PERFORMANCE MODELING OF COMMUNICATION NETWORKS OFFERED WITH A MIXTURE OF PERSISTENT TCP AND UDP FLOWS

A THESIS

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND

ELECTRONICS ENGINEERING

AND THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Gökhan Çalış

July, 2013

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Nail Akar(Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Sinan Gezici

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Şenan Ece Güran Schmidt

Approved for the Graduate School of Engineering and Science:

Prof. Dr. Levent Onural
Director of the Graduate School

ABSTRACT

PERFORMANCE MODELING OF COMMUNICATION NETWORKS OFFERED WITH A MIXTURE OF PERSISTENT TCP AND UDP FLOWS

Gökhan Çalış
M.S. in Electrical and Electronics Engineering
Supervisor: Assoc. Prof. Dr. Nail Akar
July, 2013

A damped fixed-point approximation is proposed to characterize the throughput of persistent TCP and UDP flows in a network of router links supporting per-class queuing and Deficit Round Robin (DRR) scheduling among classes. In particular, we study the case of two classes where one of the classes uses drop-tail queue management and is intended for UDP traffic. The other class targeting TCP traffic is assumed to use Active Queue Management (AQM). The effectiveness of the proposed analysis method in this scenario is validated by extensive NS3 simulations. Moreover, we study, using the proposed fixed-point algorithm, the potential gain of employing the many-to-one communications paradigm with respect to the conventional one-to-one communications in which a client downloads a specific content from one server only.

Keywords: TCP, UDP, AQM, DRR, Per-class queuing.

ÖZET

SÜREKLİ VE KARMA TCP VE UDP AKIŞLARI İLE BESLENEN İLETİŞİM AĞLARININ PERFORMANS MODELLEMESİ

Gökhan Çalış
Elektrik Elektronik Mühendisliği, Yüksek Lisans
Tez Yöneticisi: Assoc. Prof. Dr. Nail Akar
Temmuz, 2013

Sürekli TCP ve UDP akışları içeren, sınıf başına kuyruklamayı destekleyen ve sınıflar arasında ise Açık Çevrimsel Sıralı (DRR) çizelgeleme yapan ağları karakterize etmek için sönümlü sabit nokta iterasyonları önerilmektedir. Özellikle iki sınıf içeren durumlar çalışıldı. Bu sınıflardan birinde UDP trafiği için kuyruktan düşürme kuyruk yönetimi öngörüldü. TCP trafiğine yönelik diğer sınıf için ise Aktif Kuyruk Yönetimi (AQM) kullanıldı. Önerilen analiz metodunun verimliliğini doğrulamak için geniş çaplı NS3 simulasyonları yapıldı. Ayrıca, önerilen sabit noktalı algoritma kullanılarak çoktan bire iletişim tekniğinin daha konvansiyonel bire bir iletişim tekniğine göre kazancı değişik senaryolarda irdelendi.

Anahtar sözcükler: TCP, UDP, AQM, DRR, Sınıf başına kuyruklama.

Acknowledgement

I would like to thank Assoc. Prof. Dr. Nail Akar for his guidance and his support throughout this study. It was great experience and pleasure for me to work with him. Also, I would like to thank Assoc. Prof. Dr. Sinan Gezici and Assoc. Prof. Dr. Şenan Ece Güran Schmidt for agreeing to serve on my committee. In addition, I would like to thank my friends and my parents for their support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In today's Internet, a substantial amount of traffic using Simple Mail Transfer Protocol (SMTP), File Transfer Protocol (FTP) and Hypertext Transfer Protocol (HTTP) is carried at the transport layer by Transmission Control Protocol (TCP). The TCP along with User Datagram Protocol (UDP) are the most dominant protocols of today's Internet transport layer. Using the simplest drop-tail queue management mechanism on links carrying TCP traffic results in the so-called "full queues" and "lock-out" problems which are discussed in [1]. The first full queues problem can be described as the buffer being occupied entirely most of the time which leads to large queuing delays and reduced TCP throughput. On the other hand, the lock-out problem refers to a case in which a single or a few flows dominate the queue space while other flows using the same link starve because of synchronization or other timing effects. In order to mitigate the full queues problem, Active Queue Management (AQM) techniques can be used which drop packets without waiting for the queue to be full [1]. The AQM drop decision is generally probabilistic on certain queue parameters to mitigate the lock-out problem [1]. In the literature, there are several AQM techniques proposed such as Random Early Detection (RED) [2],[3], Early Random Drop (ERD) [4], Random Exponential Marking (REM) [5].

In the literature, different analytical expressions have been proposed for characterizing the throughput of TCP's congestion control mechanism as a function

of the packet loss and round trip delay [6],[7],[8],[9],[10]. In this study, we use the model proposed in [9] which captures not only the fast retransmit mechanism of TCP Reno but also the effect of the time-out mechanism. The other TCP models proposed in [6],[7],[8] ignore certain features of TCP and consequently over-estimate TCP throughput while proposing simpler expressions. Using fixed-point iterations with the analytical expression for TCP flows' sending rates, one can study the behaviour of a TCP flow in a network of AQM routers offered with persistent TCP flows [11].

Increasing use of real time voice and video applications leads to more and more UDP traffic in today's Internet. Since UDP does not have a congestion control mechanism as TCP has, TCP flows sharing the same link with UDP flows may starve because of UDP flows' non-responsive behaviour. Obviously, this is not desirable for applications using TCP. To mitigate the TCP starvation problem, "fair queuing" may be used. Fair queuing is a set of techniques that allows each flow or group of flows passing through a network device to have a fair share of network resources; Weighted Round Robin (WRR) and Deficit Round Robin (DRR) are specific examples to fair queuing [12],[13]. DRR scheduling is a fair queuing technique proposed by [13] and DRR is shown to achieve near-perfect fairness in terms of throughput while requiring less computational work to process a packet when compared with other mechanisms. In this study, we use per-class queuing and DRR scheduling in router links where TCP flows and UDP flows may choose to join different traffic classes to tackle the TCP starvation problem. The main idea in this study is to isolate UDP and TCP traffic (using DRR scheduling) so that TCP flows do not starve because of other UDP flows. Also, RED is assumed to be used as the AQM mechanism for the queues serving TCP flows and drop-tail is assumed to be used as the queue-management mechanism for the queues serving UDP flows only. Hence, we can analyze TCP and UDP throughput in different scenarios. The fixed-point algorithm has been proposed to analyze networks offered with TCP traffic only [11]. In the current study, we improve this fixed-point algorithm to analyze networks offered with both TCP and UDP flows in a network using per-class queuing and DRR scheduling among the classes. We focus on the particular case of two classes, namely classes 1 and 2,

where TCP flows are only allowed to join Class 2 whereas UDP flows either join Class 1 or Class 2. With this setting, it is possible to analyze networks with and without TCP/UDP traffic isolation. Moreover, we validate the proposed analysis method with NS3 simulations in a number of scenarios.

Another topic we investigate in this thesis is the many-to-one communications paradigm. The problem can be described as follows. A client is assumed to require a specific content which resides at multiple servers. The well-known one-to-one communications paradigm allows the client to choose one of the many servers and setup a connection with this server to retrieve the content. In the many-to-one communications paradigm, a server sets up a multipoint-to-point connection between the client and a group of servers to download the content. In this manner, content is downloaded concurrently from multiple servers. We assume that a TCP friendly congestion control mechanism is to be used between the client and each server in the group. The server placement problem in a network is studied in [14] and it is shown that distributing content can be beneficial in terms of traffic engineering. Although we do not place servers optimally, we distribute them over the network. We would like to use the fixed-point approximation algorithm to study the following questions:

1. What is the benefit of using multiple servers in terms of client throughput?
2. What is the number of servers to connect so as to achieve a reasonable gain in client throughput?
3. How are the servers to be selected?

In this thesis, we perform a study to answer these questions using the proposed fixed-point algorithm as the analysis instrument. In the scenarios we studied, we have shown that it is possible to attain an % 19 gain in client throughput when using two servers as opposed to one single server if the servers are selected randomly from the set of all servers hosting the content.

Our contributions in this thesis are as follows:

1. We introduce damped fixed point iterations for studying networks offered with TCP persistent traffic only. We note that damping was not mentioned in the original work [11].

2. We extend the work in [11] by studying networks offered with both TCP and UDP flows. We also allow per-class queuing and DRR scheduling in network links for TCP/UDP isolation.

3. We study, using the proposed fixed-point algorithm, the potential gain of employing the many-to-one communications paradigm with respect to the conventional one-to-one communication paradigm. This study needs to be viewed as a proof of concept for many-to-one communications rather than a complete suite of protocols to implement.

The thesis is organized as follows. Chapter 1 addresses the introduction to the problems studied in the thesis. In Chapter 2, we present the fixed-point iterations to obtain per-flow throughputs in networks offered with persistent TCP and UDP flows. In Chapter 3, the proposed method is validated by NS3 simulations. Chapter 4 studies the gain of employing the many-to-one communication paradigm. Finally, we conclude.

