spaced packets to measure bandwidth. This section uses the terminologies and formulas, that are introduced in the pathChirp [3]. PathChirp estimates the available bandwidth along a network path by launching a number of *packet chirp trains* from sender to receiver. The structure of a typical chirp train is shown in 2.4.

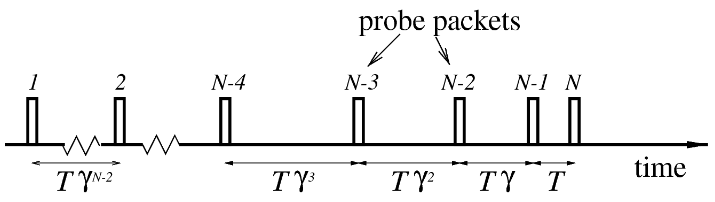


Figure 2.4: Chirp probe train [3]

Each chirp consists of N packets of size P, which are exponentially spaced. The inter packet spacing time between packet k and packet $k+1$ is denoted as Δ_k and its queuing delay as q_k. The *spreadfactor* γ is the ratio between two neighbouring gaps. The instantaneous chirp rate, with which packet k is sent, is then given as $R_k = P/\Delta_k$. $B[a, b]$ is the available bandwidth in the time interval from a to b and t_k the sending time of packet k. When the receiver has received the full chirp train, it conducts an analysis of the inter packet gaps. In a simplified scenario, if $q_k > q_{k-1}$, packet k has induced some additional delay, because $R_k > B[t_1, t_N]$. So a simple estimation for the bandwidth is $B[t_1, t_N] = R_{k*}$, where $k*$ is the last packet before the delay begins to increase. After calculating the available bandwidth, the receiver encodes the estimate in a packet and sends it back to the sender. Due to bursty traffic in a real network, the queuing delay normally does not increase monotonically. In Figure 2.5, a typical plot for the queuing delay is shown.

The estimation of available bandwidth during the total sending time of a whole chirp is based on an estimate E_k of the *per-packet available bandwidth* $B[t_k, t_{k+1}]$. E_k is estimated depending on if a packet is part of an excursion or not. Usually the last excursion does not terminate. If there is such an excursion, the last packet before the excursion starts is denoted as l. The

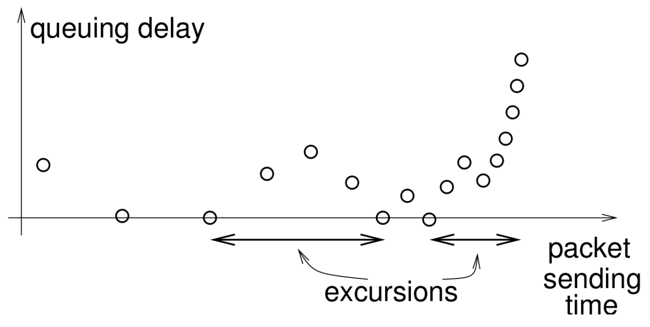


Figure 2.5: A typical chirp queuing delay signature [3]

pathChirp paper [3] differentiates between three cases:

1. If packet k belongs to an excursion that terminates and $q_k \leq q_{k+1}$, then set
$$E_k = R_k \tag{2.5}$$
2. If packet k belongs to an excursion that does not terminate then set
$$E_k = R_l, \forall k > l \tag{2.6}$$
3. For all packets k that do not fit into the upper cases, set
$$E_k = R_l \tag{2.7}$$

The pathChirp algorithm uses two variables to determine if there is an excursion and when the excursion ends. The *decrease factor* F is used to determine the end of an excursion. Let packet i be the packet where $q_i < q_{i+1}$. The end of an excursion, packet j is then the first packet where

$$q_j - q_i = \frac{max_{i \leq k \leq j}[q_k - q_i]}{F}. \tag{2.8}$$

The packets between packet i and packet j are considered to form an excursion, when $j - i > L$, whereby L is called the *busy period threshold*. The *per-chirp available bandwidth*, D is then calculated as

$$D = \frac{\sum_{k=1}^{N-1} E_k \Delta_k}{\sum_{k=1}^{N-1} \Delta_k}. \tag{2.9}$$

Finally pathChirp averages the instantaneous estimate of m chirps for the final bandwidth estimate.

Table 2.9: Comparison of RAPID to the requirements of a network friendly transport protocol listed in Section 2.1

	Yes/no	Detail
Submissive	Yes	Not reliable, Not designed to be network friendly
Detect Congestion earlier	Yes	Not reliable
Saturate bandwidth as fast as possible	Yes	
Ability to operate in shorter timescales of a few RTTs	Yes	

RAPID design

RAPID's design is motivated by the primary goal of shrinking the congestion-control timescale. To achieve this goal it does not use a congestion window anymore, but fully relies on an orchestrated rate based packet sending. To estimate the available bandwidth, it makes the same approach as pathChirp. All packets are sent in chirp trains of N packets with decreasing inter-packet gaps, the receiver observes these gaps, computes the available bandwidth by looking for increasing trends and encodes its estimation into the acknowledgement sent to the sender. When the sender receives an estimation, it updates its average sending rate to the received estimation. With the next chirp of packets it then probes for some rates below and some above the received estimation, which makes it possible to react to bandwidth changes. To solve the problem of how to set the average sending rate at connection startup, it limits the traffic at the beginning to be only as aggressive as TCP during slow start. It does so by limiting the number of packets per chirp to $N = 2$ initially, doubling it once per RTT, going up to a maximum value of $N = 16$. During this slow start phase, it uses a spreadfactor between the packets of $\gamma = 2$, making it possible to probe for a large range of different bandwidth values. After reaching $N = 16$, it switches over to its steady-state mode where it uses $N = 30$ and $\gamma = 1.07$.

Table 2.9 compares the features of the RAPID protocol to that of the requirements of a network friendly protocol listed out in Section 2.1. RAPID is not designed to be completely network friendly and is therefore submissive only under certain scenarios as shown in Chapter 5. It depends on a delay based bandwidth estimation scheme and is therefore not totally reliable. The disadvantages of a delay-based congestion detection mechanism are elaborated later in Section 2.3.

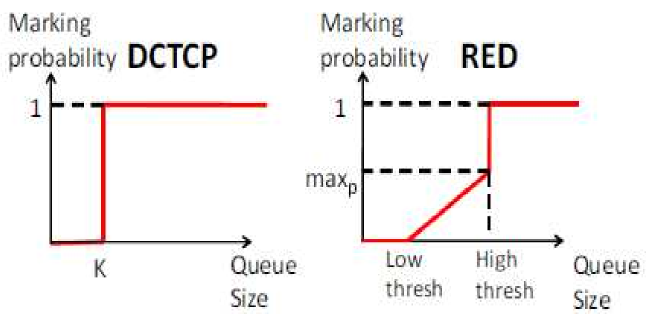


Figure 2.6: A router queue performing DCTCP marking [4]

2.2.9 Data Center TCP (DCTCP)

Data Center TCP (DCTCP [4]) is a TCP protocol designed for data centers. The advantage of this mechanism is that it is easily deployable and highly suited for a data-center environment.

The main goal of DCTCP is to achieve high burst tolerance, low latency, and high throughput, with commodity shallow buffered switches. DCTCP is therefore designed to operate with very small queue occupancies, while ensuring that the throughput is not compromised. DCTCP achieves these goals with the help of a modified router and by reacting to congestion in proportion to the extent of congestion. A DCTCP aware router performs instantaneous congestion marking when the queue size exceeds a certain threshold K as shown in Figure 2.6. One can observe that this marking scheme is quite different from the RED [24] marking behaviour. The Congestion Experienced (CE) codepoint of packets are set to indicate that the buffer occupancy has exceeded the threshold K. The DCTCP source takes into account the number of marked packets and proportionally adjusts its decrease factor. This results in a large decrease factor when it receives a large number of CE marked packets and viceversa.

Table 2.10 compares the features of the DCTCP protocol to that of the requirements of a network friendly protocol listed out in Section 2.1. DCTCP is also an ECN based approach that is able to detect congestion depending on the threshold value K. It is neither designed to be submissive nor aggressive in the usage of spare bandwidth. Moreover, the use of an instantaneous marking mechanism makes it inefficient on a large scale such as the Internet, since it would result in reacting to the highly dynamic traffic flow.

Table 2.10: Comparison of DCTCP to the requirements of a network friendly transport protocol listed in Section 2.1

	Yes/no	Detail
Submissive	No	Not designed to be network friendly
Detect Congestion earlier	Yes	
Saturate bandwidth as fast as possible	No	
Ability to operate in shorter timescales of a few RTTs	Yes	

2.3 The need for a more reliable network friendly congestion control protocol

Problems of delay based approaches

A limitation of these delay-based approaches [25–27] is that they require high precision packet time-stamps and accurate delay measurements in the implementation to identify the onset of congestion. Delay measurements are susceptible to noise in low latency networks and also in the presence of dynamic background traffic [28]. Additional difficulties include randomness and widely varying RTTs for competing traffic [29]. Evaluations conducted also highlight that LEDBAT is unfriendly to standard TCP and contributes to queue buildup in high bandwidth-delay product (BDP) networks.

Other delay-based approaches include Jain's Delay Based Congestion Control scheme [29], TCP Vegas [20] and partial delay-based approaches such as Compound TCP [30]. However, findings have shown that DCA-based approaches cannot function efficiently in high Bandwidth-Delay Product (BDP) networks. For example, Jain [29] mentioned the disadvantage of using a delay-based approach in networks with competing users and varying service times. Martin, Nilsson and Rhee [31] showed that DCA-based TCP is rarely (with only 7%-18% probability) able to detect queue build-up that precedes packet loss and react in time. Biaz and Vaidya [28] established that there is no strong correlation in DCA-based TCP approaches, between RTT and congestion window size, since the measured RTT is inaccurate and includes a high random component due to the bursty nature of the background traffic. La, Walrand and Anantharam argue about the problems of delay based approaches namely TCP Vegas in [32]. They mention that they do not work well in networks where rerouting of traffic causes a change in base delay and that it provides an unfair advantage to flows which constantly have higher RTT measurements. Evaluations conducted in this thesis also highlight that LEDBAT is unfriendly to standard TCP and contributes to queue buildup, in high bandwidth-delay product (BDP) networks.

Example of an issue with delay based approaches

Highlighting the disadvantages of a delay based approach, Mohammad Alizadeh et al. states in DCTCP [4],

> At very high data rates and with low-latency network fabrics, sensing the queue buildup in shallow-buffered switches can be extremely noisy. For example, a 10 packet backlog constitutes 120s of queuing delay at 1 Gbps, and only 12s at 10 Gbps. The accurate measurement of such small increases in queueing delay is a daunting task for todays servers.

Router support based mechanisms

VCP [33], MLCP [34] and BMCC [35] are examples of non delay-based approaches that use network-based feedback. VCP and MLCP are based on explicit congestion feedback from the network and require the router to maintain states of the link-load over a time period. They also need additional bits in the IP header, which may pose a challenge in itself. BMCC, on the other hand is another protocol based on explicit congestion feedback and requires only a single ECN bit. However, it requires knowledge about the number of flows in addition to maintaining states regarding the load. It would be difficult for the routers to identify the number of unique flows or maintain states to identify the start and end of a flow.

Problem with efficiently using bandwidth

Efficiently utilizing available bandwidth is another consideration driving towards existing solutions for high BDP environments. This problem is more prominent in the case of submissive flows due to their nature of being submissive during congestion periods. LEDBAT and TCP-LP are based on standard TCP and are not designed to aggressively utilize bandwidth during non-congestion periods and therefore achieve low overall throughput. To motivate users and application designers to use network friendly application, it is essential that the protocol is able to aggressively utilize spare bandwidth as fast as possible during non-congestion periods. High speed solutions such as High-speed TCP [36], FAST [37], Compound TCP [30], CUBIC [38], Quick-Start [39] and XCP [40] exist. However, part of their ability to opportunistically utilize the large bandwidths in these types of environments is due to their aggressive increase policies. Thus, they are not necessarily "submissive" or "friendly" to other delay-sensitive applications, and have the ability to cause a potential period of congestion. In fact, evidence of their "collateral damage" on existing TCP traffic has been documented (e.g., [41]). Moreover, each of these solutions have their respective advantages and disadvantages. Some require complex network support, whereas some are not flexible enough to easily adapt to meet the requirements of transporting delay-insensitive traffic.

Chapter 3

Decoupled Congestion Control Framework

To overcome the disadvantages of the candidate approaches mentioned in Chapter 2, this thesis proposes a Network Friendly TCP (NF-TCP). In this chapter, the modular congestion control framework that forms the foundation of the design of NF-TCP is discussed. This framework helps to assimilate the various components involved while designing NF-TCP. This architecture is briefly described in [42] and [43].

It can be observed that the traditional congestion control protocols comprise of tightly coupled mechanisms, and are designed to address a specific problem or scenario. They either lack the flexibility in meeting varying requirements or utilizing different types of available network support. For example, HighSpeed TCP [36] is likely unable to utilize the benefits of Quick-Start [39] like approaches even if the network supports it.

Motivated by this observation, this research proposes a modular congestion control framework that consists of pluggable components and utilizes it to guide the NF-TCP design. This work identifies congestion-detection, flow-control and bandwidth-estimation as the main components of a congestion control protocol, which can be loosely or strictly decoupled from each other under a modular framework as shown in Fig. 3.1. Using this framework to guide this NF-TCP design helps future extensions as mechanisms evolve. This work exploits the possibility of a measurement-based estimation to guide this aggressive increase (rather than an uninformed aggressive increase that has the potential to cause congestion). In the future sections, this thesis describes the functioning of the three modules of the NF-TCP protocol in detail.

3.1 Congestion-detection module

A network friendly congestion control avoidance protocol must be backed by an efficient congestion detection mechanism that is able to detect incipient

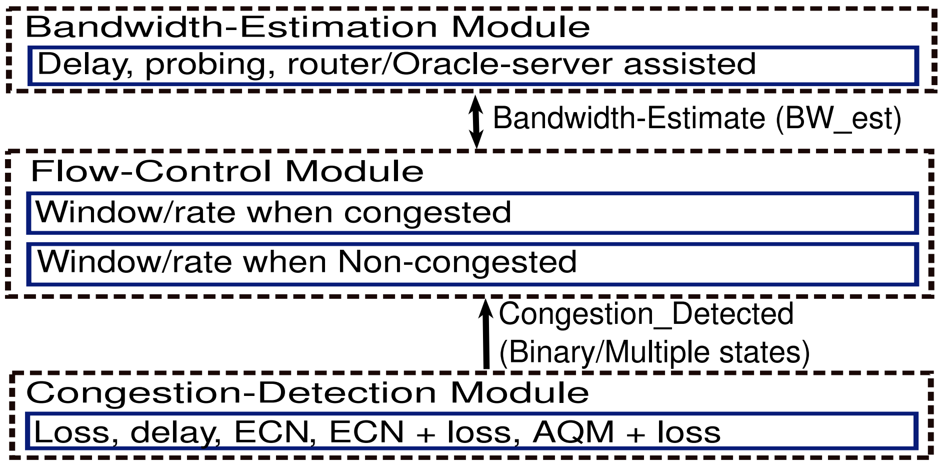

Bandwidth-Estimation Module
Delay, probing, router/Oracle-server assisted
Bandwidth-Estimate (BW_est)
Flow-Control Module
Window/rate when congested
Window/rate when Non-congested
Congestion_Detected
(Binary/Multiple states)
Congestion-Detection Module
Loss, delay, ECN, ECN + loss, AQM + loss

Figure 3.1: A guiding modular congestion control framework

congestion earlier than standard TCP, thereby enabling it to react earlier. In the case of a network friendly congestion control avoidance protocol, it is all the more important that the protocol is able to detect incipient congestion earlier than standard TCP (especially the loss-based congestion detection). Early detection of congestion assists a network friendly congestion control avoidance protocol in quickly yielding to standard TCP.

The aim is to ensure that the network friendly application packets do not contribute to queue build-up/congestion, which results in higher latencies for delay-sensitive and other traffic that may use TCP. The congestion detection scheme needs to ensure that the protocol is able to detect the onset of congestion as early as reasonable, while ensuring that it does not respond to truly short-term transients.

This module is responsible for detecting congestion by means of the existing mechanisms such as loss-based, ECN-based, delay-based or a combination of these approaches (and one could also differentiate among them based on assigned weights). It is also expected to provide a reliable indication via the CONGESTION_DETECTED variable to the flow-control module. Some example congestion detection mechanisms currently available are listed in the subsequent subsections.

3.1.1 Delay based congestion detection

Delay based schemes for detection of incipient congestion depend on the difference between the current one-way delay and the observed base delay.

The advantage of such a scheme is that it does not need network support and is easy to deploy. The current LEDBAT effort is based on such a mechanism to detect congestion. However, there is the potential that it may not be an accurate indicator of congestion, especially if the base delay was based on a congested scenario or if there were frequent route changes.