Chapter 2

TCP Analysis

2.1 Single Congested Router

Before going into network of nodes, let's start with one link case which is analyzed in [11]. Assume TCP flows share one congested link, which implements AQM. Call the link v with transmission capacity C_v bits per second and buffer size of B_v bits. There is a probability function $p_v(x_v)$ for this link v, whose argument x_v is the queue length of link v. One of the popular AQM mechanism is Random Early Detection (RED) and one of the recommended variant of RED [15] is given in (2.1):

$$p_v(x_v) = \begin{cases} 0, & 0 \leq x_v \leq t_v^{min} \\ \frac{x_v - t_v^{min}}{t_v^{max} - t_v^{min}} p_v^{max}, & t_v^{min} \leq x_v \leq t_v^{max} \\ p_v^{max} + \frac{x_v - t_v^{max}}{t_v^{max}}(1 - p_v^{max}), & t_v^{max} \leq x_v \leq 2t_v^{max} \\ 1, & 2t_v^{max} \leq x_v \leq B_v \end{cases} \tag{2.1}$$

where t_v^{min},t_v^{max} and p_v^{max} are configurable RED parameters.

Now, assume we have a network of such links in which all links implement RED as described in (2.1) as their AQM mechanism. Moreover, we use per-class queuing and DRR scheduling for the TCP and UDP traffic classes. This scenario

is of main interest to us where the goal is to isolate TCP and UDP flows in a complex network using DRR scheduling. Also, since RED is used for all links as AQM in TCP queues, using such probability functions for all links in a network, one can find TCP flows' end-to-end loss rates and with the help of the TCP flows' propagation delays, which can be calculated easily, one can derive TCP flows' sending rates and characterize the throughput analysis of such a network which includes both TCP and UDP flows. The detailed analysis for this general case is discussed next.

2.2 Network Topology Analysis with Deficit Round-Robin Scheduling and Active Queue Management

Let us assume that the flows traversing a set of routers are persistent in the network. Consider a link v, in a network with total of V links, with transmission capacity C_v bits per second and uses DRR scheduling with 2 classes. Hence, we have 2 different queues, $Q_v^{(1)}$ and $Q_v^{(2)}$, on each link v and capacities can be assigned as desired. Call the weights for each queue as $w_v^{(1)}$ and $1 - w_v^{(1)}$ respectively, denote the current given capacities as $C_v^{(1)}$ and $C_v^{(2)}$. On $Q_v^{(1)}$, simple queue management, tail drop, is used since we do not place TCP flows on $Q_v^{(1)}$, call the probability drop function of tail-drop queue management as $p_v^{(1)}(x_v^{(1)})$, where $x_v^{(1)}$ is the current queue length of $Q_v^{(1)}$. However, on $Q_v^{(2)}$, both TCP and UDP flows can be placed. Therefore, RED is used with probability drop function $p_v^{(2)}(x_v^{(2)})$, where $x_v^{(2)}$ is the current queue length of $Q_v^{(2)}$. Also, buffer sizes for $Q_v^{(1)}$ and $Q_v^{(2)}$ are denoted by $B_v^{(1)}$ and $B_v^{(2)}$ on each link v, respectively. Then, we can modify (2.1) into (2.3) for our purposes:

$$p_v^{(1)}(x_v^{(1)}) = \begin{cases} 0, & 0 \leq x_v^{(1)} \leq B_v^{(1)} \\ 1, & otherwise \end{cases} \tag{2.2}$$

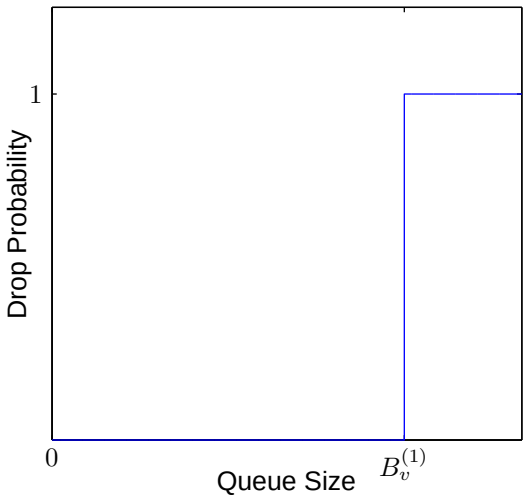

Figure 2.1: Drop Tail

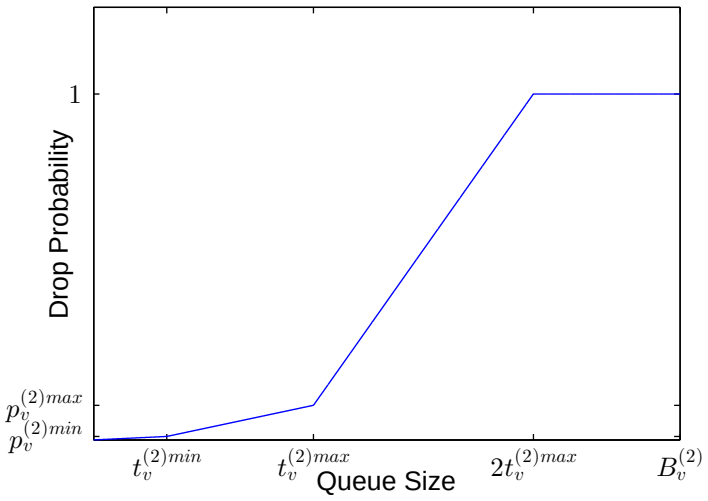

Figure 2.2: Altered RED

$$p_v^{(2)}(x_v^{(2)}) = \begin{cases} 0, & 0 \le x_v^{(2)} \le t_v^{(2)min} \\ \frac{x_v^{(2)} - t_v^{(2)min}}{t_v^{(2)max} - t_v^{(2)min}} p_v^{(2)max}, & t_v^{(2)min} \le x_v^{(2)} \le t_v^{(2)max} \\ p_v^{(2)max} + \frac{x_v^{(2)} - t_v^{(2)max}}{t_v^{(2)max}}(1 - p_v^{(2)max}), & t_v^{(2)max} \le x_v^{(2)} \le 2t_v^{(2)max} \\ 1, & 2t_v^{(2)max} \le x_v^{(2)} \le B_v^{(2)} \end{cases} \tag{2.3}$$

However, on MATLAB we need to have a one-to-one function in order to get convergence in the fixed-point algorithm therefore (2.1) is altered slightly, which is given in (2.4). Note that in MATLAB simulations, $p_v^{(2)min}$ will be set as small as possible in order to make the altered RED function look like (2.1), since NS3 simulations will be using (2.1).

$$p_v^{(2)}(x_v^{(2)}) = \begin{cases} \frac{p_v^{(2)min}}{t_v^{(2)min}} x_v^{(2)}, & 0 \le x_v^{(2)} \le t_v^{(2)min} \\ p_v^{(2)min} + \frac{x_v^{(2)} - t_v^{(2)min}}{t_v^{(2)max} - t_v^{(2)min}}(p_v^{(2)max} - p_v^{(2)min}), & t_v^{(2)min} \le x_v^{(2)} \le t_v^{(2)max} \\ p_v^{(2)max} + \frac{x_v^{(2)} - t_v^{(2)max}}{t_v^{(2)max}}(1 - p_v^{(2)max}), & t_v^{(2)max} \le x_v^{(2)} \le 2t_v^{(2)max} \\ 1, & 2t_v^{(2)max} \le x_v^{(2)} \le B_v^{(2)} \end{cases} \tag{2.4}$$

For traffic demand, we assume there are $K_U^{(1)}$ UDP class 1 flows, labelled

as $i_u^{(1)}$=1,...,$K_U^{(1)}$, $K_U^{(2)}$ UDP and $K_T^{(2)}$ TCP flows for class 2 flows labelled as $i_U^{(2)}$=1,...,$K_U^{(2)}$ and $i_T^{(2)}$=1,...,$K_T^{(2)}$ respectively. Let $V_{i_T^{(2)}}$ be the set of links ordered as the route of flow $i_T^{(2)}$, which is found using Dijkstra's Shortest Path algorithm [16]. In order to express TCP sending rates, round trip time and end to end loss rates need to be calculated. Denoting the one-way propagation delay of a link v by RTT_v, the expected round trip time for flow $i_T^{(2)}$, denoted by $R_{i_T^{(2)}}$, can be expressed as:

$$R_{i_T^{(2)}} = 2 \sum_{v \in V_{i_T^{(2)}}} RTT_v + \sum_{v \in V_{i_T^{(2)}}} x_v^{(2)} / C_v^{(2)} \tag{2.5}$$

where the first term is the two-way propagation delay for TCP flow $i_T^{(2)}$ and the second term is the queuing delay experienced by the flow $i_T^{(2)}$ on the route. Let $q_{i_T^{(2)}}$ be the end-to-end loss probability of TCP flow $i_T^{(2)}$, which can be expressed as follows:

$$q_{i_T^{(2)}} = 1 - \prod_{v \in V_{i_T^{(2)}}} (1 - p_v^{(2)}). \tag{2.6}$$

Similarly, let $V_{i_U^{(1)}}$ and $V_{i_U^{(2)}}$ be the set of links ordered as the route of UDP flows $i_U^{(1)}$ and $i_U^{(2)}$ respectively, then end-to-end loss probabilities for the flows can be expressed as follows:

$$q_{i_U^{(1)}} = 1 - \prod_{v \in V_{i_U^{(1)}}} (1 - p_v^{(1)}), \tag{2.7}$$

$$q_{i_U^{(2)}} = 1 - \prod_{v \in V_{i_U^{(2)}}} (1 - p_v^{(2)}). \tag{2.8}$$

2.3 TCP Model

In the literature, there are several expressions for TCP sending rate. In our model, we are using the TCP model given in [9]. The model proposed by [9] analytically characterizes the TCP steady-state throughput as a function of the loss rate and round trip time. The model is shown to capture both TCP's fast retransmit mechanism and TCP's timeout mechanism [9].

The maximum congestion window size, W_{max}, is determined at the beginning of TCP flow establishment [9]. Hence, if there is no loss, the window size can grow up to W_{max}. The window limitation affects TCP rate based on the relationship between the window size and W_{max} resulting with 2 different expressions for TCP sending rate [9]. Before going further into TCP formula let's define couple of equations first; denote unconstrained window size by W_u the mean of which is given at (2.9) [9].

$$E[W_u, q_{i_T^{(2)}}] = \frac{2+b}{3b} + \sqrt{\frac{8(1-q_{i_T^{(2)}})}{3bq_{i_T^{(2)}}} + (\frac{2+b}{3b})^2} \tag{2.9}$$

where the parameter b is the number of packets acknowledged by a received ACK. Many TCP receivers send one cumulative ACK for two consecutive packets received therefore b is typically 2. Let's call the probability that loss in a window of size w is detected by time-outs as $Q(w, q_{i_T^{(2)}})$:

$$Q(w, q_{i_T^{(2)}}) = min\left(, \frac{(1-(1-q_{i_T^{(2)}})^3)(1+(1-q_{i_T^{(2)}})^3)(1-(1-q_{i_T^{(2)}})^{w-3}))}{1-(1-q_{i_T^{(2)}})^w}\right). \tag{2.10}$$

Using both (2.9) and (2.10), TCP sending rate can be characterized as:

$$T(R_{i_T^{(2)}}, q_{i_T^{(2)}}) = \begin{cases} \dfrac{\frac{1-q_{i_T^{(2)}}}{q_{i_T^{(2)}}} + E[W_u] + Q(E[W_u], q_{i_T^{(2)}}) \frac{1}{1-q_{i_T^{(2)}}}}{R_{i_T^{(2)}}(\frac{b}{2}E[W_u]+1) + Q(E[W_u], q_{i_T^{(2)}}) T_0 \frac{F(q_{i_T^{(2)}})}{1-q_{i_T^{(2)}}}}, & E[W_u] < W_{max} \\ \dfrac{\frac{1-q_{i_T^{(2)}}}{q_{i_T^{(2)}}} + W_{max} + Q(W_{max}, q_{i_T^{(2)}}) \frac{1}{1-q_{i_T^{(2)}}}}{R_{i_T^{(2)}}(\frac{b}{8}W_{max} + \frac{1-q_{i_T^{(2)}}}{q_{i_T^{(2)}} W_{max}} + 2) + Q(W_{max}, q_{i_T^{(2)}}) T_0 \frac{F(q_{i_T^{(2)}})}{1-q_{i_T^{(2)}}}} & otherwise \end{cases} \tag{2.11}$$

where

$$F(q_{i_T^{(2)}}) = 1 + q_{i_T^{(2)}} + 2q_{i_T^{(2)}}^2 + 4q_{i_T^{(2)}}^3 + 8q_{i_T^{(2)}}^4 + 16q_{i_T^{(2)}}^5 + 32q_{i_T^{(2)}}^6 \tag{2.12}$$

and T_0 denotes period of timeout.

Here, only the functions appearing in TCP sending rate formula are expressed but detailed derivation of these functions can be found in [9].

2.4 Matlab Algorithm

We propose the method of damped fixed-point iterations in the algorithm. When the classical method is used, the algorithm can oscillate between 2 points as a result it does not converge, in order to solve this issue damped fixed-point iteration method is performed. The difference between the two methods is that in damped, instead of using the current iteration value for the starting point of next iteration, moving average of current value and previous iteration's value is calculated and the result is taken as the starting point of next iteration. By using damped method in the algorithm, abrupt jumps between two successive iterations are prevented which helps to convergence. For the k^{th} iteration, moving average operation for the current queue length of link v is described in (2.13).

$$(x_v^{(2)})^{k+1} = \alpha(x_v^{(2)})^k + (1-\alpha)(x_v^{(2)})^{k-1}, 0 \leq \alpha \leq 1 \tag{2.13}$$

In order to make a decision about capacities, demand on each link must be calculated. To calculate UDP demands, first call the sending rates of UDP flows

as $T(q_{i_U^{(1)}})$ and $T(q_{i_U^{(2)}})$ that goes through $Q_v^{(1)}$ and $Q_v^{(2)}$ respectively. Recall that the sending rate of TCP flows can be calculated using (2.11). We also know the individual loss probabilities on each link therefore total demand on link v's queues, $D_v^{(1)}$ and $D_v^{(2)}$, can be calculated. Let $S_{i_U^{(1)}}$ be the set of UDP flows who traverse link v and goes into queue $Q_v^{(1)}$ and also let $Z_{v,i_U^{(1)}}$ be the set of links ordered as the route of the flow $i_U^{(1)}$ until it reaches the link v. Similarly, we have $S_{i_U^{(2)}}$ and $Z_{v,i_U^{(2)}}$ for UDP flow $i_U^{(2)}$; $S_{i_T^{(2)}}$ and $Z_{v,i_T^{(2)}}$ for TCP flow $i_T^{(2)}$. Then, $D_v^{(1)}$ and $D_v^{(2)}$ can be calculated as:

$$D_v^{(1)} = \sum_{i_U^{(1)} \in S_{i_U^{(1)}}} T(q_{i_U^{(1)}}) \prod_{v \in Z_{v,i_U^{(1)}}} (1 - p_v^{(1)}), \tag{2.14}$$

$$D_v^{(2)} = \sum_{i_U^{(2)} \in S_{i_U^{(2)}}} T(q_{i_U^{(2)}}) \prod_{v \in Z_{v,i_U^{(2)}}} (1 - p_v^{(2)}) + \sum_{i_T^{(2)} \in S_{i_T^{(2)}}} T(R_{i_T^{(2)}}, q_{i_T^{(2)}}) \prod_{v \in Z_{v,i_T^{(2)}}} (1 - p_v^{(2)}). \tag{2.15}$$

In order to run the fixed-point iterations, new values for the drop probabilities are needed and on each queue they can be calculated using demands and capacities easily as follows:

$$p_v^{(1)} = \begin{cases} 1 - \frac{C_v^{(1)}}{D_v^{(1)}} & C_v^{(1)} \leq D_v^{(1)} \\ 0 & otherwise \end{cases} \tag{2.16}$$

$$p_v^{(2)} = \begin{cases} 1 - \frac{C_v^{(2)}}{D_v^{(2)}} & C_v^{(2)} \leq D_v^{(2)} \\ 0 & otherwise \end{cases} \tag{2.17}$$

After the fixed-point algorithm converges, we have the demand and capacity values on each link. Hence, the decision about the capacities needs to be done. Recall that the first given weighted capacities always should be available to be used. If the flows cannot use the all capacity, only then the other queue can use the extra capacity. Hence, the first case is that if the demand on the link is lower

than the instantaneous capacity of that link, then lower the capacity to be equal to the demand. The lowered amount of the capacity is given to the other queue. The examples are given considering the demand and capacity of $Q_v^{(1)}$ of link v but the procedure is similar to apply the $Q_v^{(2)}$ of link v.

if $D_v^{(1)} \leq C_v^{(1)}$ **then**
  $C_v^{(2)} \longleftarrow C_v^{(2)} + (C_v^{(1)} - D_v^{(1)})$
  $C_v^{(1)} \longleftarrow C_v^{(1)} - (C_v^{(1)} - D_v^{(1)})$
end if

The first case is easy to understand but when the demand is higher than the instantaneous capacity, the decision needs to be done carefully such that other inequalities needs to be taken into account as well. It's important to compare the demand and the first given weighted capacity. Even though the demand may be higher than the instantaneous capacity, the queue cannot use more than its right, the first capacity assigned by the weights, unless the other queue decides to give the unused capacity of itself to it. Hence, the instantaneous capacity cannot go beyond the first assigned value unless the other queue gives some extra capacity to it.

if $D_v^{(1)} \geq C_v^{(1)}$ and $D_v^{(1)} \leq C_v w_v^{(1)}$ and $C_v^{(1)} \leq C_v w_v^{(1)}$ **then**
  $C_v^{(1)} \longleftarrow D_v^{(1)}$
  $C_v^{(2)} \longleftarrow C_v - C_v^{(1)}$
end if
if $D_v^{(1)} \geq C_v^{(1)}$ and $D_v^{(1)} \geq C_v w_v^{(1)}$ and $C_v^{(1)} \leq C_v w_v^{(1)}$ **then**
  $C_v^{(1)} \longleftarrow C_v w_v^{(1)}$
  $C_v^{(2)} \longleftarrow C_v(1 - w_v^{(1)})$
end if

There is one case that is not analyzed yet, which is when demand is higher than the instantaneous capacity and both demand and instantaneous capacity

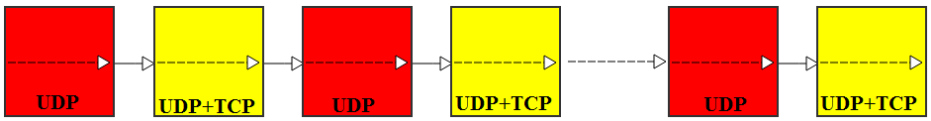

Figure 2.3: Nested Fixed-Point Algorithm

are higher than first weighted capacity. This case refers to the situation in which other queue cannot use it's right and gave some of unused capacity to other queue. In that cases, we do not change the assigned capacities. The assigned capacities can be changed if only the queue needs some of the capacity it gave earlier.