3.1.2 Explicit Congestion Marking (ECN) based congestion detection

A marking based scheme could depend on receiving explicit congestion marking from congested routers. Such a scheme would require the support of intermediate network nodes. This scheme could make use of the existing congestion marking schemes such as ECN marking [9], PCN marking [44] using the existing RED queue mechanisms. Additionally, it may be feasible to propose new marking behaviours.

3.1.3 Loss based congestion detection

The network friendly congestion control protocol may use packet loss as an indicator to detect congestion. But for this scheme to be effective, the network should be able to induce drops for the network friendly applications earlier than it does for standard TCP.

3.1.4 Combination of ECN, AQM, Loss based congestion detection

The module could also perform a congestion detection based on the combination of loss, ECN marking and/or other AQM techniques. This allows the congestion module to improve its accuracy and also perform a weighted congestion detection scheme wherein different mechanisms have different weights assigned.

$$\text{CONGESTION\_DETECTED} \leftarrow w1 * \mathrm{N}_{\mathrm{loss}} + w2 * \mathrm{N}_{\mathrm{ECN}} + w3 * \mathrm{N}_{\mathrm{AQM}} \quad (3.1)$$

where, $\mathrm{N}_{\mathrm{loss}}$,$\mathrm{N}_{\mathrm{ECN}}$ and $\mathrm{N}_{\mathrm{AQM}}$ represent number of loss, ECN markings and AQM markings experienced respectively.

3.2 Flow-control module

This module is responsible for placing load on the network. The flow-control module is in charge of the sending rate. This module is responsible for placing load on the network. Based on the feedback received from the bandwidth-estimation module and the congestion-detection module, it either increases or decreases its sending rate. The function of the flow control module can be broadly classified into two major parts:

• No congestion detected: When no congestion is detected, the NF-TCP flow control module is designed to use the feedback obtained from the bandwidth-estimation module to perform a more aggressive and guided increase than the standard linear increase. In case an estimate is not available, the NF-TCP resorts to a linear increase.

• Congestion detected: When the congestion-detection module signals the onset of congestion, the flow control module goes into submissive mode, wherein it performs a multiplicative decrease.

The sending rate can be based on the standard TCP like mechanism, wherein slow start is followed by a linear increase. To optimize the network throughput and be opportunistic in using available bandwidth, a network friendly congestion control mechanism may be designed to be aggressive during non-congestion periods. During this phase, the protocol can use the estimated bandwidth as a means to either perform multiplicative increase or jump directly to the estimated value. If the protocol is backed by an efficient congestion detection and reaction to congestion scheme, it can attempt to aggressively utilize the available bandwidth. This is driven by the consideration that not using available bandwidth means that it is most likely left un-utilized.

The decrease rate is an important feature when congestion is detected in a network friendly congestion avoidance protocol. It must be able to yield as quickly as possible to standard TCP on detection of incipient congestion. Standard TCP reduces its congestion window by half. But one may envisage schemes that have a more 'severe' multiplicative decrease or even a drastic reduction by decreasing the congestion window to one in an attempt to be extremely network friendly. The reduction rate could also depend on the accuracy of the congestion detection and available bandwidth estimation mechanisms and could therefore be lesser than 50%, e.g. 12.5%, similar to that proposed in [45]. Further investigation is necessary to arrive at a good decrease rate on the detection of incipient congestion, depending on the means used for detection as well as the bandwidth estimation method.

3.3 Bandwidth-estimation module

This module is responsible for estimating the available bandwidth. Traditionally, congestion control protocols detect congestion by placing an increasing load until congestion is detected. As a result, such protocols are required to be conservative. For example, standard TCP performs an increase of 1 packet per Round Trip Time (RTT). The proposed bandwidth-estimation module in NF-TCP should to be able to use mechanisms such as probing, estimates from a dedicated server, router assisted estimates (Quick-start for TCP [39]) or any other means.

Bandwidth estimation is the mechanism used by TCP to try to estimate the available bandwidth. Standard TCP schemes like RENO identify available bandwidth by placing a load on the network and exploring to determine the available bandwidth of the network. Standard TCP has been shown to be insufficient in high bandwidth delay product (BDP) networks, because of its conservative approach to increase the load on the network. Other schemes such as HighSpeed TCP [36], CUBIC TCP [38] and Quick-Start for TCP [39] try to capture the bandwidth more quickly.

An efficient bandwidth utilization mechanism backed by an efficient detection of incipient congestion will potentially improve the throughput of a network friendly TCP protocol during non-congestion periods. Since the network friendly congestion control protocol desires to be "submissive" in the presence of standard TCP during network congestion, an efficient bandwidth estimation scheme will enable it to be opportunistic in using available bandwidth. Thus it optimizes the throughput achieved as well as increases network utilization without causing congestion. The choice of a good starting behaviour could also depend on the network topology, the BDP of the network, and the support provided by the network.

A Bandwidth-estimation module would be expected to provide a reasonable estimate of the available bandwidth (BW_ESTIMATE) to the Flow-control module. The subsequent subsections introduce some of the prevalent bandwidth estimation schemes.

3.3.1 Slow start followed by linear increase

This is the standard TCP's implicit bandwidth estimation mechanism that works by placing load on the network until it causes congestion (queueing or loss at a bottleneck in the path). It starts with a slow start phase where it places an extra load of one packet per acknowledgement received and switches over to a linear increase phase wherein it places a load of an extra packet per RTT.

3.3.2 Probing packets based bandwidth estimation

The standard TCP bandwidth estimation mechanism has been shown to be insufficient in a high BDP network. An alternative bandwidth estimation method could be based on probing for available bandwidth.

For example, in a pathchirp-like probing mechanism, the end host sends a train of packets interspaced by specified intervals and tries to probe an estimate of available bandwidth between a lowrate and highrate. The estimate is deduced from changes in the delays experienced by the packets, and works in a somewhat similar conceptual approach as that used in delay-based methods for congestion control. Except here, the flow control reaction is not intimately bound to the delay experienced by the packets. The range depends on various factors such as the number of probe packets, the spread factor etc. The choice of lowrate and highrate can be varied to perform dynamic bandwidth estimation.

3.3.3 Network support based bandwidth estimation

Another means to overcome some of the limitations of slow start is to request the routers along the path to help with the bandwidth estimation.

For example, in a Quick-Start TCP [39] like mechanism, the end host would indicate its desired sending rate using a quick start option in the IP header of the TCP packet. Each router along the path will in turn, either approve the rate or reduce the rate. The receiver would in turn indicate the final rate to the sender. The router should be aware of the Quick-Start option to be able to help.

Chapter 4

Design

In this chapter, the design of NF-TCP [46] is illustrated in detail. It is guided by a modular congestion control framework, which separates estimation of available bandwidth and congestion detection from the reaction-to-incipient congestion. Since the main design goal is to be submissive to standard TCP during congestion periods, NF-TCP exploits the availability of Active Queue Management (AQM) routers that use a lower threshold to begin marking or dropping of packets belonging to NF-TCP flows. TCP packets are marked or dropped much later according to the standard marking schemes. This approach provides an earlier indication of the onset of congestion for NF-TCP, preventing it from occupying the queue. This way, NF-TCP could aggressively use spare bandwidth during non-congestion periods.

NF-TCP begins with a slow-start phase similar to standard TCP. Meanwhile, it seeks to take advantage of an associated available bandwidth estimation process that uses a probe-based approach to obtain a rough estimate of the available bandwidth. It uses this rough estimate as a safe guideline to increase its sending rate rapidly. It competes for equal share of the link capacity if and only if all the other existing flows are NF-TCP flows, since they are all influenced by the same control laws.

Figure 4.1 shows the various entities and how they interact. Router-1 is an NF-TCP aware core router whereas, Router-2 is either NF-TCP aware or not. The NF-TCP packets traverse a bottleneck link between Router-1 and Router-2, which is also shared with packets from standard TCP flows. When Router-1 is congested, it performs early marking for NF-TCP packets. The NF-TCP receiver forwards the feedback received to the sender to enable the sender to become submissive. Figure 4.2 illustrates the interworking of the congestion-detection module, the flow-control module and the bandwdith-estimation module, at the end host on the sender side.

4.1 Congestion-Detection Module

NF-TCP's network friendly congestion control is achieved by taking advantage of a congestion detection mechanism that detects incipient conges-

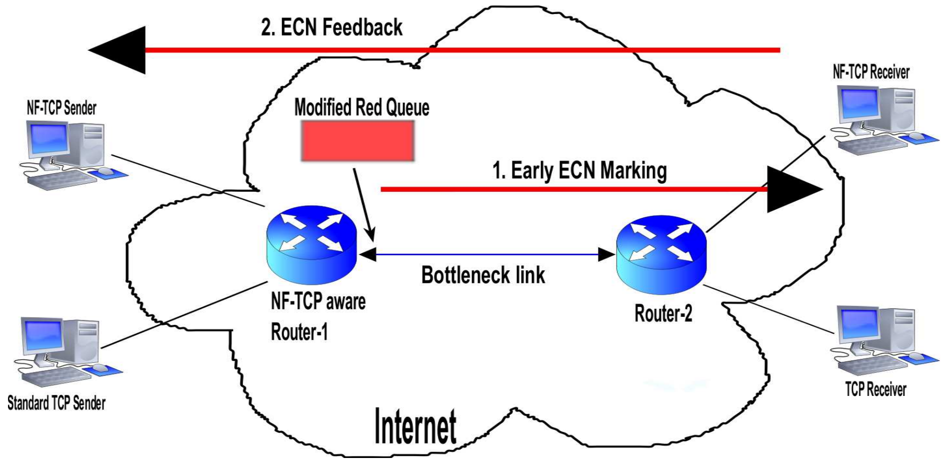


Figure 4.1: Various entities and their interaction

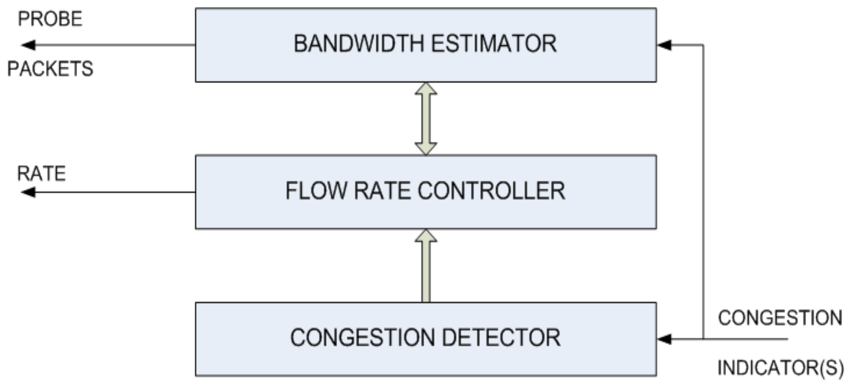


Figure 4.2: Flowchart showing the interaction between the various modules in the case of NF-TCP

tion earlier than standard TCP. NF-TCP exploits the availability of Active Queue Management (AQM) routers that are configured to use a lower threshold to begin marking or dropping of packets belonging to NF-TCP flows. The aim is to ensure that packets of network friendly applications do not contribute to queue build-up, which results in higher latencies for delay-sensitive traffic. The congestion-detection module is responsible for detecting congestion by means of any existing mechanism such as loss-based, ECN-based, delay-based or a combination of these approaches. This module is expected to provide a reliable indication via the CONGESTION_DETECTED variable to the flow-control module. The standard AQM mechanism is slightly modified to provide feedback for NF-TCP based on ECN, within time-scales of an RTT. ECN-unaware routers can drop NF-TCP packets earlier to indicate the onset of congestion. Next, how a modified Random Early Detection (RED) queue [24] is used for this purpose is shown.

Active Queue Management (AQM) is a queuing discipline in which packets are dropped or marked before the queue is full. Typically, they operate by maintaining one or more drop/mark probabilities, and probabilistically dropping or marking packets even when the queue is short. Random Early Detection (RED) [24] is a popular AQM technique wherein a RED enabled router calculates the average queue size using a Equally Weighted moving average (EWMA) and compares it to two threshold variables: the *minimum* and the *maximum* threshold. As shown in Figure 4.3, when the average is less than the minimum threshold no packets are marked, while all packets are marked when the average queue size exceeds the maximum threshold. When the average queue size is in between the minimum and the maximum threshold the packets are marked with a certain probability p_a, which is itself a function of the queue average. While TCP uses packet drops as an indication of congestion, the addition of Explicit Congestion Control (ECN) [9] was suggested to detect congestion earlier. ECN uses Active Queue Management (AQM) at the routers that allows the routers to not only drop packets, when its buffers are overflowing, but to react to congestion earlier. With AQM the routers can use a new Congestion Experienced (CE) codepoint in the packet header to indicate that the packet was buffered for some time. When a TCP receiver receives a packet with the CE codepoint set, it sets the ECN-Echo flag in its next acknowledgement informing the sender that the send data has experienced congestion. Through this the TCP sender is able to react to congestion earlier, than when it is only going by packet losses.

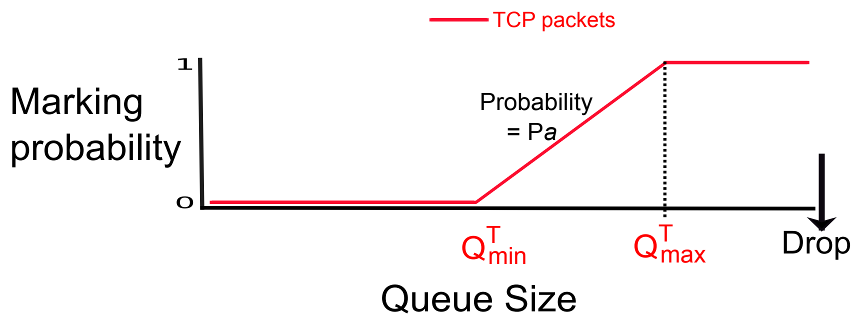


Figure 4.3: A router queue performing normal RED marking

Table 4.1: The used RED queue parameters for evaluation (Bandwidth = 100Mbps, RTT = 100ms)

	NF-TCP	Standard TCP
Min-threshold (packets)	12 (= 1% of queue size)	1133
Max-threshold (packets)	625 (= 50% of total queue size)	1250

4.1.1 Modified RED queue for NF-TCP

The modified RED queue consists of different parameters for NF-TCP compared to standard TCP. To detect the onset of congestion as early as possible while ensuring that one does not respond to truly short-term transients, the marking threshold values in the RED queue are set much lower than that for standard TCP. Fig. 4.4 illustrates the setting of the queue threshold values. The early marking and ECN for feedback to the source enables early reaction to the onset of congestion. The queue thresholds are based on the same mechanism as for a RED queue, except that the MinThreshold for NF-TCP flows (Q^N_{min}) is set much lower. The MaxThreshold for NF-TCP flows (Q^N_{max}) is set to about half of the buffer size. This ensures that once the "average" queue (based on an exponentially weighted moving average (EWMA)) begins to build up, NF-TCP packets are probabilistically marked (p). All NF-TCP packets are marked or dropped when the average queue size exceeds MaxThreshold, Q^N_{max}. Note that this mechanism uses a FIFO queue and is therefore different from the approach adopted by a priority queue [47] and other approaches such as [48–50]. This ensures that the delay-insensitive applications receive timely feedback and are therefore able

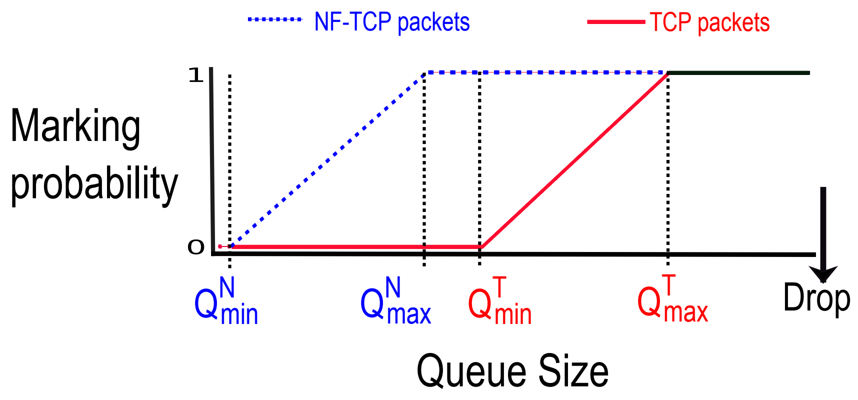


Figure 4.4: A router queue performing network friendly marking

to become submissive on their own.