Until $Q_v^{(1)}$ capacity assignments are finalised for each link v, which require some number of iterations in fixed-point algorithm, we do not change $Q_v^{(2)}$ for any v. After the $Q_v^{(1)}$ are set, the same procedure can be applied for $Q_v^{(2)}$ which would yield different capacity assignments for both $Q_v^{(1)}$ and $Q_v^{(2)}$, therefore we would have to go back to $Q_v^{(1)}$ iterations and so on, which results in a nested fixed-point algorithm. At some point, we converge some values so that between 2 consecutive UDP and TCP blocks separately, the difference is so negligible, in that case we finish the iterations. Illustration of the whole process can be seen on 2.3.

Initialization;
$p_v^{(1)} \leftarrow 0$; $p_v^{(2)} \leftarrow 0$; $x_v^{(2)} \leftarrow 0$; $\forall v \in V$;
$C_v^{(1)} \leftarrow C_v w_v^{(1)}$; $C_v^{(2)} \leftarrow C_v(1 - w_v^{(1)})$; $\forall v \in V$;
while *The maximum throughput difference between 2 iterations is greater than* ϵ **do**
    Start with Class 1 first;
    while *The maximum throughput difference between 2 iterations of Class 1 is greater than* ζ **do**
        Calculate the individual end-to-end drop probabilities $q_{i_U^{(1)}}$ of flows from $p_v^{(1)}$ (2.7);
        Calculate the demand on each link v, $D_v^{(1)}$ (2.14);
        Calculate new $p_v^{(1)}$ for each link v according to $D_v^{(1)}$ and $C_v^{(1)}$ (2.16);
    end
    Make the decision on new capacities;
    if $D_v^{(1)} \leq C_v^{(1)}$ **then**
        $C_v^{(2)} \longleftarrow C_v^{(2)} + (C_v^{(1)} - D_v^{(1)})$;
        $C_v^{(1)} \longleftarrow C_v^{(1)} - (C_v^{(1)} - D_v^{(1)})$;
    else if $D_v^{(1)} \geq C_v^{(1)}$ *and* $D_v^{(1)} \leq C_v w_v^{(1)}$ *and* $C_v^{(1)} \leq C_v w_v^{(1)}$ **then**
        $C_v^{(1)} \longleftarrow D_v^{(1)}$;
        $C_v^{(2)} \longleftarrow C_v - C_v^{(1)}$;
    else if $D_v^{(1)} \geq C_v^{(1)}$ *and* $D_v^{(1)} \geq C_v w_v^{(1)}$ *and* $C_v^{(1)} \leq C_v w_v^{(1)}$ **then**
        $C_v^{(1)} \longleftarrow C_v w_v^{(1)}$;
        $C_v^{(2)} \longleftarrow C_v(1 - w_v^{(1)})$;
    end
    Continue with Class 2;
    while *The maximum throughput difference between 2 iterations of Class 2 is greater than* η **do**
        Calculate the individual end-to-end drop probabilities of flows $q_{i_U^{(2)}}$ and $q_{i_T^{(2)}}$ from $p_v^{(2)}$ (2.8, 2.6) ;
        Calculate new sending rates for TCP flows, $T(R_{i_T^{(2)}}, q_{i_T^{(2)}})$ and the demand on each link v, $D_v^{(2)}$ (2.11, 2.15) ;
        Calculate new $p_v^{(2)}$ for each link v according to $D_v^{(2)}$ and $C_v^{(2)}$ (2.17);
        Use dampening to adjust the values of $p_v^{(2)}$(2.13);
    end
    Make the decision on new capacities;
    if $D_v^{(2)} \leq C_v^{(2)}$ **then**
        $C_v^{(1)} \longleftarrow C_v^{(1)} + (C_v^{(2)} - D_v^{(2)})$;
        $C_v^{(2)} \longleftarrow C_v^{(2)} - (C_v^{(2)} - D_v^{(2)})$;
    else if $D_v^{(2)} \geq C_v^{(2)}$ *and* $D_v^{(2)} \leq C_v(1 - w_v^{(1)})$ *and* $C_v^{(2)} \leq C_v(1 - w_v^{(1)})$ **then**
        $C_v^{(2)} \longleftarrow D_v^{(2)}$;
        $C_v^{(1)} \longleftarrow C_v - C_v^{(2)}$;
    else if $D_v^{(2)} \geq C_v^{(2)}$ *and* $D_v^{(2)} \geq C_v(1 - w_v^{(1)})$ *and* $C_v^{(2)} \leq C_v(1 - w_v^{(1)})$ **then**
        $C_v^{(2)} \longleftarrow C_v(1 - w_v^{(1)})$;
        $C_v^{(1)} \longleftarrow C_v w_v^{(1)}$;
    end
end

Algorithm 1: Nested Fixed-point iterations

Chapter 3

Numerical Results

In order to understand the accuracy of the algorithm proposed, different network topologies are studied. Both simple topologies with few nodes and few flows and complex topologies with relatively more nodes and more flows are considered. In addition, the topologies are analyzed using not only MATLAB but also simulated using the NS3 simulator.

In all MATLAB analysis, we used the same RED parameters when AQM is performed. In terms of packets, we have $t_v^{(2)min}$=30, $t_v^{(2)max}$=90 and $B_v^{(2)}$=180 also probability values of $p_v^{(2)min}$=0.001 and $p_v^{(2)max}$=0.1; see Figure 2.2. The timeout period, T_0 in (2.11) is set to 0.2s. However, in NS3 simulations we have $p_v^{(2)min}$=0. The reason for adding slope in first interval for MATLAB simulations was discussed before. Also, in MATLAB analysis ϵ,ζ and η values used in Algorithm 1 are set ϵ=ζ=η=1bps. Finally, the α parameter used in the dampening phase of Algorithm 1 is set to 0.0003.

Since the default NS3 simulator does not include any source code related to our study, simulations are done based on the study by [17]. The study involves NS3 source codes which implements DRR scheduling with AQM queues. The codes presented there are altered in order to simulate our network topologies.

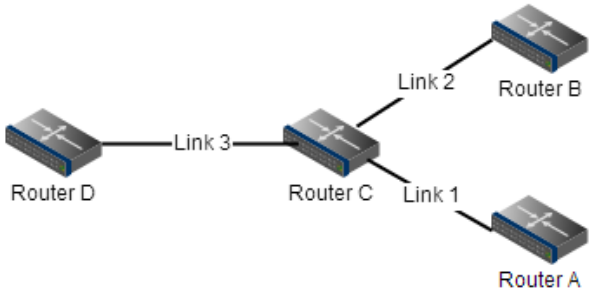

Figure 3.1: Y Network

To mention briefly, there are 2 types of nodes, edge and core, in NS3 simulations. Edge nodes color the flows for their type. It may be the port numbers, IP addresses of source or destination etc., but the idea is make some of flows have the same color. In our case, port numbers are used for coloring operation, UDP flows and TCP flows are separated according to their port numbers. Core nodes simply take the traffic from edge nodes in which flows are already colored, and uses DRR scheduling over these colored flows. RED policy is used as AQM in one of the queues and drop-tail is used in the other queue. There is also another queue which is called Expedited Forwarding (EF) queue which uses drop-tail queue management. The EF queue is the high priority queue while the queues used in DRR scheduling are low priority queues. In our simulations, we only put ACK packets to EF queue while TCP and UDP flows go through 2 different queues in low priority queues. Also note that, in NS3 simulations, Dijkstra's Shortest Path algorithm is used to find routes and the exact routes are taken into consideration for MATLAB simulations to have the exact scenario.

3.1 Y Network

Y network is a simple network with only 4 nodes and consist of 2 types of flows. Network topology can be seen on Figure 3.1. Although the network seems simple, it is not trivial to guess the behavior of the flows in a topology given below.

In this network, we have 5 identical TCP flows from Router A to D and

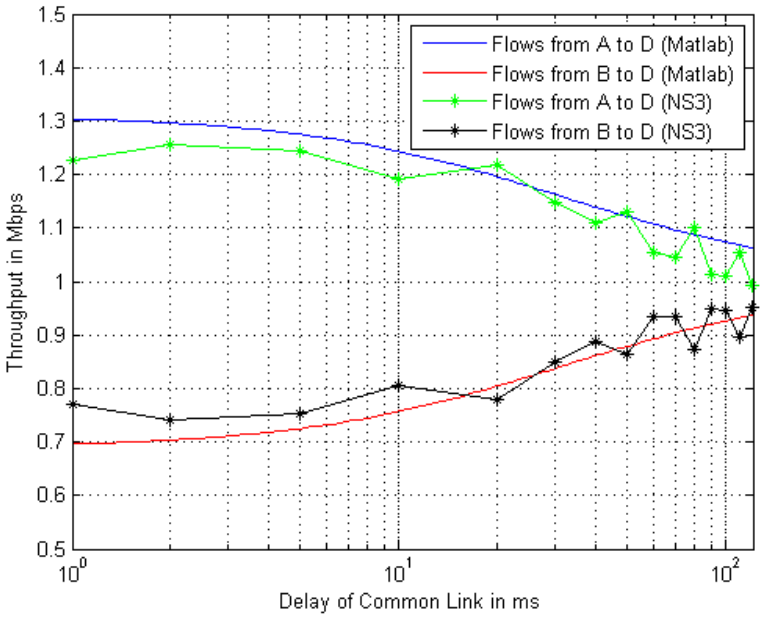

Figure 3.2: Y Network Results

another 5 TCP flows from Router B to D. However, the uncommon links have different capacities and propagation delays. Let's denote the capacity of Link 1 as C_1 and one way propagation delay of that link as RTT_1. Similarly we can define C_2, C_3, RTT_2 and RTT_3. By keeping Link 3 as the bottleneck, we can analyze the effect of RTT_3 in simulations. In the simulation settings, we have C_1=10 Mbps, C_2=100 Mbps, C_3=10 Mbps, RTT_1=1 ms and RTT_2=20 ms. The one-way propagation delay of the common link, RTT_3, varies between 1 ms and 120 ms. Hence, we can analyze the effect of having a link with high capacity but also with a long propagation delay versus a link with less capacity but with less propagation delay.

The results of both MATLAB and NS3 simulations can be seen on Figure 3.2. For NS3 simulations, 15 points are selected such that we have RTT_3 as 1 ms, 2 ms, 5 ms, 10 ms, 20 ms, 30 ms, 40 ms, 50 ms, 60 ms, 70 ms, 80 ms, 90 ms, 100 ms, 110 ms and 120 ms. For each point on the graph, 20 simulations are done with simulation length of 300 ms in NS3. In MATLAB, logarithmically spaced vector is generated for the delay values between 1 ms and 120 ms and the vector has 100 elements.

It can be observed that flows from A to D get more throughput in all cases despite coming from a link with less capacity. However, as the common link's delay increases, both types of flows experience more and more similar propagation delays which results in more and more closer throughput values for the two types of flows. The MATLAB analysis presents similar results with NS3 simulations and it captures the general behavior. We observe that if the topology is assumed to have some bottleneck link, then it may not worth to get links with higher capacities at other ends since the flows coming from there will eventually experience the bottleneck links and would have to drop lots of packets anyway. However, it is obvious that, if we had higher capacity on the common link such that it would not be bottleneck (e.g. C_3=150 Mbps), then flows from B to D would get much more throughput.

Overall, the Y Network indicates that TCP throughput may not be easy to estimate even in such simple networks. However, the topology does not include any DRR scheduling with UDP flows, which are needed to verify our analysis tool.

3.2 Simple Network

In this hypothetical network, which can be seen on Figure 3.3, we analyze a topology which includes both TCP and UDP flows. We have both AQM and DRR scheduling in this scenario as discussed before. However, for the sake of simplicity, all UDP flows goes through the same queue so we have full isolation between UDP and TCP flows. As discussed before, TCP flows go for Class 2 queues which implement RED as their AQM, while UDP flows join Class 1 queues which implement Drop-Tail queue management.

The network includes 1 TCP and 1 UDP flow from Router 1,2 and 3 to 10,11 and 12 respectively. All links in the topology have the same capacity and propagation delay such that C_v=10 Mbps for v=1,...,11 and RTT_v=2 ms for v =1,...,11. Since all links are identical, the weights to be used for DRR scheduling

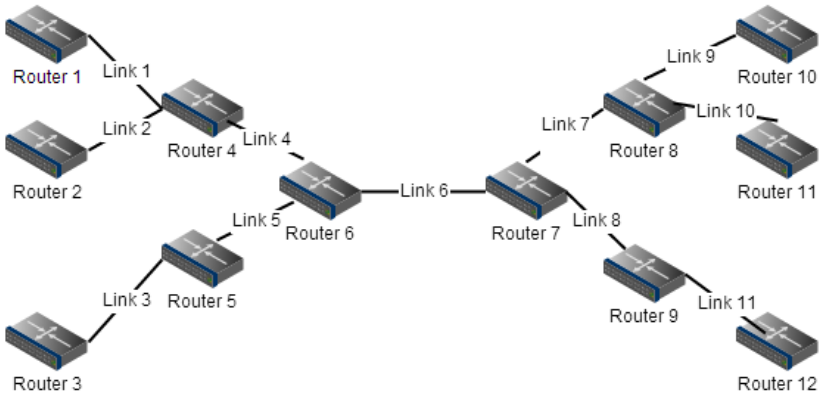

Figure 3.3: Simple Network

Flows vs. Tput in Mbps	$w_v^{(1)}$=0.25 (NS3)	$w_v^{(1)}$=0.25	$w_v^{(1)}$=0.50 (NS3)	$w_v^{(1)}$=0.50	$w_v^{(1)}$=0.75 (NS3)	$w_v^{(1)}$=0.75
TCP1-10	2,590	2,525	1,724	1,679	0,911	0,837
TCP2-11	2,596	2,525	1,762	1,679	0,869	0,837
TCP3-12	2,437	2,451	1,724	1,643	0,887	0,827
UDP1-10	0,455	0,494	1,050	1,107	1,726	1,784
UDP2-11	0,456	0,494	1,046	1,107	1,733	1,784
UDP3-12	1,434	1,511	2,667	2,788	3,850	3,932

Table 3.1: Simulation Results for Simple Network

are also same in each individual simulation. In total, 3 different simulations are performed in which $w_v^{(1)}$ is set to be 0.25, 0.50 and 0.75 respectively for all v. Recall that $w_v^{(1)}$ is the weight of a queue which accepts UDP flows. Also, all UDP sending rates are set to 10 Mbps so that both queues can be full and there is no capacity transfer between the queues. Finally, NS3 simulations are done 5 times each having length of 500 ms. The results for both NS3 and MATLAB simulations are presented in Table 3.1 for which NS3 simulation results are indicated by "(NS3)" while others are MATLAB results. The exact data on Table 3.1 can be seen on Figure 3.4 for better comparison and understanding of difference between NS3 and MATLAB simulations.

It can be observed from both Table 3.1 and Figure 3.4 that the analysis tool we have on MATLAB produces reasonable results compared to NS3 simulation results. There are slight differences but it was expected since we cannot have %100 similarity between real simulations and analysis results. However, still the network is not complex and we need more flows in a bigger topology to verify our analysis tool.

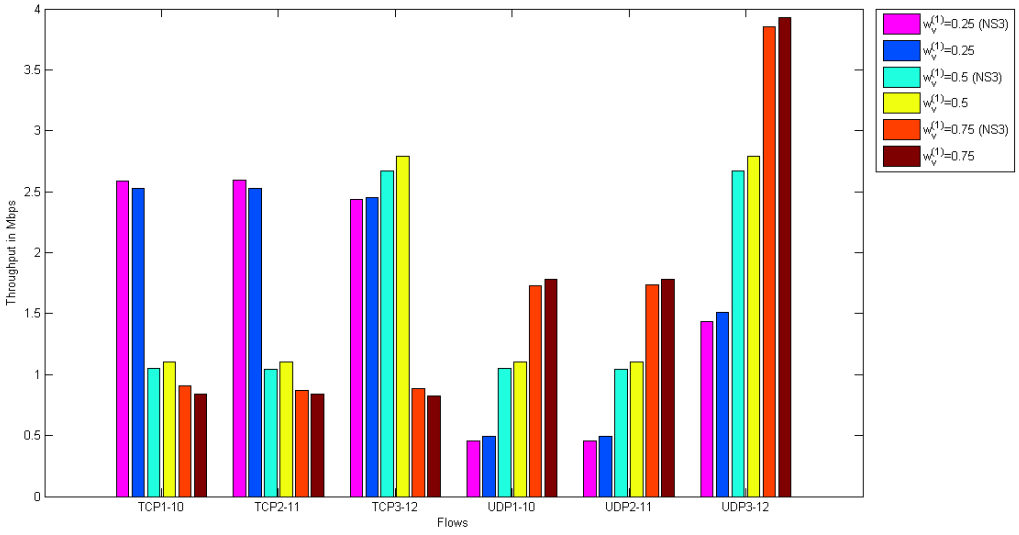

Figure 3.4: Simple Network Results on Bar Graph

3.3 US Network

US network, based on a former NSF network topology which has been used in many studies that have been published, is used as our final topology to verify our analysis tool. The US network consists of 14 nodes with 21 links and because it is a real network, some links are short while others are long which results in a different propagation delay for each link. Physical distances between nodes, which are given in [18], are used to find propagation delays and can be seen on Figure 3.6. Detailed information about US Network can be found in [18],[19].

The capacities of the links are chosen to be identical in this scenario, C_v=10 Mbps, v=1,2,...,21. Furthermore, we have 20 TCP and 20 UDP flows in the simulation. For simplicity, all UDP flows' rates are fixed to a constant and equal to 3 Mbps. The sources and destinations are chosen randomly but it is considered to have a scenario where we have wide range of TCP and UDP throughput values. All flows' sources and destinations can be seen on Table 3.2.