The stochastic equations for the case of a standard TCP flow coexisting with an NF-TCP flow are now introduced. They share the same bottleneck link that involves a RED queue with modified parameters for an NF-TCP flow. Let B and q denote the size of the buffer and the queue respectively. The varied parameters result in the mean loss or marking rate of standard TCP (denoted by $N_{TCP}(q)$):

$$N_{TCP}(q) = \begin{cases} 0, & \text{if } \mathrm{q} \leqslant \mathrm{Q}^{\mathrm{T}}_{\mathrm{min}} \\ pmax_T, & \text{if } \mathrm{Q}^{\mathrm{T}}_{\mathrm{min}} < \mathrm{q} \leqslant \mathrm{Q}^{\mathrm{T}}_{\mathrm{max}} \\ 1, & \text{if } \mathrm{q} > \mathrm{Q}^{\mathrm{T}}_{\mathrm{max}}. \end{cases} \tag{4.1}$$

and the mean loss or marking rate of NF-TCP (denoted by $N_{NF-TCP}(q)$):

$$N_{NF-TCP}(q) = \begin{cases} 0, & \text{if } \mathrm{q} \leqslant \mathrm{Q}^{\mathrm{N}}_{\mathrm{min}} \\ pmax_N, & \text{if } \mathrm{Q}^{\mathrm{N}}_{\mathrm{min}} < \mathrm{q} \leqslant \mathrm{Q}^{\mathrm{N}}_{\mathrm{max}} \\ 1, & \text{if } \mathrm{q} > \mathrm{Q}^{\mathrm{N}}_{\mathrm{max}}. \end{cases} \tag{4.2}$$

where $pmax_T$ and $pmax_T$ are the marking probability for TCP and NF-TCP packets respectively. Figure 4.4 shows a graphical representation of how the respective threshold values will affect the marking behaviour. Figure 4.5 shows the effect of this differential marking behaviour on the network.

4.1.2 Analytical model of the modified queue parameters

Using the fluid model representation given in [11], the evolution of a single TCP congestion window during the congestion avoidance phase is described

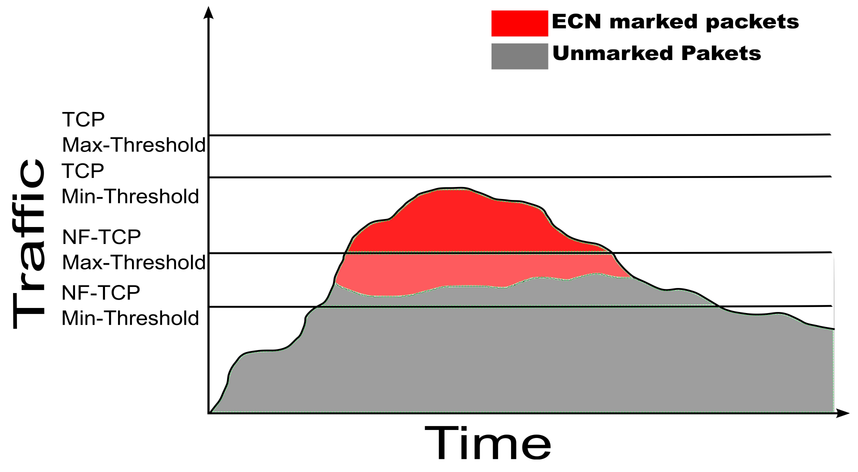


Figure 4.5: Instantaneous traffic distribution caused by NF-TCP on the ISP network

by the following hybrid differential equation:

$$dW(t) = \frac{a}{R}dt - \frac{W(t)}{b}N(dt). \tag{4.3}$$

This indicates that the congestion window increases between two loss events is linear with a slope of $\frac{a}{R}$, where a is the rate of increase of the congestion window per RTT (denoted by R) and W is the size of the congestion window. The loss events or marks result in reductions of the congestion window by a factor b. In this stochastic differential equation, $N(dt)$ represents the loss point process or ECN marking. In the case of NF-TCP, the window is increased by a units for every window acknowledgement where a is determined by the bandwidth-estimation module.

The modified RED queue is modelled using stochastic equations for the case of a standard TCP flow coexisting with an NF-TCP flow. They share the same bottleneck link that involves a RED queue with modified parameters for an NF-TCP flow. Let B and $Q(t)$ denote the maximum and instantaneous size of the buffer respectively. The stochastic equations consist of the following four operational phases (let X and Y denote the throughput of NF-TCP and standard TCP respectively).

Phase 1 or the free phase The buffer is nearly empty and therefore the aggressive growth of NF-TCP does not affect the growth of the TCP flow.

$$\{0 < Q(t) < Q^N_{min}\} \Rightarrow \begin{cases} dX(t) & = \frac{adt}{R} \\ dY(t) & = \frac{dt}{R} \end{cases} \tag{4.4}$$

Phase 2 or the submissive phase The queue size is greater than Q^N_{min} and NF-TCP packets get ECN markings with probability p, whereas TCP packets are not affected (and increase window by 1).

$$\{Q^N_{min} < Q(t) \leq Q^N_{max}\} \Rightarrow \begin{cases} dX(t) & = \frac{adt}{R} - \frac{W(t)*p}{b} \\ dY(t) & = \frac{dt}{R} \end{cases} \tag{4.5}$$

Equation 4.5 models the queue behavior when NF-TCP flows gets an indication of congestion earlier than TCP and start reducing their congestion window. This enables NF-TCP to be submissive to TCP.

Phase 3 or the starvation phase The queue size is greater than Q^N_{max} and all NF-TCP packets are marked.

$$\{Q^N_{max} < Q(t) \leq Q^T_{min}\} \Rightarrow \begin{cases} dX(t) & = - \frac{W(t)}{b} \\ dY(t) & = \frac{dt}{R} \end{cases} \tag{4.6}$$

Eq. 4.6 shows that the NF-TCP flows reduce their sending rate. By introducing loss or marking to NF-TCP packets earlier than that of TCP, it is ensured that NF-TCP is submissive to standard TCP. Thus the NF-TCP flow detects congestion at the very onset of queue buildup and backs off. The backing off results in yielding capacity to standard TCP flows. A non-ECN aware TCP would realize the onset of congestion only when $Q(t) > B$, but the NF-TCP flows would have already reduced their rate by then.

Phase 4 or the loss phase The queue size is greater than the Q^{TCP}_{min} and therefore both TCP and NF-TCP packets are marked or dropped.

$$\{Q^T_{min}(orB) < Q(t)\} \Rightarrow \begin{cases} dX(t) & = -\frac{W(t)}{b} \\ dY(t) & = -\frac{W(t)}{2}N(dt) \end{cases} \tag{4.7}$$

The evolution of the buffer occupancy in the intermediate router for $Q(t) > 0$ is given by

$$Q(t) = Q(0) + \int_0^t (X(u) + Y(u))\, du. \tag{4.8}$$

Eq.(4.4)-(4.8) imply that in the absence of a TCP flow, the instantaneous queue size is upper-bound close to Q^N_{min} and when NF-TCP and TCP coexist, the queue is upper-bounded by Q^T_{min}. The choice of a much lower value for Q^N_{max} than Q^T_{min} ensures that all packets of NF-TCP flows are being marked before TCP packets are marked or dropped. Therefore, the robustness of the approach is proportional to the difference between Q^N_{max} and Q^T_{min}. For example, for a bottleneck bandwidth of 100Mbps, with a buffer size equivalent to the BDP (=1250 packets), the queue parameters are set as depicted in Table 4.1. The results obtained (see Figure 5.3(a)) are in perfect agreement with the evolution of the queue size as predicted by Eq.(4.4)-(4.8). The RED parameters can be enhanced by using approaches such as [51–55].

Figure 4.6 shows a flow chart representation of an NF-TCP aware router. One can observe how the router handles various packets it receives, namely the probe packets, the NF-TCP packets and standard TCP packets.

4.2 Flow-Control Module

Based on the feedback received from the bandwidth-estimation module and the congestion-detection module, the flow control module adjusts the source's sending rate.

4.2.1 Congestion window decrease

On receiving a feedback of CONGESTION_DETECTED from the congestion-detection module (e.g., obtained from ECN), the NF-TCP flow control module reduces its sending window to yield to higher priority applications. Note that the NF-TCP flow does not differentiate between the existence or non-existence of standard TCP and therefore is designed to decrease its congestion window as follows:

$$\text{CONGESTION\_DETECTED} : w \leftarrow w - b * w. \tag{4.9}$$

From Eq.(4.4)-(4.7) we observe that even a low value for b would still result in NF-TCP being submissive. With a small value of b, NF-TCP flows will take longer to attain fairness among themselves. A high b on the other hand potentially results in low link utilization. Determined evaluations are based on a value of $b = 12.5\%$, similar to that proposed in [45]. It strikes a balance between being submissive to standard TCP, improving the link utilization and fairness in the presence of only NF-TCP flows. A 50%

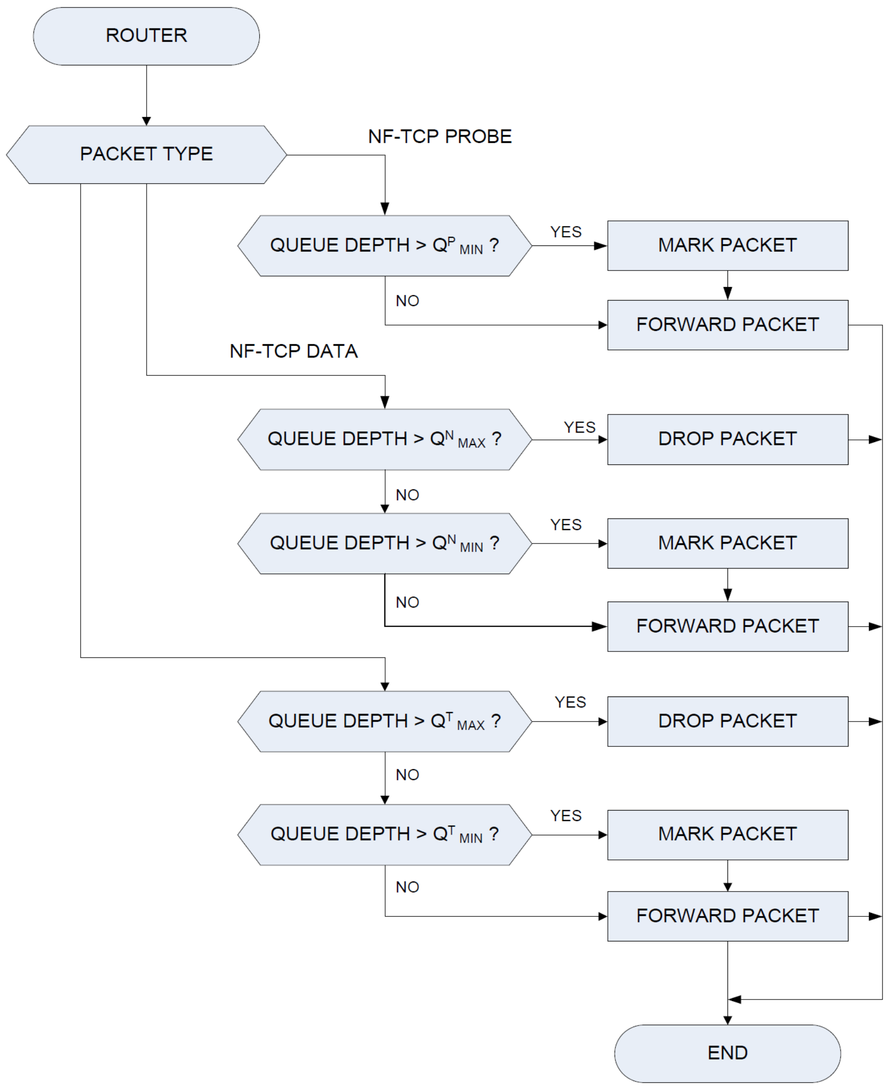


Figure 4.6: Router: How an NF-TCP aware router handles the various packets traversing it.

reduction on noticing ECN marks/packet drops affects overall throughput, especially in high BDP networks.

4.2.2 Congestion window increase

NF-TCP is designed to be aggressive during non-congestion periods, in order to be opportunistic in using available bandwidth. NF-TCP takes advantage of a bandwidth-estimation module to determine the rate of increase, so as to have an informed aggressive increase mechanism. This is unlike approaches that use an aggressive increase for large BDP networks, where they potentially cause congestion before backing off. The approach in this thesis enables NF-TCP to be truly friendly to existing transport connections. The estimate of the available bandwidth allows NF-TCP flows to aggressively utilize a certain percentage of the remaining available bandwidth. The increase is limited to a factor α of the estimated available bandwidth to allow inaccuracies in the measured estimate as well as differences in the time scales. On receiving an estimate of the available bandwidth ($\mathrm{BW_{est}}$), NF-TCP switches over to an aggressive increase phase wherein the congestion window is adjusted as follows:

$$\mathrm{ACK} : w \leftarrow w + \frac{\alpha * \mathrm{BW_{est}}}{w}, \text{ if } \mathrm{BW_{est}} > 0 \tag{4.10}$$

$$\mathrm{ACK} : w \leftarrow w + \frac{1}{w}, \text{ if } \mathrm{BW_{est}} = 0 \tag{4.11}$$

where ACK stands for acknowledgement received.

4.3 Bandwidth-Estimation Module

NF-TCP explores the use of a novel combination of bandwidth measurement and congestion control. The bandwidth estimation mechanism guides the decision of the congestion control framework in the appropriate time-scale to opportunistically use spare bandwidth. This feature is especially applicable for a network friendly transport that needs to be both submissive during congestion and aggressive during non-congestion periods. NF-TCP uses the bandwidth estimation module to estimate the available bandwidth. This estimate is used as a target value, up to which the flow can increase its rate, as long as it has not received an ECN marking. This enables NF-TCP to be opportunistic in using available bandwidth resulting in throughput optimization as well as increased network utilization without causing congestion.

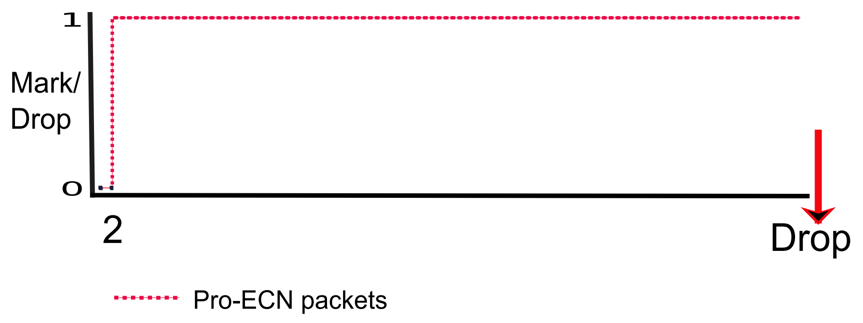


Figure 4.7: Instantaneous queue marking for ECN based probing mechanism

The bandwidth estimation module separates the measurement process for obtaining the available bandwidth from the congestion control. Whenever an estimate is not available, NF-TCP continues to use the standard conservative increase of 1 packet per RTT to be able to achieve fairness among NF-TCP flows.

4.3.1 Probing based on ECN (ProECN)

An ECN complimented probing mechanism that is based on pathChirp [3] is proposed here. Similar to pathChirp, ProECN uses a series of packets that have an exponentially reducing inter-packet spacing to measure a wide range of available bandwidths. The sender sends this stream of packets to simulate an increasing sending rate and utilizes self-induced congestion to identify available bandwidth. ProECN differs from pathChirp by using an ECN complimented approach to measure the available bandwidth instead of depending only on increasing delay estimates. For this purpose, a modified AQM queue is used here, that is able to perform instantaneous marking instead of the traditional EWMA based RED marking. The modified AQM queue identifies probe packets by the DSCP bit set in the header and marks them if the instantaneous queue size is greater than 1 so as to indicate self-induced congestion.

The bandwidth-estimation module sends the probe packets with varying inter-packet gaps to emulate a range of sending rates. When the sending rate is higher than the available bandwidth, the probe packets undergo self-induced congestion. This results in the packets having to wait in the queue and thus being ECN marked. The ECN marking acts as a reliable and early indicator of a packet having to wait in the queue and therefore enables the bandwidth-estimation module to obtain a reliable estimate of the available bandwidth. The estimated available bandwidth ($\mathrm{BW_{est}}$ (bps)) is described by:

$$\mathrm{BW_{est}} = \begin{cases} \mathrm{MinRate} * \mathrm{SF_{N-1}}, \\ \quad \text{if } \mathrm{BW_{Avail}} > \mathrm{Maxrate} \\ \mathrm{MinRate} * \mathrm{SF_{N-k}}, \\ \quad \text{if } \mathrm{Minrate} < \mathrm{BW_{Avail}} < \mathrm{Maxrate} \\ 0, \ \text{if } \mathrm{BW_{Avail}} < \mathrm{Minrate} \end{cases} \tag{4.12}$$

N is the number of probe packets and k is the first packet in the series that arrives with an ECN marking and/or at a time greater than all the previous packets and $\mathrm{BW_{Avail}}$ is the actual available bandwidth on the link.

The ECN marking compliments the delay-based approach of pathChirp since it exploits feedback received from the intermediate routers instead of having to depend only on delay measurements that could have high noise in low latency networks. With this enhancement, a source is able to identify excursion segments (a period of increasing delays) more accurately. The actual analysis and the heuristics utilized are similar to that described in [3] to account for bursty traffic.

RAPID [14] also uses a pathChirp like mechanism to perform a rate-based transmission in which all data packets are part of a continuous logical group of N probe packets; NF-TCP on the other hand uses ProECN only for probing and employs a window based transmission for the data packets.

4.3.2 Dynamic probing algorithm

NF-TCP's bandwidth estimation module is designed to dynamically adjust the measurement probe rate to get an estimate of the available bandwidth quickly, while limiting the overhead introduced in the network. This ensures that the probing mechanism can function efficiently in networks ranging from low BDP networks to high BDP networks. Currently, by design, the minimum probe rate is set to 1Mbps to prevent probing in networks with an available capacity that is lesser than 1Mbps.

An NF-TCP flow generates two kinds of packets: normal data and probe packets. The bandwidth-estimation module starts after the first RTT on receiving an acknowledgement of the initial data packets, and only if there are no losses or ECN markings received. This is to ensure that newly starting flows do not contribute to congestion caused by the probes. The estimate of the available bandwidth $\mathrm{BW_{est}}$ depends on the $MinRate$, the spread factor (SF) and the number of packets (N) (Eq. 4.12). The N probe packets are sent with varying inter-packet spacing so that available bandwidth can be measured in the range of $MinRate$ to $MaxRate$, such that the probes rate (r_i) is given by:

$$r_i = MinRate * (SF)^{i-1} \tag{4.13}$$

$$MaxRate = MinRate * (SF)^{N-2} \tag{4.14}$$

Similar to RAPID [14] the bandwidth-estimation module starts probing in a slow-start manner starting with 2 packets and doubling the number of packets afterwards. The slow-start phase is exited when the size of the probe train reaches N or when the estimated bandwidth is lower than the $MaxRate$. On exiting slow-start, the probing transits into a dynamic probing mode. Here, the average sending rate (r_{avg}) is set to $\alpha * \mathrm{BW}_{\mathrm{est}}$. PathChirp probe packets are limited in quantity for a particular probe event and the SF is set to a fixed value (for evaluations conducted in this thesis, $N = 15$ and $SF = 1.2$). On receiving an estimate of the available bandwidth, the probing mechanism is restarted after a uniformly distributed time period with a mean value equal to that of the baseRTT. The new $MinRate$ is calculated according to the last $\mathrm{BW}_{\mathrm{est}}$ and is given by:

$$MinRate = \frac{SF^{N-1} - 1}{(N-1)(SF-1) * SF^{N-2}} * r_{avg} \tag{4.15}$$

Figure 4.8 shows a flow chart representation of the probing mechanism at the sender side. One can observe how the probe mechanism increases and decreases its probing range based on the estimated bandwidth.

Figure 4.9 shows a flow chart representation of the probing mechanism and its interaction with the flow rate module, at the sender side. One can observe that the mechanism switches off on receiving a continuous bandwidth estimate of zero and is restarted after a time period of `probefreq`.

Dealing with the loss of chirp packets

There is a possibility of the chirp packets or for the packet carrying the estimated bandwidth getting lost in the network. It could also happen that the a chirp is restarted due to the expiry of timers. To deal with such extreme cases is important.

In the current implementation, a simple algorithm is used to avoid problems with loss events. Figure 4.10 shows a flow chart of the chirp packet loss control at the receiver. The receiver can detect packet losses or packets that are out of sequence by tracking their packet numbers in `pktnum`. When a

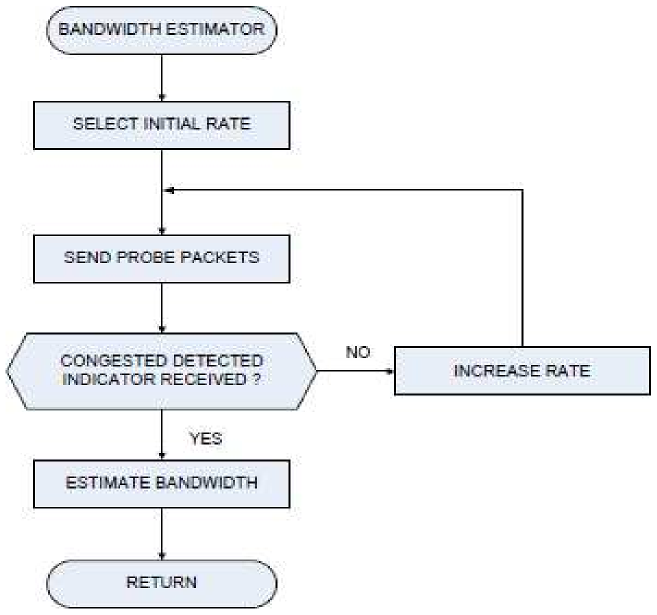


Figure 4.8: Sender side: Flow chart of the probe based bandwidth estimation mechanism

packet is lost or when the chirp packets are transmitted in the wrong order, the receiver will observe an unexpected packet number on the next received packet. When the last packet of a chirp is lost, the timer implemented in `PcTimeoutTimer` times out and the whole chirp will get retransmitted. When an unexpected packet arrives, the receiver simply bases its bandwidth estimation on the packets that were received before the unexpected packet arrived. Further on, it ignores all the incoming chirp packets until a chirp packet arrives which carries a packet number of zero, thus indicating that a new chirp was started.

4.4 Candidate approaches and their need for network support and endhost support

NF-TCP and various other solutions try to provide the ISPs with different business models and means to efficiently use capacity, and try to cater to the varying needs of the applications. Table 4.2 outlines some of the proposals and their need for endhost as well as network support.

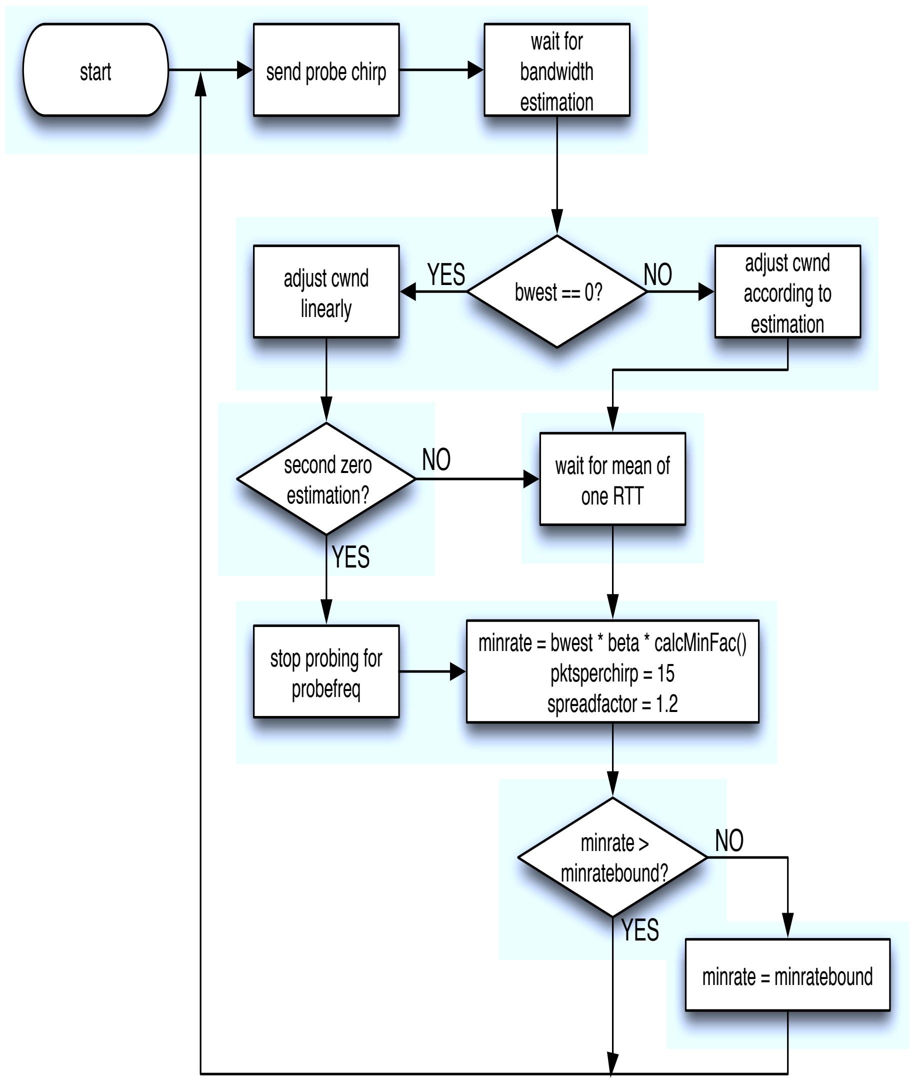


Figure 4.9: Sender side: Flow chart of probe based Bandwidth Estimation module and Flow Rate module interaction

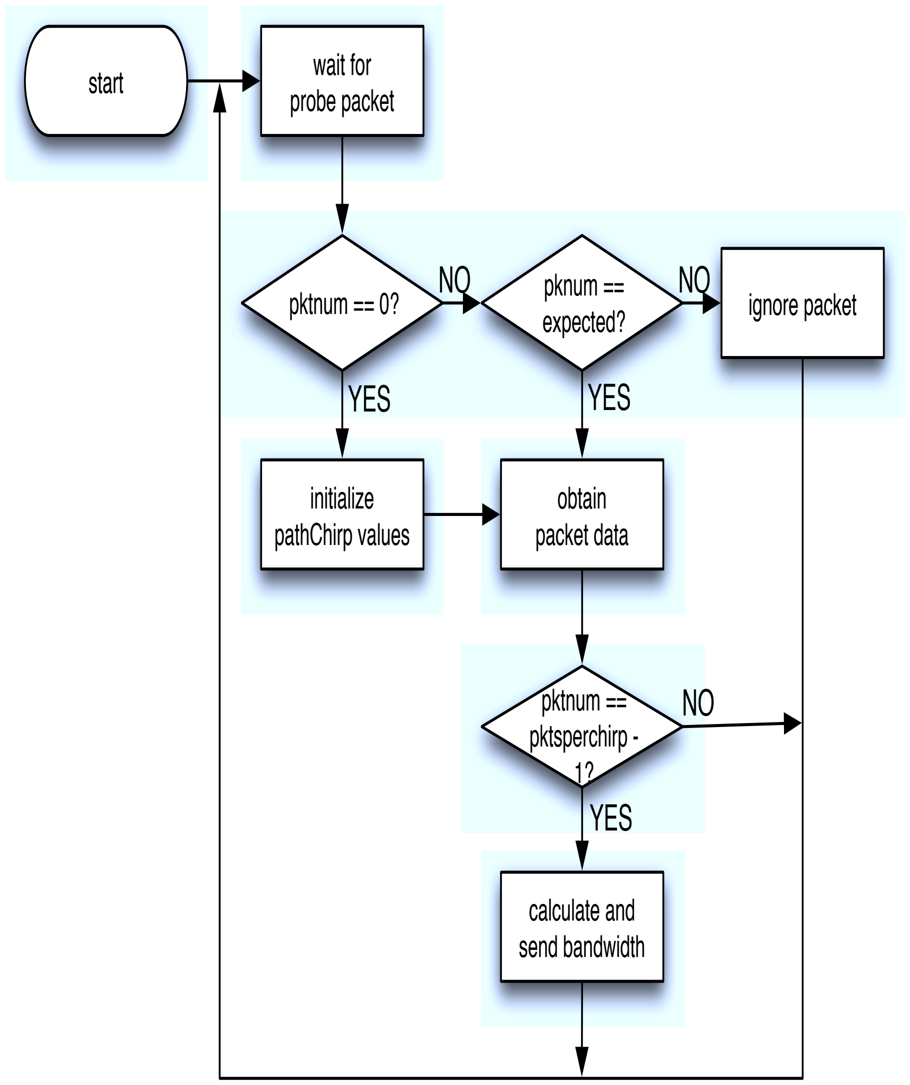


Figure 4.10: Receiver side: Flow chart of chirp packet loss control

Table 4.2: The various solutions and their requirement of application support, network support and the applications that they can support.

	Usage of Congestion	Network Support	Endhost support	Example Applications
ALTO [12]	Can be used	Yes	Yes	P2P, Live streaming
LEDBAT [2]	Implicit (delay based)	No	Yes	P2P, delay-insensitive
Comcast solution [56]	70-80% bandwidth usage	Yes	No	All
RAPID [14]	via probing	No	Yes	P2P
NF-TCP	Explicit	Yes	Not mandatory	P2P, delay-insensitive, download and play

Chapter 5

Evaluation

NF-TCP was implemented on Linux kernel 2.6.31 and the Linux TCP implementation ns-2 tool [57, 58] was used to import the Linux-based implementation of NF-TCP as well as the existing TCP Reno and TCP-LP onto ns-2. This enabled tests in a wide range of topology, scale and simulation time. Additionally, this allowed comparison of NF-TCP performance against other candidate proposals such as LEDBAT and RAPID which were developed on ns-2.

Starting with a single bottleneck scenario, more sophisticated scenarios with RTT heterogeneity and multiple bottlenecks with several flows (Fig. 5.9) are illustrated further. The bottleneck link routers (Router1, 2 and 3) maintain a modified RED queue with different threshold values for NF-TCP flows, and a normal RED queue [59][60][61][62] for TCP flows, as shown in Table 5.1. Routers have a buffer capacity equal to the link BDP. FTP is used and a SACK is generated for every received data packet. Packet size is set to 1000 bytes (including the IP header) and the initial *ssthresh* for Reno/SACK is set to 100 packets. Bottleneck capacity is 600 Mbps and RTT is 100 ms.

5.1 Comparison with Other Approaches

In this section, NF-TCP and other candidate approaches are evaluated. At first, the focus is on the performance of a single candidate (NF-TCP/LEDBAT/

Table 5.1: The used RED queue parameters for evaluation (Bandwidth = 600Mbps, RTT = 100ms, queue size= 7500 packets)

	NF-TCP	Standard TCP
Min-threshold (packets)	75 (= 1% of queue size)	6800
Max-threshold (packets)	3750 (= 50% of total queue size)	7500

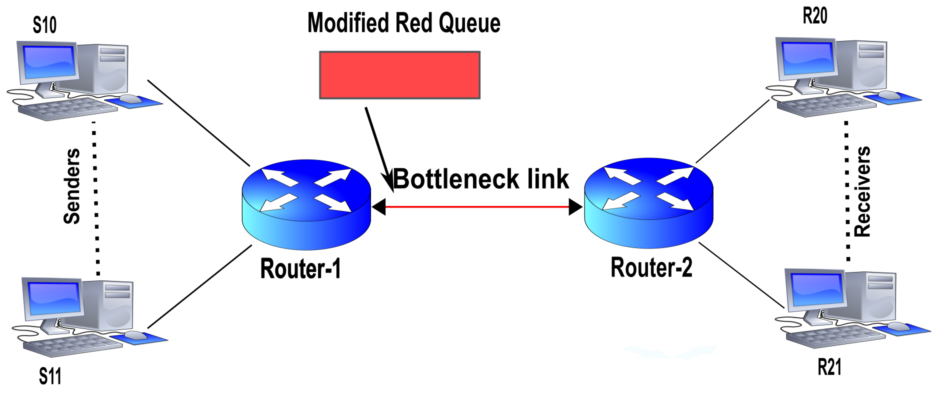
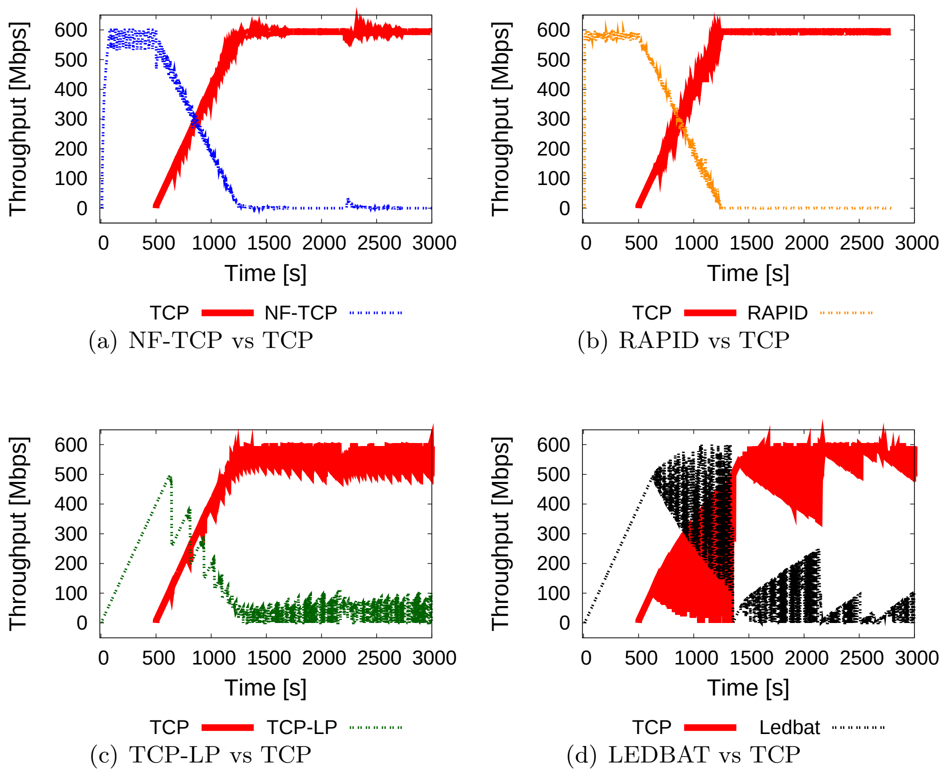
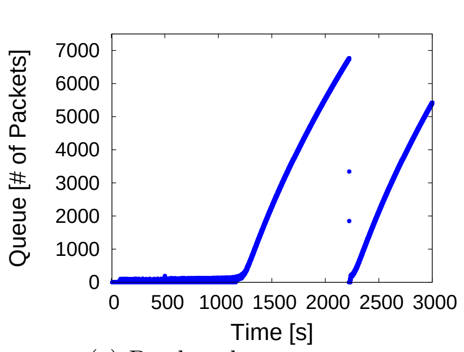