In total, 3 different simulations are performed in which $w_v^{(1)}$ is set to be 0.25, 0.50 and 0.75 respectively $\forall v$. For NS3 simulations, all cases are simulated 2 times with one simulation having length of 1000 ms. Histograms of TCP and

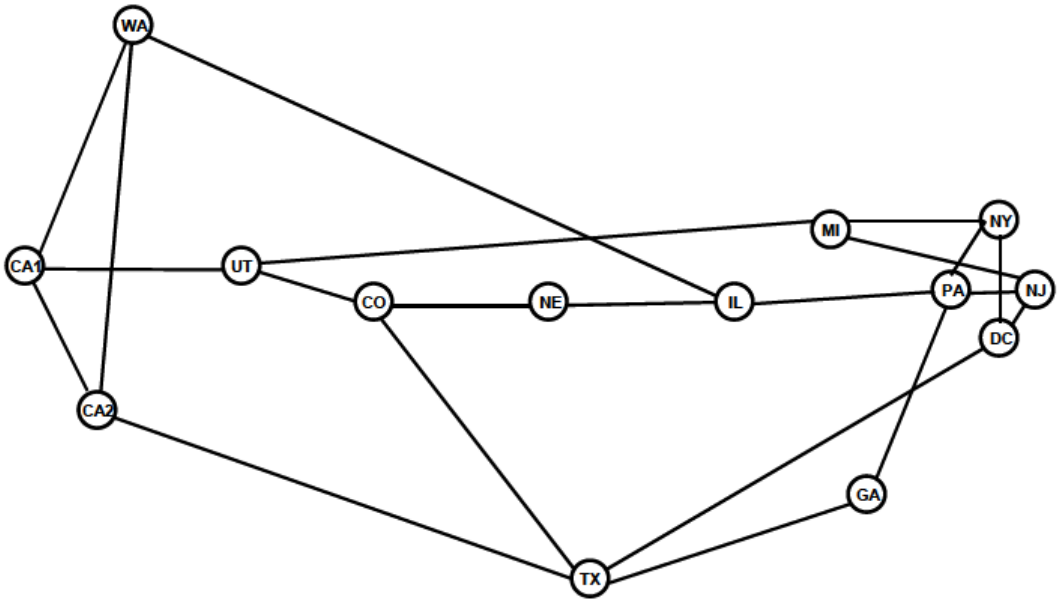

Figure 3.5: US Network

Physical Distances														
	CA1	CA2	CO	DC	GA	IL	MI	NE	NJ	NY	PA	TX	UT	WA
CA1														
CA2	834													
CO														
DC														
GA														
IL														
MI														
NE			870			864								
NJ				312			942							
NY				468			720							
PA					1008	846			540	438				
TX		2520	1746	2364	1350									
UT	1152		684				2820							
WA	1338	2056				3408								
Fibre Length [km]														

Figure 3.6: US Network Physical Distances

Table 3.2: All Flows

Table 3.3: TCP Flows

Flow	Source	Destination
TCP1	CO	DC
TCP2	TX	CA1
TCP3	MI	TX
TCP4	UT	TX
TCP5	IL	WA
TCP6	UT	CO
TCP7	NJ	WA
TCP8	WA	CA2
TCP9	PA	MI
TCP10	PA	NE
TCP11	GA	CO
TCP12	MI	CA2
TCP13	PA	CA2
TCP14	TX	IL
TCP15	MI	CA1
TCP16	NY	MI
TCP17	NE	CO
TCP18	PA	TX
TCP19	NJ	GA
TCP20	CO	NY

Table 3.4: UDP Flows

Flow	Source	Destination
UDP1	DC	GA
UDP2	CA2	CO
UDP3	NE	UT
UDP4	NJ	CA2
UDP5	UT	CA2
UDP6	UT	NE
UDP7	CO	DC
UDP8	NE	CA2
UDP9	DC	NY
UDP10	NE	CA1
UDP11	UT	TX
UDP12	IL	UT
UDP13	IL	PA
UDP14	UT	CA1
UDP15	TX	GA
UDP16	CA1	UT
UDP17	NJ	WA
UDP18	NY	MI
UDP19	NE	IL
UDP20	UT	NE

UDP throughput values can be seen on Figure 3.7, 3.8, 3.9, 3.10, 3.11, 3.12. Also, no-isolation case results are presented on Figure 3.13, 3.14 in which class 1 and class 2 flows goes into the same RED implemented queues.

Generally, our analysis tool capture the behaviour of the network. There are some differences but it was expected from such a large network. Some of the differences in the results are due to the fact that in MATLAB analysis, we did not consider the amount of traffic generated by ACK packets but in NS3 simulation, ACK traffic is generated and it uses some of the capacity in the opposite way of the original flow. Although ACK traffic may not seem significant, as number of TCP flows increases so does the ACK flows which results in increased ACK traffic. In NS3 for every 2 packets (2x1000 bytes) received, 1 ACK packet (40 bytes) is send so ACK traffic is % 2 of the traffic received and MATLAB code could be modified accordingly.