Figure 5.1: Single-hop dumbbell topology

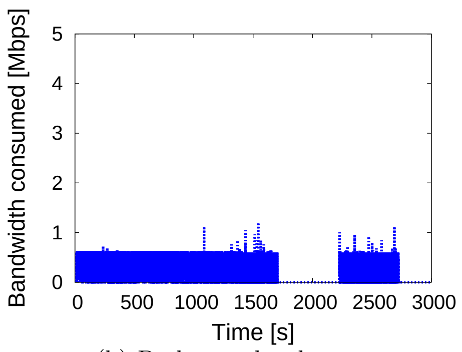


(a) NF-TCP vs TCP
(b) RAPID vs TCP
(c) TCP-LP vs TCP
(d) LEDBAT vs TCP

Figure 5.2: Dumbbell topology: Instantaneous throughput of a candidate flow in the presence of a TCP flow

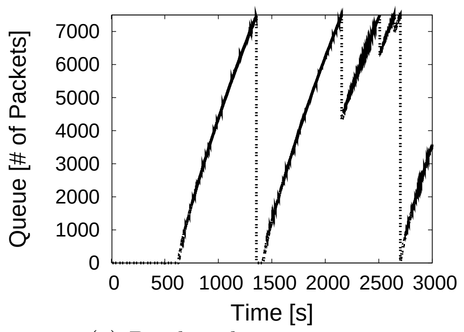


(a) Bottleneck queue

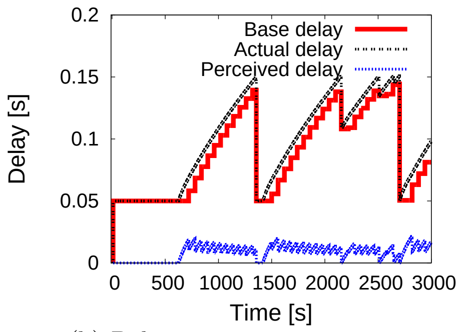


(b) Probe overhead

Figure 5.3: Impact of NF-TCP on the network



(a) Bottleneck queue



(b) Delay measurements

Figure 5.4: LEDBAT is unfriendly: Causes substantial delay

TCP-LP/ RAPID) flow in the presence of a single standard TCP flow. The topology is used as shown in Fig. 5.1 with the candidate flow being sent from S11 – R20 and the reference flow being sent from S10 – R21 and all flows traversing the bottleneck link at Router1.

Fig. 5.2 illustrates the instantaneous throughput of the network friendly flows versus a competing standard TCP flow. Fig. 5.2(a) shows that the NF-TCP flow is able to opportunistically utilize the bandwidth in the period from 0-500s with the support of its ProECN bandwidth estimation. NF-TCP is comparable in its aggressive increase phase to the most aggressive of the alternatives, RAPID (Fig. 5.2(b)). RAPID was designed for use in high BDP networks and hence is aggressive in its startup. On the other hand, TCP-LP (Fig. 5.2(c)) and LEDBAT (Fig. 5.2(d)) are much slower in their increase and hence are unable to fully utilize the uncongested network during this time period.

From 500s onwards, as TCP increases its demand, NF-TCP quickly reduces its load, as a result of ECN marking, allowing TCP to grow its bandwidth as much as it desires. NF-TCP is thus submissive to TCP. TCP is not impacted after time t=1200s. In this particular case, RAPID is also submissive. Again, TCP-LP and LEDBAT are much less submissive, yielding bandwidth to TCP more slowly. Further, once TCP nearly attains its full window at 1200 secs, TCP-LP and LEDBAT impact the TCP flow to different extents. In fact, when co-existing with LEDBAT, TCP continually experiences significant loss and reduced throughput, and thus incurs both additional delay and lower throughput. Thus, this experiment demonstrates both the capabilities of NF-TCP: to be opportunistic in its use of available bandwidth and submissive in the presence of TCP.

The performance of a LEDBAT flow was evaluated versus a standard TCP flow. It is true that LEDBAT flows are network friendly to standard TCP flows as reported in [63], however only under *low BDP* scenarios. When the bottleneck bandwidth becomes higher (i.e., 200Mbps and more) and the buffer size is set equivalent to the link BDP (more realistic with higher speed links), results obtained show that LEDBAT flows are no longer friendly to TCP flows. Fig. 5.2(d) demonstrates that LEDBAT is more aggressive than NF-TCP (and TCP-LP) during congestion periods. Fig. 5.4(a) shows that, as the queue builds up, the base-delay stored by LEDBAT increases (Fig. 5.4(b)). This is due to the resetting of the baseRTT every 2-10 minutes and results in LEDBAT increasing its throughput. In short, the results demonstrate that LEDBAT does not satisfy the requirement of a network friendly protocol to maintain low queues and being submissive to TCP over a reasonable wide range of system parameters.

Table 5.2: The used RED queue parameters for evaluation (Bandwidth = 600Mbps, RTT = 200ms, queue size= 15000 packets)

	NF-TCP	Standard TCP
Min-threshold (packets)	150 (= 1% of queue size)	13600
Max-threshold (packets)	7500 (= 50% of total queue size)	15000

5.1.1 Bottleneck Bandwidth = 600 Mbps, RTT = 200ms

In this section, NF-TCP and other candidate approaches were evaluated using the topology shown in Fig. 5.1, but the RTT was increased to 200ms. The candidate flow was sent from S11 – R20 and the reference flow was sent from S10 – R21 and all flows traversed the bottleneck link at Router1. The RED queue parameters were changed as shown in Table 5.2 to reflect the increase in RTT.

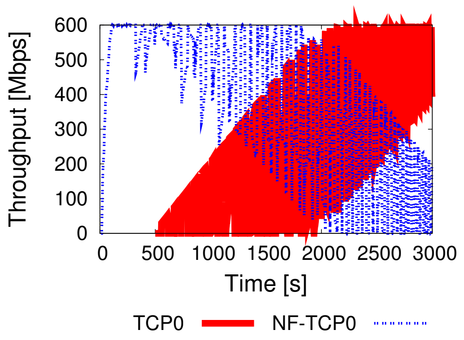
Fig. 5.5 illustrates the instantaneous throughput of the network friendly flows versus a competing standard TCP flow. Fig. 5.5(a) shows that the NF-TCP flow is still able to opportunistically utilize the bandwidth in the period from 0-500s with the support of its ProECN bandwidth estimation. Moreover NF-TCP and RAPID are able to be submissive, whereas the other candidate flows struggle in their performance to be submissive.

5.1.2 Bottleneck Bandwidth = 1Gbps, RTT = 100ms

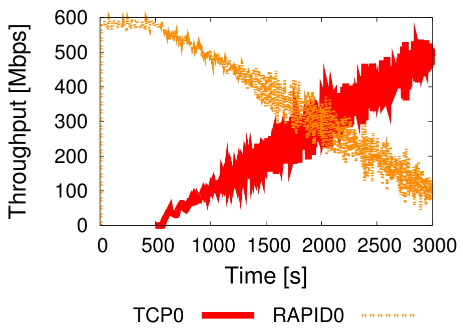
In this section, NF-TCP and other candidate approaches were evaluated using the topology shown in Fig. 5.1, but the bottleneck bandwidth was increased to 1Gbps. The candidate flow was sent from S11 – R20 and the reference flow was sent from S10 – R21 and all flows traversed the bottleneck link at Router1. The RED queue parameters were changed as shown in Table 5.3 to reflect the increase in Bandwidth.

Fig. 5.6 illustrates the instantaneous throughput of the network friendly flows versus a competing standard TCP flow. Fig. 5.6(a) shows that the NF-TCP flow is still able to opportunistically utilize the bandwidth in the period from 0-500s with the support of its ProECN bandwidth estimation. Moreover NF-TCP and RAPID are able to be submissive, whereas the other candidate flows struggle in their performance to be submissive.

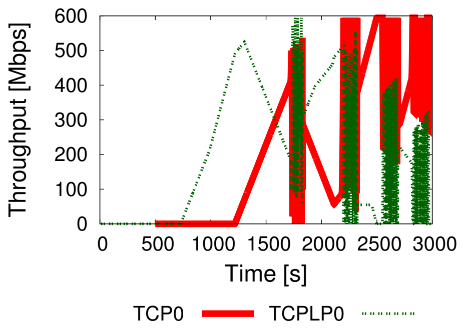
The readers are requested to refer to [43] for results illustrating the performance of NF-TCP in networks with different BDPs.



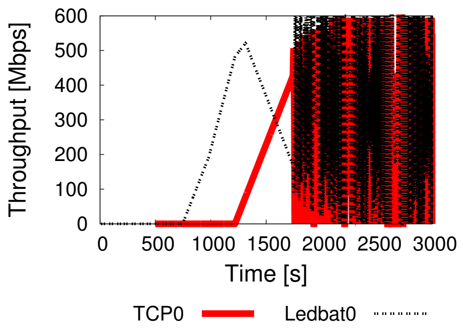
(a) NF-TCP vs TCP



(b) RAPID vs TCP



(c) TCP-LP vs TCP

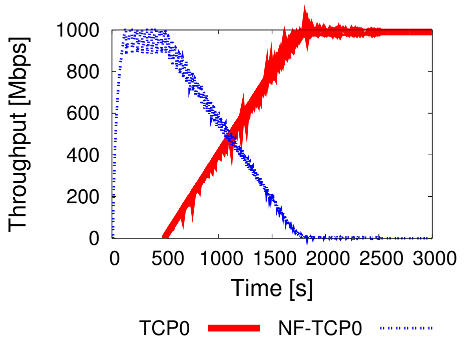


(d) LEDBAT vs TCP

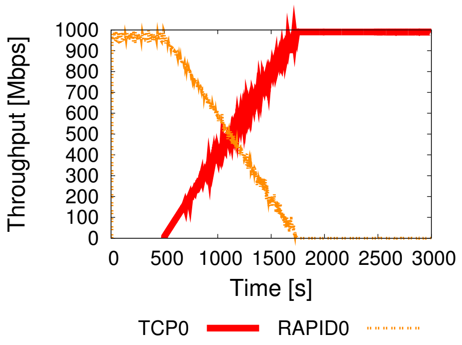
Figure 5.5: Dumbbell topology: Instantaneous throughput of a candidate flow in the presence of a TCP flow (**Bottleneck Link Bandwidth = 600Mbps, RTT = 200ms**)

Table 5.3: The used RED queue parameters for evaluation (Bandwidth = 1Gbps, RTT = 100ms, queue size= 125000 packets)

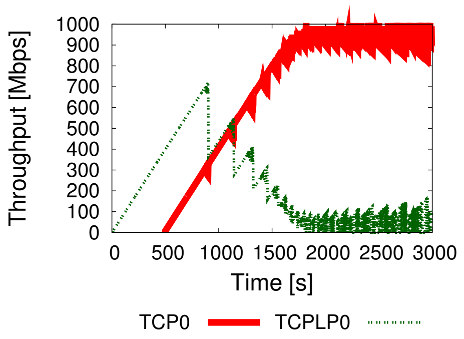
	NF-TCP	Standard TCP
Min-threshold (packets)	1250 (= 1% of queue size)	113300
Max-threshold (packets)	62500 (= 50% of total queue size)	125000



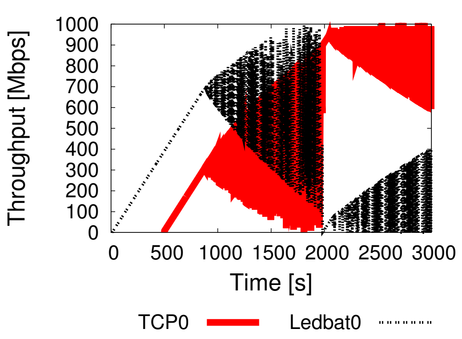
(a) NF-TCP vs TCP



(b) RAPID vs TCP



(c) TCP-LP vs TCP



(d) LEDBAT vs TCP

Figure 5.6: Dumbbell topology: Instantaneous throughput of a candidate flow in the presence of a TCP flow (**Bottleneck Link Bandwidth = 1000Mbps, RTT = 100ms**)

5.2 NF-TCP in the Presence of Standard TCP

5.2.1 The impact on queue sizes

Fig. 5.3(a) illustrates the instantaneous queue length for NF-TCP with bandwidth estimation competing against a standard TCP flow. The NF-TCP flow maintains a near zero buffer ($< Q^N_{min}$) until TCP achieves maximum utilization of the link at about 1200s. 1200s onwards, it is the TCP flow that contributes to the increase in buffer utilization and the increase in queue coincides with the increase in throughput for TCP (Fig. 5.2(a)). This behavior of NF-TCP ensures that it yields its share of the bandwidth to TCP flows and is also able to have a minimum impact on TCP flows that start later. The results also show that an NF-TCP flow maintains a low queue size irrespective of the number of NF-TCP flows, as reported in subsection 5.5.

5.2.2 ProECN dynamic bandwidth estimation

The performance of ProECN bandwidth estimation tool within NF-TCP with a varying measurement range is determined here. Probes are sent about once per 2 RTTs. Fig. 5.3(b) illustrates that it is able to provide NF-TCP with an estimate of the available bandwidth while having an average probe throughput of about 0.6Mbps. The minimum rate that can be probed is limited to 1Mbps to prevent congestion when network capacity is less than 1Mpbs. With ProECN, NF-TCP should ideally switch off during congestion periods, as confirmed by experiments conducted: in the period ranging from 1800-2200s and 2700s and beyond, when the NF-TCP becomes completely submissive.

5.2.3 NF-TCP with and without bandwidth estimation

Fig. 5.2(a) and Fig. 5.7(a) show the difference between NF-TCP with and without bandwidth estimation. The NF-TCP' flow in the latter case is a TCP-RENO flow that gets preferential early marking. In both cases, due to the early warning provided by the modified RED queue, NF-TCP' is able to be submissive. However, in the case of NF-TCP' (without bandwidth estimation), the flow cannot opportunistically utilize the bandwidth and therefore suffers by having a lower overall throughput.

5.2.4 NF-TCP vs UDP cross traffic

The ability of NF-TCP to opportunistically utilize available bandwidth in the presence of UDP cross traffic (rate generated from a Poisson distribution) has also been checked and evaluated. Fig. 5.7(b) illustrates that NF-TCP opportunistically get close to the available bandwidth and then resorts to a slower increase. This results in better link utilization and also allows it to be friendly to other flows.

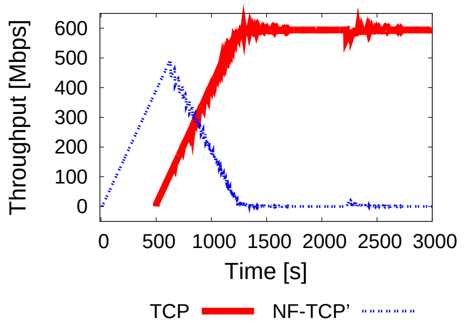
5.3 Fairness among NF-TCP Flows

To study the fairness among NF-TCP flows, the following two scenarios are chosen: a) 5 NF-TCP flows started at the same time, b) 5 NF-TCP flows starting one after another every 50 secs. Fig. 5.8(a) and Fig. 5.8(b) show that the 5 NF-TCP flows are fair to one another. Fig. 5.8(b) also shows that since the NF-TCP flow (NF-TCP0) that started at time zero did not have any competing flow, it was able to utilize the available bandwidth completely and yielded its share of the bandwidth when the other flows started. A decrease factor of 12.5% instead of the standard 50% makes it longer to achieve fairness but ensures that the overall link utilization is still high. With different RTTs, NF-TCP is also able to achieve a fairness similar to that achieved by standard TCP in the same scenario.

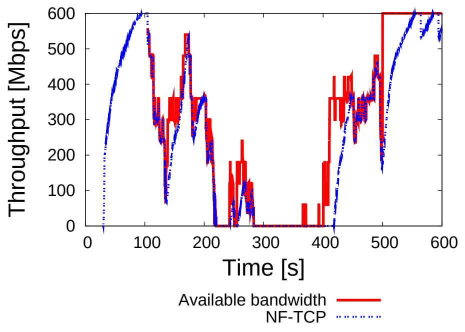
5.4 Candidate Flows with Multihop, Varying RTTs

More realistic scenarios where the TCP flow traverses multiple hops going over several congested routers are looked upon. They compete with flows that have different RTTs. For this purpose, the testbed is setup as shown in Fig. 5.9 with three 'bottleneck' links. The RTT on the longest path from S10/S11 to R40/R41 is three times the RTT on the shorter single hop paths. Evaluations were conducted with a TCP flow and a competing candidate flow from S10 to R20, S20 to R30 and S30 to R40. The candidate flows are started at 0s and the TCP flow is started at 500s.

Fig. 5.10(a) illustrates that although NF-TCP has a much shorter RTT, it is friendly towards TCP flows while also opportunistically utilizing the spare bandwidth. TCP-LP (Fig. 5.10(c)) is also friendly to standard TCP, but is not completely submissive. Both RAPID (Fig. 5.10(b)) and LEDBAT (Fig. 5.10(d)) are also not submissive. For RAPID, this is due to the com-

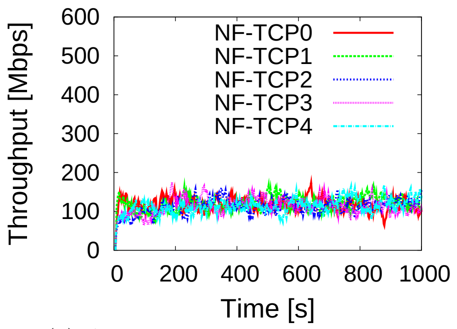


(a) NF-TCP w/o BW_{est}

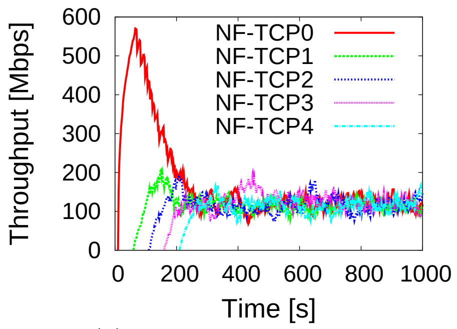


(b) vs Poisson traffic

Figure 5.7: Need for bandwidth estimation



(a) All flows begin together



(b) Staggered start

Figure 5.8: Fairness among NF-TCP flows in a single hop scenario

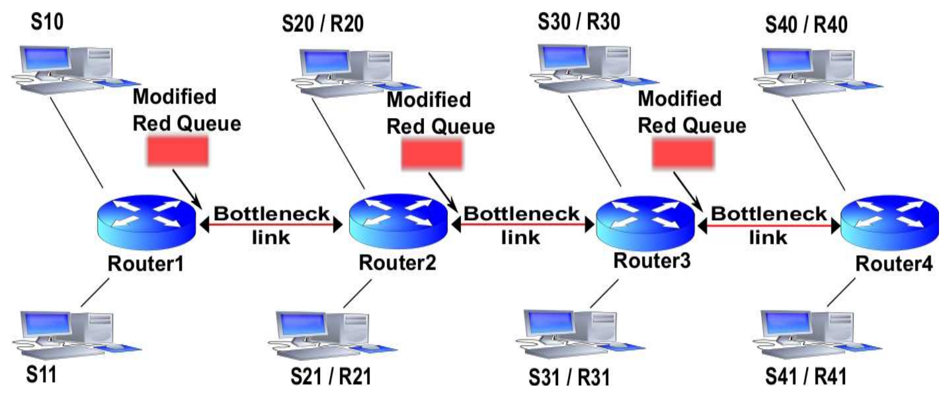


Figure 5.9: Multi-hop topology (RTT for TCP = 3 times RTT of candidate flow)

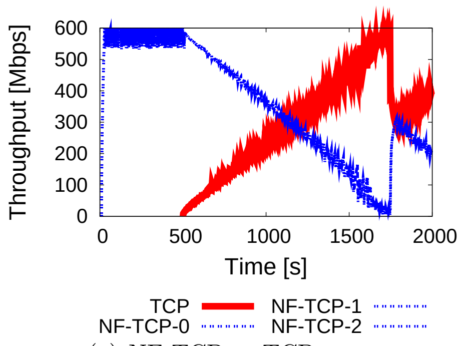
bination of its aggressive nature and its complete reliance on delay-based probing to measure available bandwidth. RAPID sends a continuous burst of about 20 packets in a chain with the average rate of the chain equal to that of the recently estimated available bandwidth. Each of these chains measures an estimate of the available bandwidth and this is a continuous process. Unfortunately, as the RTTs become diverse, some of the probe trains measure total available capacity instead of the available bandwidth resulting in an increase in throughput for RAPID. Extensive evaluations show that for RAPID to function well, its parameters have to be carefully adjusted for varying network scenarios and this is nearly impossible for complex networks.

5.5 Scalability Test

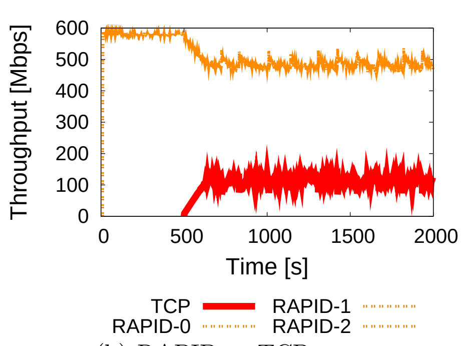
For this purpose, a dumbbell topology with a bottleneck bandwidth of 250Mbps and an RTT of 100ms has been created.

5.5.1 Impact of varying number of candidate Flows

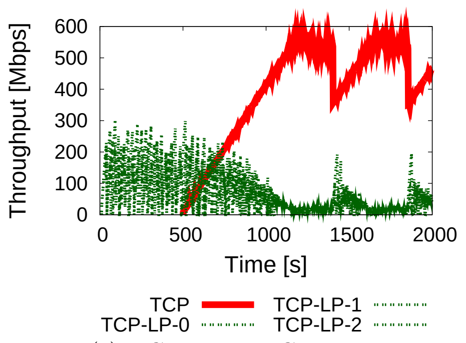
First, the average throughput of five standard TCP flows competing against a varying number (from 1 to 200) of candidate flows is determined. All the flows are started at the beginning. Fig. 5.11 shows that the NF-TCP flows have the least impact on TCP and they obtain an average throughput of 220Mbps. It can be observed that TCP-LP and LEDBAT reduce the throughput of TCP further by about 50Mbits. Of course, in the pres-



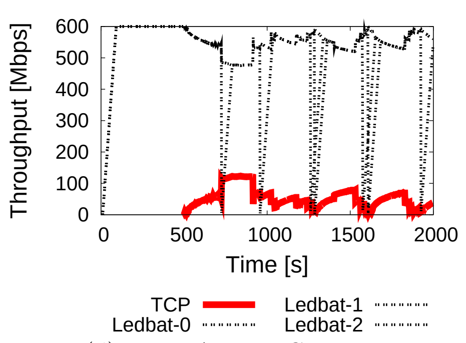
(a) NF-TCP vs TCP



(b) RAPID vs TCP



(c) TCP-LP vs TCP



(d) LEDBAT vs TCP

Figure 5.10: Multihop topology: Instantaneous throughput of candidate flows in the presence of a TCP flow (RTT of candidate flows = 1/3 RTT of TCP flow)

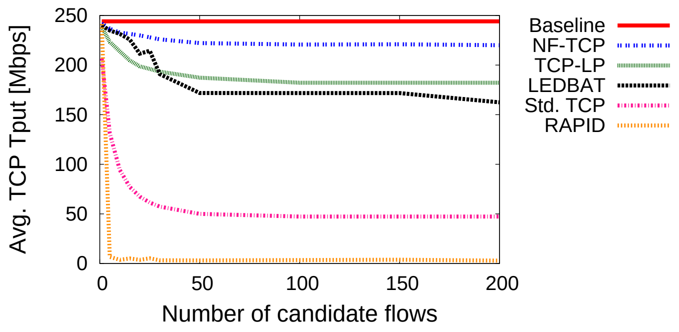


Figure 5.11: Impact of candidate flows on TCP (5 TCP flows)

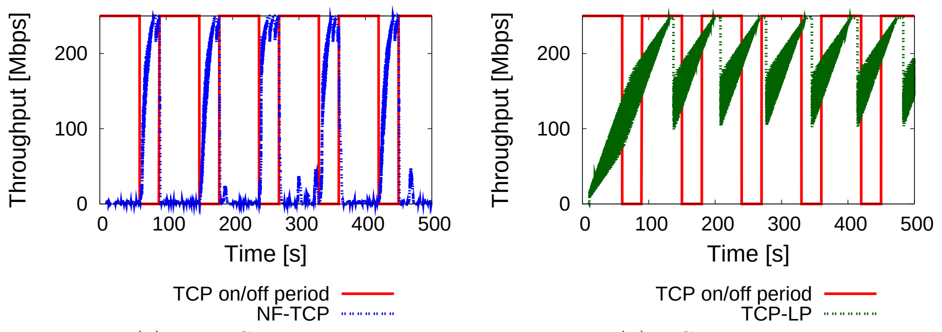


Figure 5.12: A candidate flow with 100 TCP flows that arrive in bursts

ence of other standard TCP flows (as in the current Internet), the reference TCP flows lose about 80% of their throughput. Finally, with RAPID, the reference TCP flows get near zero throughput.

5.5.2 Impact of candidate flows on bursty TCP traffic

Now, the instantaneous throughput of a candidate flow and its impact on a burst of TCP traffic (100 TCP flows) that arrive every 90s and have a lifetime of 60s (i.e., ON-OFF TCP traffic) are deduced. Fig. 5.12 shows that NF-TCP yields quickly to TCP and is able to aggressively utilize the spare bandwidth. TCP-LP, which is one of the better candidates is unable to be submissive in such short time scales.

Chapter 6

Deployment and Other Considerations

The key requirement of NF-TCP like protocols is to detect the onset of congestion earlier than standard TCP flows. This provides ample time for such flows to become submissive, thereby reducing the adverse impact on delay sensitive flows. To achieve this goal, the following deployment options are considered:

low-priority DSCP codepoint [1]

re-ECN

PCN

6.1 Usage of low-priority DSCP codepoint [1]

NF-TCP is designed to use the standard ECN bits to receive congestion feedback. Nevertheless, it is required to distinguish the NF-TCP flows from other kinds of flows.

RFC [1] proposes the use of a low-priority DSCP code to identify traffic that would fit under the Low-Priority Data service class for selected background applications where data transfer can be delayed or suspended during peak network load conditions. In this work, the use of the same codepoint to identify NF-TCP flows is proposed. Intermediate core routers can then identify NF-TCP flows and use the standard ECN bits to perform early marking.

6.2 Usage of Pre-congestion Notification(PCN) marking

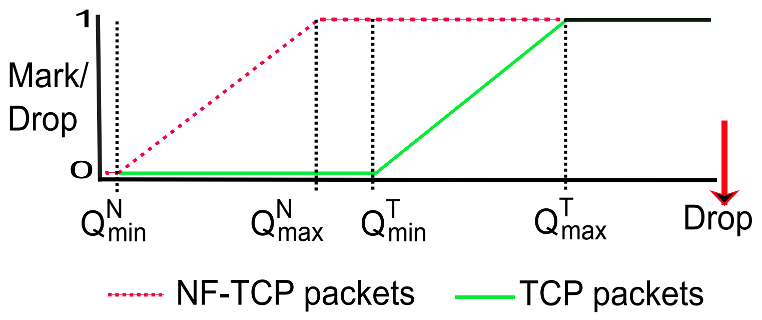
Pre-Congestion Notification [44][64][65][66] is currently proposed for non-elastic traffic to perform admission control and flow termination. The possibility of using a PCN like approach is considered wherein the flows are AC marked and FT marked. The NF-TCP flows could react on seeing AC marking whereas FT marking could be analogous to the current ECN marking and all flows could react accordingly. Note that this approach does not interfere with the standard PCN packets that use a separate virtual queue. This marking is performed by intermediate routers to indicate admission control and flow termination. It is used to perform early marking for NF-TCP flows. With PCN being standardized by the IETF, the routers could employ a PCN like marking wherein two marking thresholds exist. The lower marking threshold is for AC and marks all the packets whereas the FT threshold is higher and performs excessive marking. The same token bucket could be used to perform AC marking on NF-TCP flows to indicate that they could reduce their rate. The downside is that AC marking marks all packets and therefore cannot be used for volume charging and intermediate router cannot distinguish between an AC and an FT marked packet. Figure 6.1 illustrates the difference between the early marking required by NF-TCP and how PCN AC and FT marking can be adapted to perform NF-TCP marking.

For this solution to be effective, it requires three code-points. One code-point to identify NF-TCP flows, a second code-point to identify admission control marking and a third code-point to identify flow termination marking. In case, there is a lack of code-points, one could a solution similar to that provided in [67], [68] and [69]. Similar to this solution, a single code-point can be used to identify NF-TCP flows and the other code-point can be used to signal both admission control and flow termination.

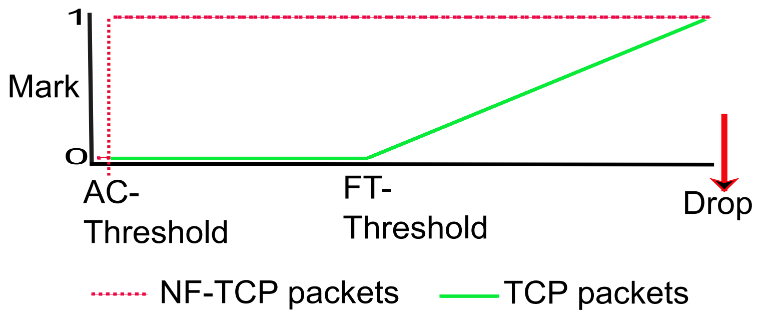
6.3 Usage of Re-ECN

Re-ECN [70] aims to place a premium on congestion and the trasnfer of data during this period. The applications on seeing congestion marks can decide to either lower their rate, continue to transfer at the same rate or increase their rate if they are willing to pay for congestion. To enable NF-TCP support in re-ECN, the following key issues need to be addressed:

- Distinguish between NF-TCP flows and standard re-ECN flows
- The routers need to be able to mark NF-TCP flows differentially



(a) NF-TCP early marking



(b) PCN marking adapted for NF-TCP

Figure 6.1: A router queue performing NF-TCP early marking (top) versus how PCN marking can be adapted for NF-TCP flows (below)

Providing a separate codepoint for NF-TCP

To differentiate a delay-sensitive flow from a non delay-sensitive flow, the unused codepoint in the re-ECN proposal is put into use. As shown in Table 6.1, the combination to indicate that the flow or packet is of a delay-insensitive application and therefore of low priority. As seen in Table 6.1, re-ECN uses the combination of ECN flag =00 and RE flag =1 is used to identify a re-ECN flow and the combination of ECN flag =00 and RE flag=1 is used to identify a NF-TCP flow, thus making it backward compatible too.

ECN Flag	RE Flag	RFC 3168 codepoint	re-ECN codepoint	**NF-TCP & Re-ECN codepoint**	Description
00	0	Not-ECT	Not-ECT	**Not-ECT**	Not re-ECN capable
00	1	-	FNE	**FNE**	Feedback not established
01	0	ECT(1)	Re-Echo	**Re-Echo**	Re-echoed congestion and RECT
01	1	-	RECT	**RECT**	Re-ecn capable
10	0	ECT(0)	ECT(0)	**ECT(0)**	ECN use only
10	1	-	-	**NF-TCP**	**NF-TCP capable**
11	0	CE	CE(0)	**CE(0)**	Re-Echo cancelled
11	1	-	CE(-1)	**CE(-1)**	Congestion experienced

Table 6.1: Description of the extended NF-TCP enabled re-ECN codepoints

Early Marking for NF-TCP flows

Early marking is a marking scheme in which a router needs to be enabled to perform differential marking. The router needs to have a different set of RED thresholds for NF-TCP flows in addition to the standard RED marking. The early marking is the ideal form of marking since it provides ample time for NF-TCP to react to the onset of congestion and become submissive. The network provider could use existing approaches such as Cisco's DWRED or the Linux GRED [71]. This ensures that the delay-insensitive flows are warned earlier and can therefore be of less impact to the delay-sensitive flows.

This allows the high priority application to make a decision on seeing marking. It can either decide to continue and get charged or if the rate is not appropriate enough, it can switch off. The alternative is to be unsure of how many low priority flows exist and therefore wait for an RTT for the low priority flows to reduce their sending rates, then wait for another RTT to decide on whether to continue sending at the rate or not.

The other advantages of earlier marking ensures that newly starting flows can be switched off at the beginning itself. Moreover, in a highly dynamic environment, this gives the router enough time to ensure that the low priority traffic are in fact switched off.

Selective marking of NF-TCP flows

Similar to the above case, this marking can also be performed by intermediate routers unaware of re-ECN as well as NF-TCP enabled re-ECN marking. This is the easiest to deploy. This does not require a change in the routers except that the router is set to mark NF-TCP packets as much as possible before resorting to the marking on RE-ECN packets.