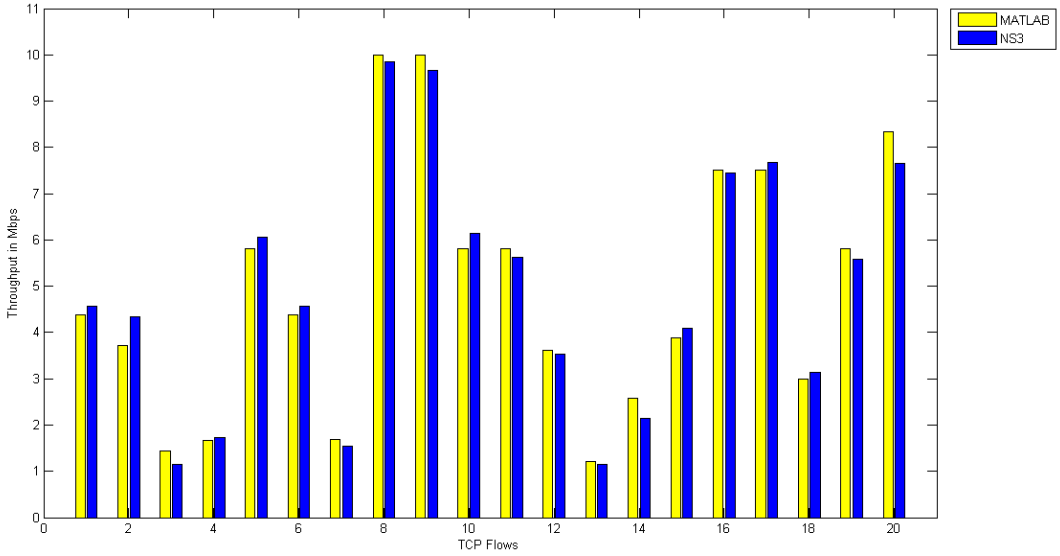

Figure 3.7: TCP Flows when $w_v^{(1)}$=0.25 $\forall v$

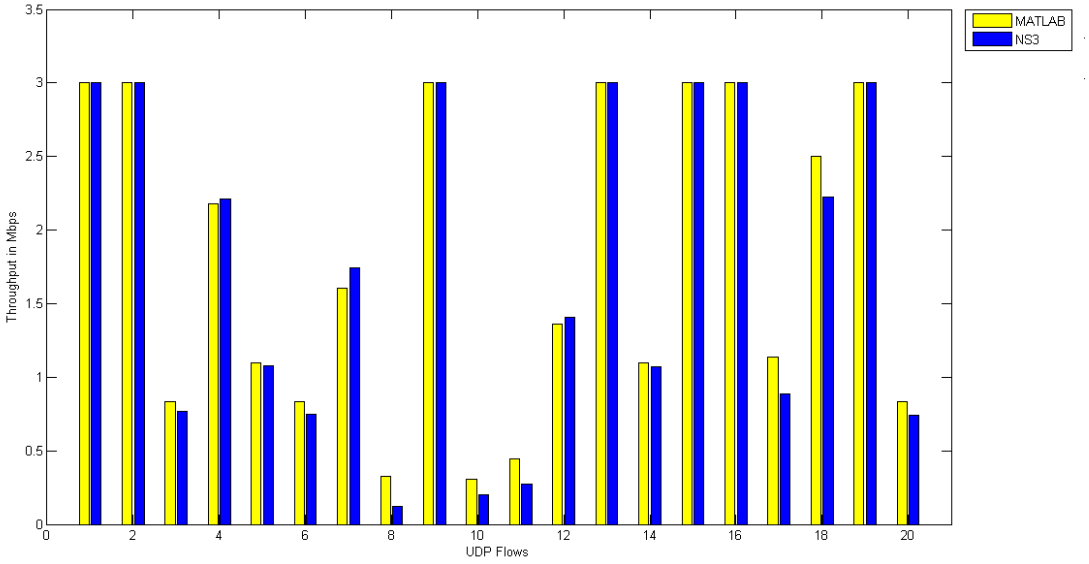

Figure 3.8: UDP Flows when $w_v^{(1)}$=0.25 $\forall v$

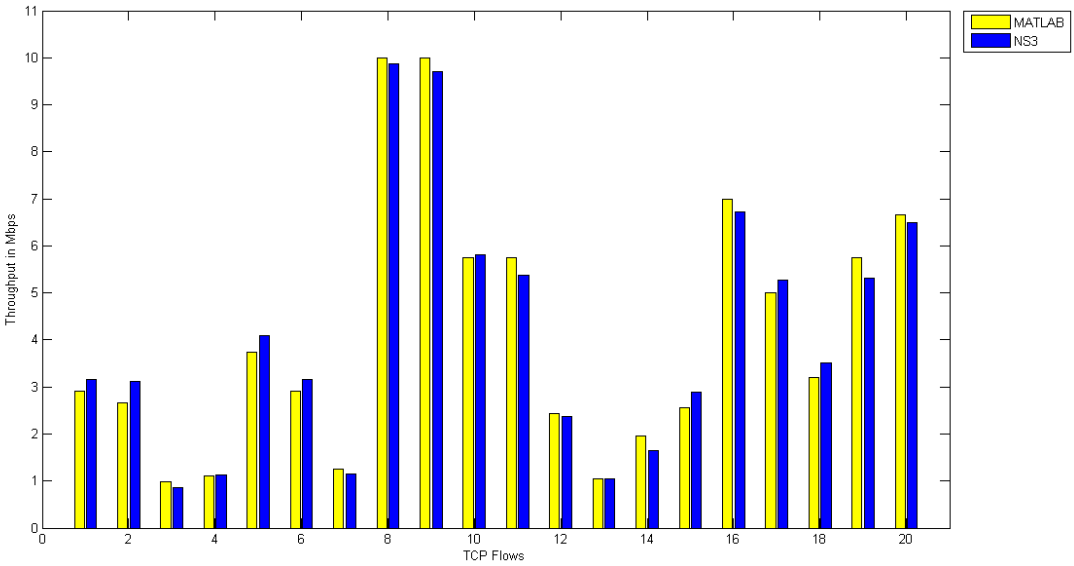

Figure 3.9: TCP Flows when $w_v^{(1)}$=0.50 $\forall v$

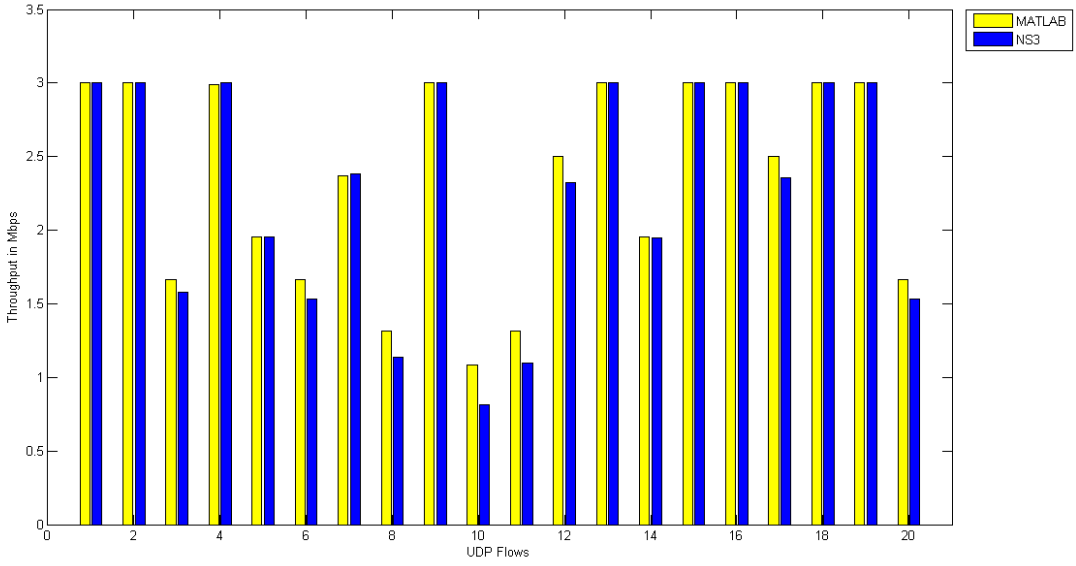

Figure 3.10: UDP Flows when $w_v^{(1)}$=0.50 $\forall v$

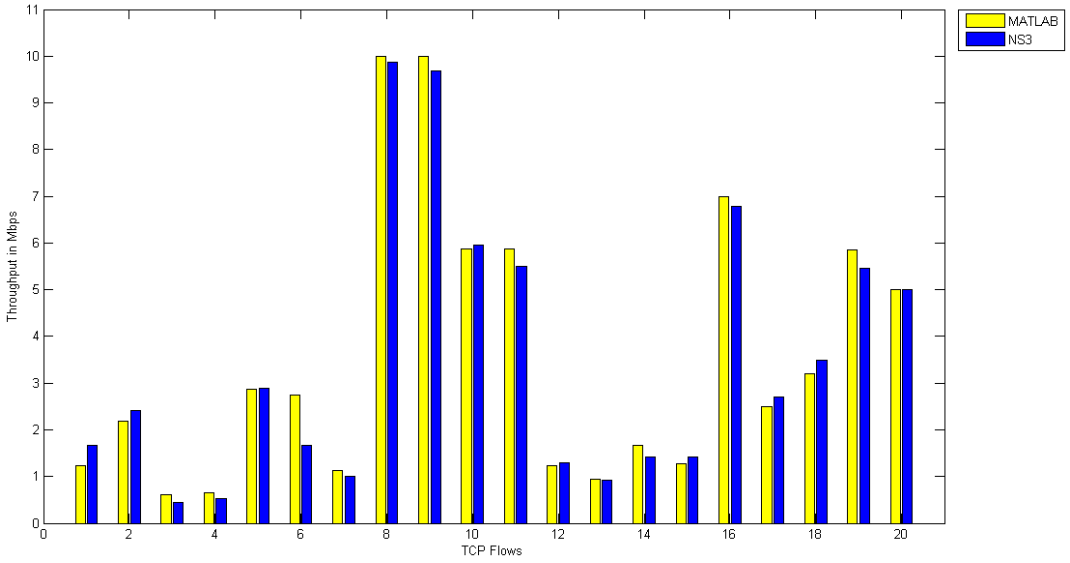

Figure 3.11: TCP Flows when $w_v^{(1)}$=0.75 $\forall v$

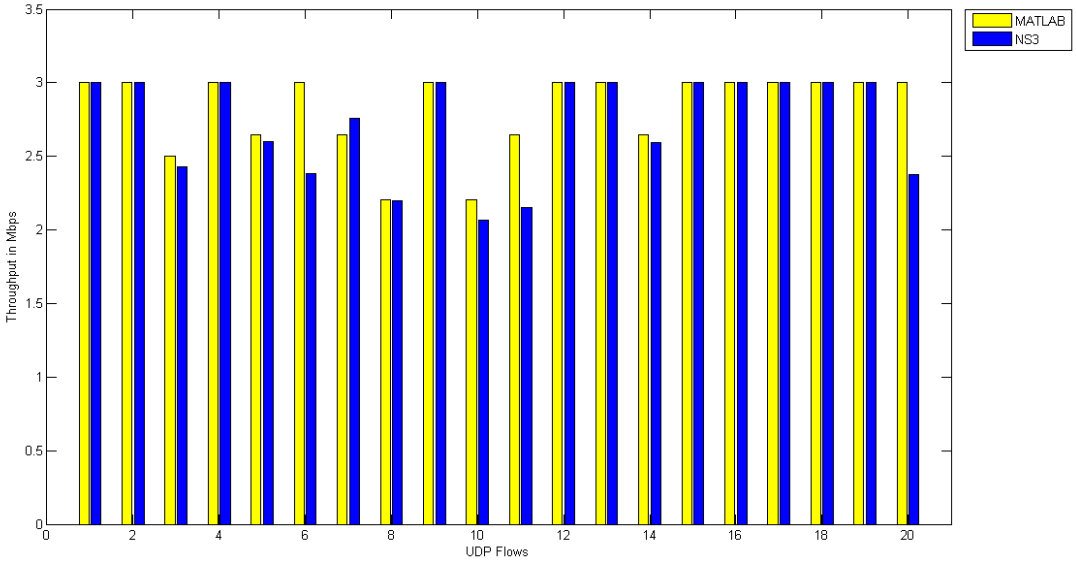

Figure 3.12: UDP Flows when $w_v^{(1)}$=0.75 $\forall v$

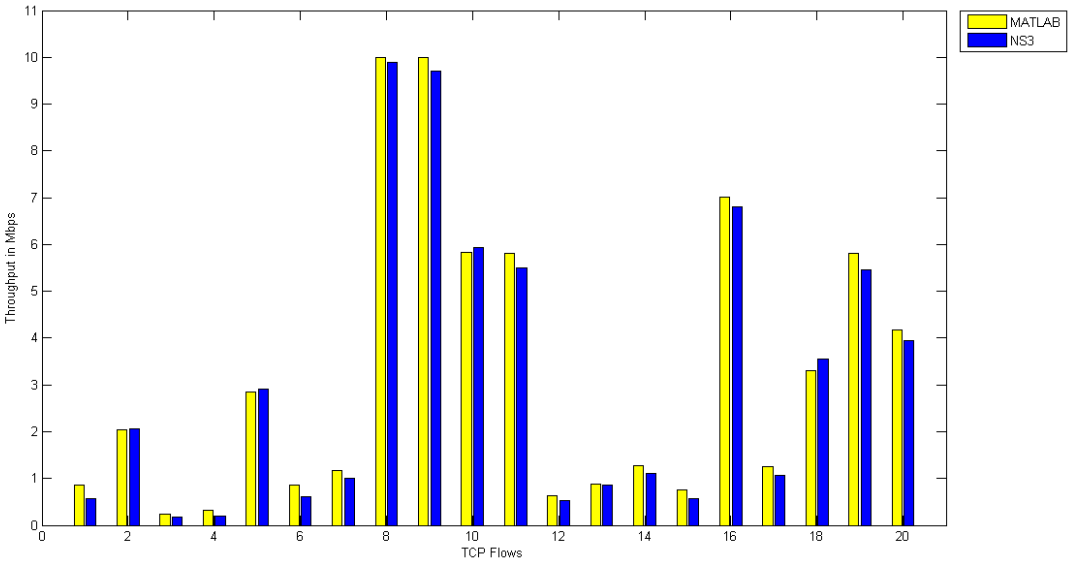

Figure 3.13: TCP Flows when no-isolation is employed

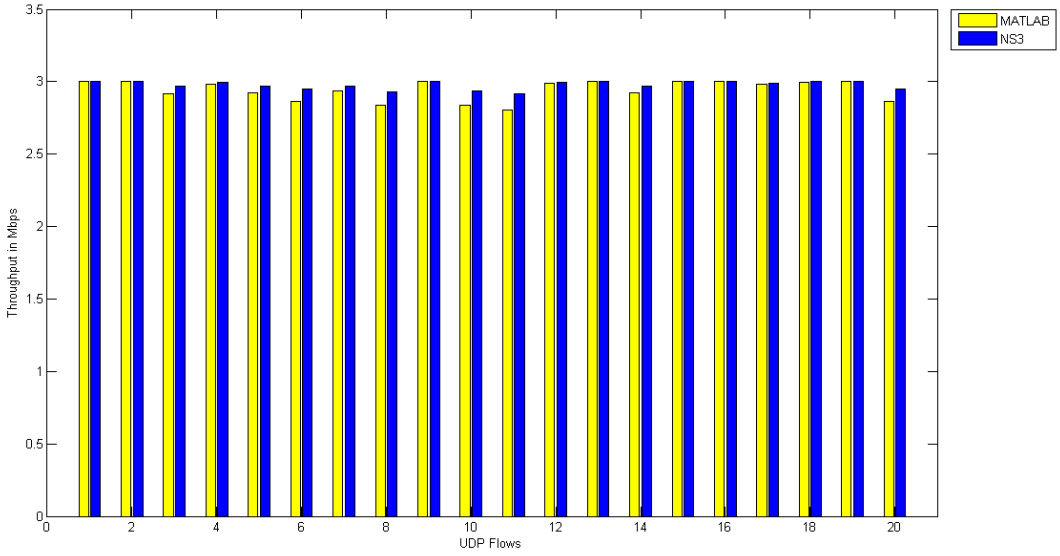

Figure 3.14: UDP Flows when no-isolation is employed

Chapter 4

Many-to-one Communication

Assume we have a network in which n different contents are available and each content can be placed in any k_n servers, we have m_n clients for content n and these clients can connect any of k_n servers to download content n. We create different scenarios by randomizing k_n, m_n and location of each server and client for each simulation while keeping n constant. In the network, one client may want to download more than one content and one server can contain more than one content. Same as before, we have C_v capacity for link v and one way propagation delay of RTT_v.

We study different policies to analyze many-to-one communication.

1. Randomly choose 1 server for each client.
2. Randomly choose 2 servers for each client.
3. Randomly choose 3 servers for each client.
4. Choose the server closest to each client.
5. Choose 2 servers closest to each client.

In policies 4 and 5, by closest we mean fewest number of hops between client and server and if there are more than one server which has the same cost to the

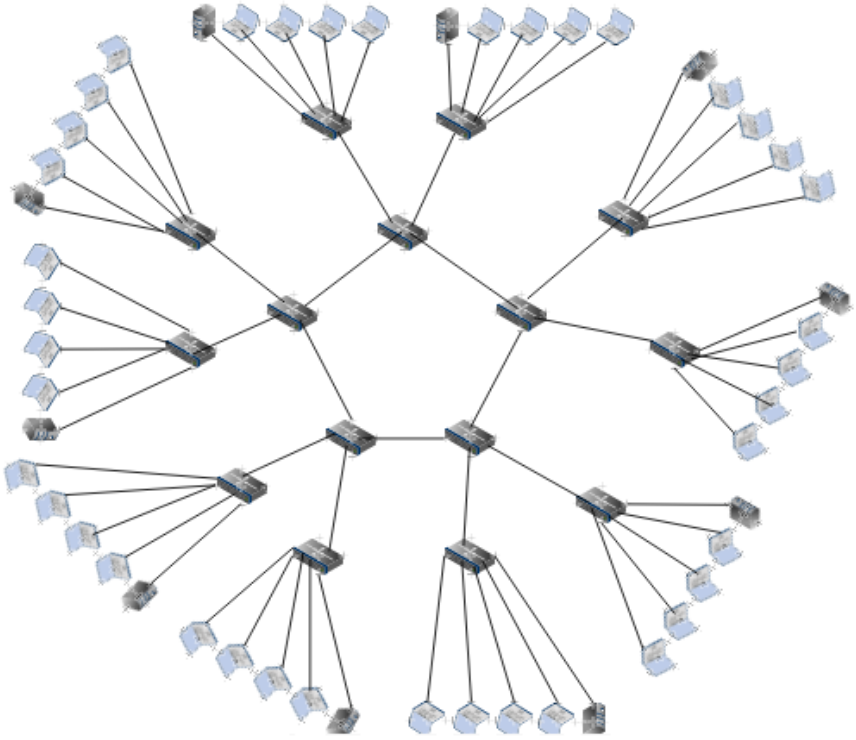

Figure 4.1: Ring Topology