Standard ECN congestion marking

This marking is performed by standard traditional routers that are unaware of re-ECN and NF-TCP enabled re-ECN and represent the currently deployed routers. The NF-TCP flows would see such routers during the incremental deployment phase. In such a scenario too, the use of early marking indicating the onset of congestion is recommended instead of the traditional marking that signals possible loss. The downside is that both NF-TCP and re-ECN flows would get early marking. This could result in re-ECN flows seeing more marking than the normal case resulting in a higher congestion charge. To mitigate this, it is recommended that when the egress notices large amount of NF-TCP traffic, it must remove the congestion marking for re-ECN flows so as to provide the NF-TCP flows ample time to reduce their sending rate. Moreover early marking must be factored in the congestion charging too.

Re-ECN performs congestion marking and it is expected that this information is used to penalize the heavy congestion causing flows by either charging them or performing rate base control. The problem with this approach is that delay-sensitive and delay-insensitive approaches see the marking at the same time. This leads to the confusion at the application side as to what is the next step. It is unreasonable to expect the application to make an informed decision since it has no idea of how many high priority flows are existing. Therefore the high priority flow has to either make a decision of slowing down or getting charged. The low priority flow on the other hand, on seeing the marking can reduce its sending rate, thus allowing the high priority flow to utilize the available bandwidth.

6.4 Discussion

The design, analysis and deployment of NF-TCP happened over a long period. In this chapter, I try to answer a number of questions that could be still lingering in the minds of the reader. This is based on questions from reviewers, conference and workshop attendees and peers.

Question 1: Is there a potential for unfairness if delay-sensitive applications decide not to use NF-TCP?

NF-TCP is designed by users and developers of delay-insensitive applications willing to use non-congested periods for such applications. In the future, schemes such as congestion-based pricing [72], caps on usage during peak hours, etc may encourage more users to choose NF-TCP. To overcome the perceived unfair and sometimes unnecessary disadvantage that delay-sensitive applications are subjected to by traffic from background delay-insensitive applications, ISPs have resorted to throttling or even blocking of traffic from heavy users during congestion. For example, the Fairshare solution [56] implemented recently by Comcast have already started throttling the traffic of heavy users during congestion periods. Such a scheme is unfair as there is no means for the user to identify congestion periods in the network. The use of NF-TCP would give users a tool to spread out their usage without manual intervention.

Question 2: Does it require any change at the end host?

In its simplest form, it does not need any change of the end-host other than a way of identifying that it is a NF-TCP flow. This kind of differentiation can be performed by enabling ECN as default in such flows and other standard flows not using ECN.

Question 3: Does it require change in the IP header?

NF-TCP is designed to use the standard ECN bits. It can, for example, use the low-priority DSCP code point [1] to identify an NF-TCP flow.

Question 4: What is the effect of ECN not widely enabled on NF-TCP?

ECN is deployed in most of the current routers and is available in most of the popular OSes. I recognize that it is not yet enabled by default, but envision that NF-TCP is another compelling reason, since it will allow more efficient use of network resources.

Question 5: How does the use of NF-TCP as a re-ECN mechanism affect applications?

Delay-insensitive applications can choose to use NF-TCP as their default transport protocol. This would ensure that such applications do not perform data transfer during congestion periods and therefore do not contribute to congestion debts. Moreover, this allows the application to function seamlessly as a submissive protocol without the need for user intervention. Users can have the possibility to manually change the transport protocol used in standard re-ECN to override the submissive nature in case they want a certain file urgently or when enough congestion credits are in hand.

Question 6: How does the NF-TCP enhancement affect Re-ECN positive marking?

On receiving early marking, NF-TCP is designed to reduce its sending rate. The NF-TCP can then set positive re-ECN marking proportional to the marking received. This would help intermediate ingress and egress routers to monitor the NF-TCP flows and take action when required. It is important to note that the positive packets should have less weightage when taken into congestion debts. This also ensures that the delay-sensitive flows are congestion marked only in the presence of other high priority delay sensitive flows and can therefore decide if the cost is worth it.

Question 7: How does the NF-TCP enhancement affect volume based charging mechanisms?

NF-TCP flows will support the seamless reduction of congestion and result in only high priority delay sensitive flows using the network during congestion periods. The users of such applications must be accountable for the congestion they cause. The congestion caused by NF-TCP flows must have less weightage and in fact neglected if the NF-TCP flow takes evasive action by reducing its rate to a very low value.

Question 8: How does the NF-TCP enhancement for re-ECN function in the absence of early marking queues?

In this case, the application on seeing re-ECN marks would reduce their sending rate, whereas delay-sensitive high priority flows would continue to transfer data at the rate at which they were sending and thereby accumulate congestion payments. The disadvantage of such an approach is that NF-TCP flows would recognise congestion at the same time re-ECN flows do and hence not giving them enough time to react. Moreover, the re-ECN flows would not know if they are in the presence of other re-ECN or NF-TCP flows and therefore would not know whether to maintain the same rate or reduce their rate. This would have an impact on re-ECN flows and would break the traditional congestion approaches.

Question 9: Does the NF-TCP solution require changes to all routers along the path?

NF-TCP can function efficiently even when only the bottleneck routers are NF-TCP enabled. ISPs that are interested in optimizing their network capacity can choose to deploy NF-TCP enabled routers in access networks and other potential bottlenecks.

Question 10: What are the deployment incentives?

In addition to the use of NF-TCP for delay insensitive applications, there are several other deployment incentives such as:

• Usage of unused bandwidth: NF-TCP with the help of the bandwidth estimation tool is able to make better use of unused bandwidth. Due to the support of the network assisted early congestion detection, it has a robust mechanism to react to the onset of congestion.

• Data pre-fetching: NF-TCP can be used by smart applications that can anticipate future requirements and pre-fetch data accordingly whenever bandwidth is available.

• Multiple paths: NF-TCP can be modified to also support multipath so that it is able to use non-congested paths for the bulk of their transfers.

• Rerouting: NF-TCP depends on ECN marking and can therefore recover reasonably within a few RTTs after a change in route unlike delay based mechanisms, which will have to rely on their prediction models.

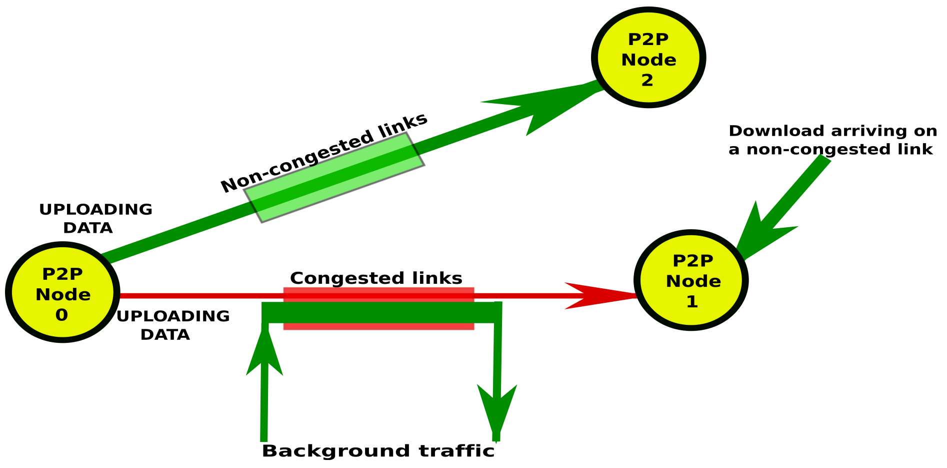


Figure 6.2: An example of how an uploading peer reduces the traffic it sends on congested link and how a downloading peer uses non-congested links to download data

- Usage of non-congested routes: Figure 6.2 illustrates the scenario wherein the application is able to use a non-congested route based on the congestion feedback it obtains.

Question 11: What is the purpose of the congestion control framework with respect to the NF-TCP?

The congestion control framework ensures that the NF-TCP is adaptable in the future. It ensures that NF-TCP can make use of available mechanisms and when efficient mechanisms are discovered, NF-TCP can seamlessly use it.

Question 12: Is the congestion control framework similar to the congestion manager?

The congestion manager [73] is present in an endhost and performs congestion management for all the connections present. The congestion control framework on the other hand is applicable per flow. The two approaches can co-exist and also compliment each other.

Chapter 7

Conclusion and Future Work

This thesis presents a "Network Friendly" transport protocol, which is able to meet the modern application requirements whilst optimizing the usage of network resources. Optimizing network resources include efficient use of spare capacity, postponing their use of the network as well as avoiding congestion that harms flows in the networks.

This thesis proposes a **congestion control framework** and uses it to design the NF-TCP protocol. Taking the NF-TCP design as an example, this thesis shows that the decoupled framework provides flexibility and adaptability to transport protocols. This allows applications to request for a specific transport protocol behaviour. Additionally, the transport protocols can use network support when available, to optimize their performance.

Network friendly TCP (NF-TCP) allows delay-insensitive applications to be **submissive** to standard TCP flows during congestion periods. Additionally, NF-TCP exploits a novel combination of adaptive measurement of available bandwidth and traditional window based congestion control to **efficiently utilize network capacity**. **NF-TCP outperforms other candidate approaches** such as LEDBAT, TCP-LP and RAPID and is shown to be more reliable due to its use of network feedback. With the use of extensive evaluations, this thesis also illustrates the following:

• NF-TCP meets the requirements of a network friendly transport protocol to be submissive.

• NF-TCP is able to detect congestion earlier than standard TCP using a reliable mechanism.

• NF-TCP contributes very little to the queueing at bottlenecks.

• NF-TCP is able to use spare bandwidth efficiently by using proECN. The available bandwidth estimation scheme, proECN, is able to estimate available bandwidth in a reasonable time frame that makes it suitable for use by the NF-TCP protocol.

• NF-TCP outperforms other candidate approaches in a wide range of network scenarios.

• Furthermore, this research shows that even when the number of NF-TCP flows increases, its impact on standard TCP is negligible.

This thesis underlines that NF-TCP is viable and practical as an efficient network friendly protocol for delay-insensitive applications. Development of new transport protocols are necessary to meet the requirements of modern day and future applications, whilst ensuring network resources are efficiently utilized. These transport protocols need to be able to optimize the feedback they receive from the network to further better their performance and reduce their impact on the network. Additionally, the notion of flow fairness should be ceased and a premium be placed on data transfer during congested periods in the networks. This thesis touches upon such a solution, but a lot of work needs to be done. Some of the open issues are:

• Available Bandwidth Estimation: This thesis proposes proECN, which uses a combination of ECN markings and delay measurements to measure available bandwidth. This approach can be further improved to make it more reliable by relying completely on network feedback.

• Congestion Control Framework: The congestion control framework needs to be expanded to fit most of the existing transport layer protocols.

• NF-TCP deployment: NF-TCP must be deployed in a real network to further study its benefits and improve it.

• This thesis implements the modified RED queue in a Linux based router. It would be beneficial to implement and deploy it in an actual router hardware to study its impact.

Bibliography

[1] J. Babiarz, K. Chan, and F. Baker. Configuration Guidelines for Diff-Serv Service Classes. RFC 4594, August 2006.

[2] S. Shalunov and G. Hazel. Low Extra Delay Background Transport (LEDBAT). Internet-Draft draft-ietf-ledbat-congestion-00.txt, Internet Engineering Task Force, July 2010. Work in progress.

[3] Vinay J. Ribeiro, Rudolf H. Riedi, Richard G. Baraniuk, Jiri Navratil, and Les Cottrell. PathChirp: Efficient Available Bandwidth Estimation for Network Paths. In *Proc. Passive and Active Measurement Workshop*, 2003.

[4] M. Alizadeh, A. Greenberg, D. Maltz, J. Padhye, P. Patel, B. Prabhakar, Sudipta Sengupta, and M. Sridharan. DCTCP: Efficient Packet Transport for the Commoditized Data Center. In *Proc. SIGCOMM*, 2010.

[5] Information Sciences Institute. Transmission control protocol. RFC 793, September 1981.

[6] Internet Study 2008/2009. http://www.ipoque.com/resources/internet-studies/internet-study-2008_2009.

[7] Murat Yuksel, K.K. Ramakrishnan, Shiv Kalyanaraman, Joseph D. Houle, and Rita Sadhvani. Class-of-Service in IP Backbones: Informing the Network Neutrality Debate. *Sigmetrics*, June 2008.

[8] Bob Briscoe. Flow Rate Fairness: Dismantling a Religion. *ACM SIGCOMM Computer Communication Review*, July 2007.

[9] K.K. Ramakrishnan, S. Floyd, and D. Black. The Addition of Explicit Congestion Notification (ECN) to IP. RFC 3168, September 2001.

[10] The network simulator – ns-2. http://www.isi.edu/nsnam/ns/.

[11] François Baccelli, Giovanna Carofiglio, and Serguei Foss. Proxy Caching in Split TCP: Dynamics, Stability and Tail Asymptotics. In *Proc. INFOCOM*, 2008.

[12] J. Seedorf and E. Burger. Application-Layer Traffic Optimization (ALTO) Problem Statement. RFC 5693, October 2009.

[13] A. Kuzmanovic and E.W. Knightly. TCP-LP: a distributed algorithm for low priority data transfer. In *Proc. INFOCOM*, 2003.

[14] V. Konda and J. Kaur. RAPID: Shrinking the Congestion-Control Timescale. In *Proc. INFOCOM*, 2009.

[15] M. Allman, V. Paxson, and W. Stevens. Tcp congestion control. RFC 2581, April 1999.

[16] W. Stevens. Tcp slow start, congestion avoidance, fast retransmit, and fast recovery algorithms. RFC 2001, January 1997.

[17] R. Alimi, R. Penno, and Y. Yang. ALTO Protocol. Internet-Draft draft-ietf-alto-protocol-05, Internet Engineering Task Force, July 2010. Work in progress.

[18] S. Kiesel, S. Previdi, M. Stiemerling, R. Woundy, and R. Yang. Application-Layer Traffic Optimization (ALTO) Requirements. Internet-Draft draft-ietf-alto-reqs-05, Internet Engineering Task Force, June 2010. Work in progress.

[19] K. Yamada, R. Wang, M. Y. Sanadidi, and M. Gerla. Tcp westwood with agile probing: Dealing with dynamic, large, leaky pipes. In *Proc. ICC*, 2004.

[20] Lawrence S. Brakmo, Sean W. O'Malley, and Larry L. Peterson. TCP Vegas: New Techniques for Congestion Detection and Avoidance. In *Proc. SIGCOMM*, 1994.

[21] Lawrence S. Brakmo and Larry L. Peterson. Tcp vegas: End to end congestion avoidance on a global internet. *IEEE Journal on selected Areas in communications*, 13:1465–1480, 1995.

[22] A. Venkataramani, R. Kokku, and M. Dahlin. Tcp nice: a mechanism for background transfers. In *Proc. OSDI*, 2002.

[23] M. Welzl and D. Ros. A Survey of Lower-than-Best-Effort Transport Protocols. Internet-Draft draft-ietf-ledbat-survey-01, Internet Engineering Task Force, October 2010. Work in progress.

[24] Sally Floyd and Van Jacobson. Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Trans. on Netw.*, January 1993.

[25] G. McCullagh and D. Leith. Delay-based congestion control: Sampling and correlation issues revisited. Technical report, Hamilton Institute, 2008.

[26] R. Prasad, M. Jain, and C. Dovrolis. On the effectiveness of delay-based congestion avoidance. In *Proc. PFLDnet*, 2004.

[27] S. Rewaskar, J. Kaur, and D. Smith. Why don't delay-based congestion estimators work in the real-world? Technical report tr06-001, University of North Carolina at Chapel Hill, Dept. of Computer Science, 2006.

[28] Saad Biaz and Nitin H. Vaidya. Is the round-trip time correlated with the number of packets in flight? In *Proc. IMC*, 2003.

[29] R. Jain. A delay-based approach for congestion avoidance in interconnected heterogeneous computer networks. *SIGCOMM Comput. Commun. Rev.*, 1989.

[30] Kun Tan and Jingmin Song. A compound TCP approach for high-speed and long distance networks. In *Proc. INFOCOM*, 2006.

[31] J. Martin, A. Nilsson, and Injong Rhee. Delay-based congestion avoidance for TCP. *IEEE/ACM Transactions on Networking*, 2003.

[32] Richard J. La, Jean Walrand, and Venkat Anantharam. Issues in TCP Vegas. Technical Report, EECS Dept, UC Berkeley, 2001.

[33] Yong Xia, L. Subramanian, I. Stoica, and S. Kalyanaraman. One More Bit is Enough. *IEEE/ACM Transactions on Networking*, 2008.

[34] I.A. Qazi and T. Znati. On the Design of Load Factor based Congestion Control Protocols for Next-Generation Networks. In *Proc. INFOCOM*, 2008.

[35] I.A. Qazi, T. Znati, and L.L.H. Andrew. Congestion Control using Efficient Explicit Feedback. In *Proc. INFOCOM*, 2009.

[36] S. Floyd. HighSpeed TCP for Large Congestion Windows. RFC 3649, December 2003.

[37] Cheng Jin, D.X. Wei, and S.H. Low. FAST TCP: motivation, architecture, algorithms, performance. In *Proc. INFOCOM*, 2004.

[38] Sangtae Ha, Injong Rhee, and Lisong Xu. CUBIC: a new TCP-friendly high-speed TCP variant. *SIGOPS Oper. Syst. Rev.*, 2008.

[39] S. Floyd, M. Allman, A. Jain, and P. Sarolahti. Quick-Start for TCP and IP. RFC 4782, January 2007.

[40] Dina Katabi, Mark Handley, and Charlie Rohrs. Congestion control for high bandwidth-delay product networks. In *Proc. SIGCOMM*, 2002.

[41] Lawrence Stewart, Grenville Armitage, and Alana Huebner. Collateral damage: The impact of optimised tcp variants on real-time traffic latency in consumer broadband environments. In *Proc. Networking*, 2009.

[42] Mayutan Arumaithurai, Xiaoming Fu, and K.K. Ramakrishnan. LEDBAT architecture framework consisting of pluggable components. Internet-Draft draft-mayutan-ledbat-congestionarchitecture-00, Internet Engineering Task Force, 2010. Work in progress.

[43] Network friendly transport for delay-insensitive background traffic. http://www.net.informatik.uni-goettingen.de/research_projects/nft.

[44] P. Eardley. Pre-Congestion Notification (PCN) Architecture. RFC 5559 (Informational), June 2009.

[45] K. K. Ramakrishnan and Raj Jain. A binary feedback scheme for congestion avoidance in computer networks with a connectionless network layer. In *Proc. SIGCOMM*, 1988.

[46] Mayutan Arumaithurai, Xiaoming Fu, and K.K. Ramakrishnan. NF-TCP: A Network Friendly TCP Variant for Background Delay-Insensitive Applications. Technical Report No. IFI-TB-2010-05 draft-mayutan-ledbat-congestionarchitecture-00, Institute of Computer Science, University of Göttingen, Germany, 2010. ISSN 1611-1044.

[47] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. In *Proc. SIGCOMM*, 1989.

[48] J. Aweya, M. Ouellette, D. Y. Montuno, and A. Chapman. A Control Theoretic Approach to Active Queue Management. In *Computer Communications*, 2001.

[49] T. J. Ott, T. V. Lakshman, and L. H. Wong. SRED: Stabilized RED. In *Proc INFOCOMM*, 1999.

[50] W. Feng, D. Kandlur, D. Saha, and K. Shin. Blue: A New Class of Active Queue Management Algorithms. In *Technical report, UM CSE-TR-387-99*, 1999.

[51] C. Hollot, V. Misra, D. Towsley, and W. Gong. On Designing Improved Controllers for AQM Routers Supporting TCP Flows. In *Proc INFOCOMM*, 2001.

[52] S. Athuraliya, V. H. Li, S. H. Low, and Q. Yin. REM: Active Queue Management. In *Proc. IEEE Network*, 2001.

[53] Shao Liu, Tamer Basar, and R. Srikant. Exponential RED: A Stabilizing AQM Scheme for Low- and High-speed TCP Protocols. In *Proc. IEEE/ACM Transactions on Networking*, 2005.

[54] S. Kunniyur and R. Srikant. A time-scale decomposition approach to adaptive ECN marking. In *Proc INFOCOMM*, 2001.

[55] S. Kunniyur and R. Srikant. Analysis and Design of an Adaptive Virtual Queue Algorithm for Active Queue Management. In *Proc SIGCOMM*, 2001.

[56] C. Bastian, J. Livingood, J. Mills, and R. Woundy. Comcast's Protocol-Agnostic Congestion Management System. Internet-Draft draft-livingood-woundy-congestion-mgmt-05, Internet Engineering Task Force, July 2010. Work in progress.

[57] David X. Wei and P. Cao. NS-2 TCP-Linux: an NS-2 TCP implementation with congestion control algorithms from Linux. In *Proc. WNS2*, 2006.

[58] A Linux TCP implementation for NS2. http://netlab.caltech.edu/projects/ns2tcplinux/ns2linux/index.html.

[59] M. Christiansen, K. Jeffay, D. Ott, and F.D. Smith. Tuning RED for Web Traffic. In *Proc. ACM SIGCOMM*, 2001.

[60] L. Le, J. Aikat, K. Jeffay, and F.D. Smith. The Effects of Active Queue Management on Web Performance. In *Proc. ACM SIGCOMM*, 2003.

[61] L. Le, J. Aikat, K. Jeffay, and F.D. Smith. The Effects of Active Queue Management and Explicit Congestion Notification on Web Performance. In *Proc. IEEE/ACM Transactions on Networking*, 2005.

[62] C. Hollot, V. Misra, D. Towsley, and W. Gong. A Control Theoretic Analysis of RED. In *Proc. INFOCOMM*, 2001.

[63] Dario Rossi et al. News from the Internet congestion control world. *CoRR*, abs/0908.0812, 2009.

[64] T. Moncaster, B. Briscoe, and M. Menth. Baseline Encoding and Transport of Pre-Congestion Information. RFC 5696 (Standards Track), November 2009.

[65] P. Eardley. Metering and Marking Behaviour of PCN-Nodes. RFC 5670 (Informational), November 2009.

[66] G. Karagiannis, T. Taylor, K. Chan, and M. Menth. Requirements for Signaling of (Pre-) Congestion Information in a DiffServ Domain. Internet-Draft draft-ietf-pcn-signaling-requirements-01, Internet Engineering Task Force, October 2010. Work in progress.

[67] M. Arumaithurai, R. Geib, R. Rex, and X. Fu. An encoding method to signal 3 states with a single PCN bit. Student workshop, INFOCOM, April 2009.

[68] Mayutan Arumaithurai, Ruediger Geib, Rene Rex, and Xiaoming Fu. Pre-Congestion Notification based Flow Management in MPLS-based DiffServ Networks. In *Proc. The 28th IEEE International Performance Computing and Communications Conference (IPCCC 2009)*, 2009.

[69] B. Briscoe and T. Moncaster. PCN 3-State Encoding Extension in a single DSCP. Internet-Draft draft-ietf-pcn-3-in-1-encoding-00, Internet Engineering Task Force, July 2009. Work in progress.

[70] Bob Briscoe, Arnaud Jacquet, Toby Moncaster, and Alan Smith. Re-ECN: Adding accountability for causing congestion to TCP/IP, September 2009. Work in progress.

[71] W. Almesberger, E. Ica, Jamal H. Salim, A. Kuznetsov, and I. Moscow. Differentiated Services On Linux. In *Proc. GLOBECOM*, 1999.

[72] L. Krug, M. Menth, J. Araujo, S. Blake, and R. Woundy. The Need for Congestion Exposure in the Internet. Internet-Draft draft-moncaster-conex-problem-00, Internet Engineering Task Force, March 2010. Work in progress.

[73] H. Balakrishnan and S. Seshan. The Congestion Manager. RFC 3124, September 2001.

M.Sc. Mayutan Arumaithurai
Address: Bonhöfferweg 2
37075 - Göttingen
Germany
Phone: (+49) 17620322049
E-mail: mayutan.arumaithurai@gmail.com

My research interests include Network Layer, Transport Layer, Congestion Control, QoS, Emergency Services, Social Networks, Cloud computing.

Personal

Date of Birth:	18th March, 1981
Marital status:	Single
Nationality:	Srilankan
Languages known:	English, German (level III), Tamil, Hindi.

Education

April, 2007 - November, 2010 — **Ph.D, Computer Science**
University of Göttingen, Göttingen and Nokia Siemens Networks, Munich, Germany.

PhD supervisors: Prof. Dr. Xiaoming Fu, University of Göttingen.
Prof. Dr. Dieter Hogrefe, University of Göttingen.
Mentor and supervisor at Nokia Siemens Network: Hannes Tschofenig, NSN, Helsinki.
Thesis Mentor: K.K. Ramakrishnan, AT&T Labs-Research, U.S.A.

October, 2003 - November, 2006 — **Master of Science, Information and Communication Systems**
Technical University Hamburg Harburg (TUHH), Hamburg, Germany.

Master Thesis: Performance Analysis of the NSIS QoS-NSLP Protocol Suite using OMNET++.
Supervisors: Prof. Dr. Ulrich Killat and Prof. Dr. Dieter Gollmann, TUHH, Germany.
Mentors: Hannes Tschofenig, Siemens and Dr. Xiaoming Fu, University of Göttingen.

August, 1999 - April, 2003 — **Bachelor of Engineering, Computer Science**
Sathyabama Engineering College, Madras University, Chennai, India.

Bachelor Thesis: Congestion Control in ATM Networks.
1st class distinction

Professional Experience

March, 2007 - Till date — **Research Consultant, Nokia Siemens Networks (NSN), Munich, Germany**
Emergency Services:
- Implementation: https://sourceforge.net/projects/heldandlost/files/
- Support standardization efforts
- Development on Nokia N800

Standardization activities:
- IETF Diameter (DIME) WG
- IETF ECRIT WG

March 2007 - till date	**Research Fellow, University of Göttingen, Göttingen, Germany** Research: Congestion Control, QoS, Routing Protocols Teaching: Practical courses and Networking Lab Thesis Supervision: Bachelor thesis Project acquisition support Create and maintain the PCS homepage: http://www.uni-goettingen.de/en/138989.html
November 2005 - March 2007	**Work student, Siemens CT-IC3, Munich, Germany** Reviewed, modified the Diameter AAA QoS Application draft and simulated it on Open Diameter Contributed to the WiMAX draft for Pre-provisioning using RADIUS Implemented and evaluated NSIS and QoS-NSLP on OMNET++ Quality of Service Attributes and Parameters for Diameter AAA
December 2004 - May 2005	**Team technical coordinator and programmer, Sistech Software Gmbh, Italy** Galileo Avionica Space military project: Assisted in creating Galileo Avionica's very own NSU (Navigational Sensor Unit) Architectural design of subsystems and components, and also played an active role in the phases of Unit testing and Integration testing Headed a team of three Engineers and testers onsite in Italy Management between between the onsite team in Italy and Sistech's offshore group in India Languages and tools used: Ada, C, Adamulti and CCC Harvest
January 2004 - December 2004	**Student co-worker, Communication Networks Department TUHH, Germany** Implemented and evaluated Gnutella 0.4v P2P protocol in Ns-2 Implemented and evaluated MPLS/GMPLS in Ptolemy

Research Visits

September 2010 - October 2010	DAAD sponsored visit, Henning Schulzrinne, Department of Computer Science and Department of Electrical Engineering, Columbia University, New York.
November 2009	DAAD sponsored visit, Henning Schulzrinne, Department of Computer Science and Department of Electrical Engineering, Columbia University, New York.

Professional Activities

Dec 2009 onwards	**Student Speaker,** GAUSS Ph.D-Programme in Computer Science (PCS), University of Göttingen, Germany.
Jan 2007 onwards	**Reviewed Papers for the following conferences:** INFOCOM 2011, P2PNet 2010, ICCCN 2010, ICNP 2010, GLOBECOM 2010, Computer Communications 2009, GLOBECOM 2009, ICNP 2009, ICCCN 2009, ICNP 2008, INFOCOM 2009, GLOBECOM 2008, ICNP 2008, WoWMoM 2009, ON-MOVE 2007.

Teaching Experience

2009 onwards	Practical Course Networking Lab
2007, 2008	Praktikum-Telematik, Teaching Assistant, University of Göttingen.

List of Publications

August 2010	NF-TCP: A Network Friendly TCP Variant for Background Delay-Insensitive Applications, Mayutan Arumaithurai, Xiaoming Fu, K. K. Ramakrishnan, Technical Report No. IFI-TB-2010-05, Institute of Computer Science, University of Göttingen, Göttingen, Germany, ISSN 1611-1044.
August 2010	Network Friendly TCP (Extended Abstract) , Mayutan Arumaithurai, Fabian Glaser, Xiaoming Fu, K. K. Ramakrishnan, EuroView2010: Visions of Future Generation Networks, Joint ITG, ITC, and Euro-NF Workshop , 10th Würzburg Workshop on IP, Co-located with Official G-Lab Status Meeting August 2nd - August 3rd 2010 Würzburg, Germany.
May 2010	How secure is the next generation of IP-based emergency services architecture?, Hannes Tschofenig, Mayutan Arumaithurai, Henning Schulzrinne, Bernard Aboba, International Journal of Critical Infrastructure Protection, Volume 3, Issue 1, pages 41-50, Elsevier, doi:10.1016/j.ijcip.2010.02.001.
May 2010	NF-TCP: Network Friendly TCP, Mayutan Arumaithurai, Xiaoming Fu, K. K. Ramakrishnan, 17th IEEE Workshop on Local and Metropolitan Area Networks (LANMAN 2010), Long Branch, New Jersey, USA, IEEE.
March 2010	LEDBAT architecture framework consisting of pluggable components, Mayutan Arumaithurai, Xiaoming Fu, K. K. Ramakrishnan, IETF, draft-mayutan-ledbat-congestionarchitecture-00.txt.
February 2010	Traffic Classification and Quality of Service (QoS) Attributes for Diameter, Jouni Korhonen, Hannes Tschofenig, Mayutan Arumaithurai, Mark Jones, and Avi Lior, Request For Comment (RFC) 5777 (Proposed Standard), Internet Engineering Task Force, ISSN 2070-1721.
December 2009	Pre-Congestion Notification based Flow Management in MPLS-based DiffServ Networks, Mayutan Arumaithurai, Ruediger Geib, Rene Rex, and Xiaoming Fu, The 28th IEEE International Performance Computing and Communications Conference (IPCCC 2009), Phoenix, AZ, USA, IEEE.
April 2009	An encoding method to signal 3 states with a single PCN bit, Mayutan Arumaithurai, Ruediger Geib, Rene Rex, and Xiaoming Fu, IEEE INFOCOM 2009 Student Workshop.
December 2008	Performance Study of the NSIS QoS-NSLP Protocol, Mayutan Arumaithurai, Xiaoming Fu, Bernd Schloer, and Hannes Tschofenig, The 51th Annual IEEE Global Telecommunications Conference (GLOBECOM 2008), Next Generation Networks, Protocols, and Services Symposium, New Orleans, LA, USA, IEEE.
October 2008	Decoupling Congestion Control Using Traffic Aggregates and Middleboxes, Niklas Neumann, Ralf Lübben, Mayutan Arumaithurai, and Xiaoming Fu, IEEE International Conference on Network Protocols (ICNP 2008), poster session, Orlando, FL, USA.
September 2007	NSIS PCN-QoSM: A Quality of Service Model for Pre-Congestion Notification (PCN), Mayutan Arumaithurai, IETF draft, Work in Progress.

March 2007 - February 2010	Quality of Service Attributes for Diameter, draft-korhonen-dime-qos-attributes-(version 0 to 15), J.Korhonen, H. Tschofenig, M. Arumaithurai, M. Jones, A. Lior.

Thesis Supervised

October 2010	Bachelor Thesis: Simulation and evaluation of a network friendly congestion control algorithm, Fabian Glaser, Bachelor's thesis, No. ZFI-BSC-2010-09, Center of Computational Sciences, University of Göttingen, Germany, ISSN 1612-6793.
October 2008	Bachelor Thesis: Implementation and Analysis of HTTP Enabled Location Delivery and Location-to-Service Translation Clients for IP-based Emergency Calls, Benedikt Schäffler, Bachelor's thesis, No.ZFI-BSc-2008-04, Center of Computational Sciences, University of Göttingen, Germany, ISSN 1612-6793.

Patents

September 2009	(WO/2009/115507) AAA Based Location Retrieval, Mayutan Arumaithurai (DE) and Hannes Tschofenig (FI).
March 2009	(WO/2009/033954) Location Information Delivery In Telecommunication Networks, Mayutan Arumaithurai (DE) and Hannes Tschofenig (DE).

Minor Contributions/Acknowledgements received

October 2010 onwards	A Survey of Lower-than-Best-Effort Transport Protocols, draft-ietf-ledbat-survey, M. Welzl, D. Ros.
October 2009 onwards	Congestion Exposure Problem Statement, draft-tschofenig-conex-ps, H. Tschofenig, A. Cooper.
August 2009	RFC 5624: Quality of Service Parameters for Usage with Diameter, J.Korhonen, H. Tschofenig, E. Davies.
September 2008	Signaling 3 PCN states with baseline encoding, draft-geib-baseline-encoding-3state-00, R. Geib.
November 2008 onwards	Synchronizing Location-to-Service Translation (LoST) Protocol based Service Boundaries and Mapping Elements, H. Schulzrinne, H. Tschofenig.
February 2007 onwards	Anti-SPIT : A Document Format for Expressing Anti-SPIT Authorization Policies, draft-tschofenig-sipping-spit-policy, H. Tschofenig, D. Wing, H. Schulzrinne, T. Froment, G. Dawirs.