client, then we randomly choose any of such servers to serve the client. In the following scenarios, it is assumed that, if the client connects more than one server, the total throughput it gets from the servers is sum of the individual throughput values. However, in real life this may not be easy and specific coding (e.g. Raptor Codes [20]) may be required so that the client would not download the same parts of the data from the servers.

4.1 Ring Topology

Assume we have a ring network which has 5 core nodes with 2 edge nodes connected to each core. Also there are 1 potential server and 4 potential clients connected to each edge node. The topology can be seen on Figure 4.1. Moreover, in this network we have following properties:

Policies	Mean Throughput in Mbps	Coefficient of Variation
Random1	0.29433	0.6372
Random2	0.35040	0.5287
Random3	0.37736	0.5260
Minimum1	0.32772	0.7750
Minimum2	0.38587	0.5103

Table 4.1: Simulation Results for Ring Network

1. n=3
2. k_n = U(4,10)
3. m_n = U(1,14)
4. C_v = 1 Mbps, v=1,2,...,65
5. RTT_v = 1 ms, v=1,2,...,65

The content n is placed at k_n servers out of 10 servers and we have m_n clients for that specific content, which can be placed at any other 40 spaces.

In total, 100 simulations are done in MATLAB with 2272 clients are generated for each policy. Mean throughput values of 2272 clients for each policy can be seen on Table 4.1 along with the coefficient of variation of each policy, which is the ratio of the standard deviation to the mean. Moreover, histogram of each policy can be observed on Figure 4.2, 4.3, 4.4, 4.5, 4.6. As we can see from the results, connecting more servers significantly improves the total throughput client gets. In random server cases, we have %19 gain between "Random1" and "Random2" and %8 gain between "Random2" and "Random3". Furthermore, similar results are acquired for minimum cost cases in which we have %17 gain between "Minimum1" and "Minimum2". Also note that by connecting more than one server, the fairness of the systems improves since coefficient of variation decreases.

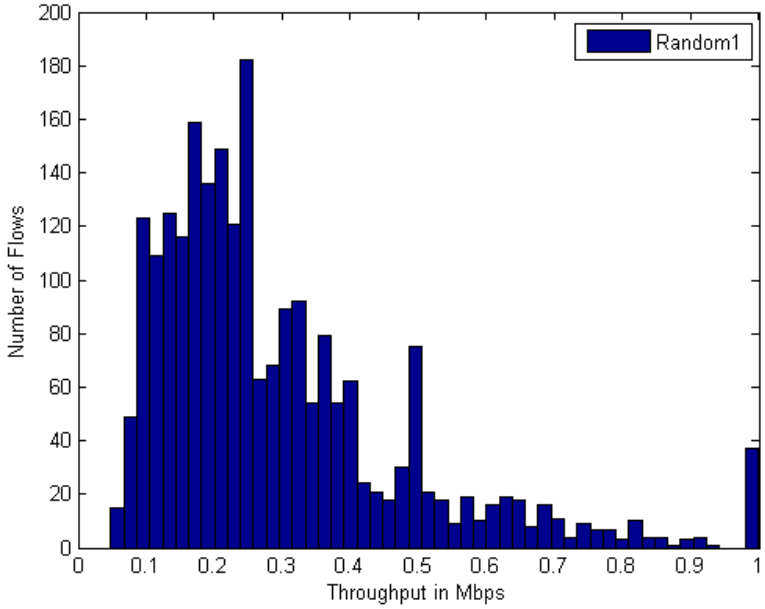

Figure 4.2: Random1 Policy Results for Ring Network

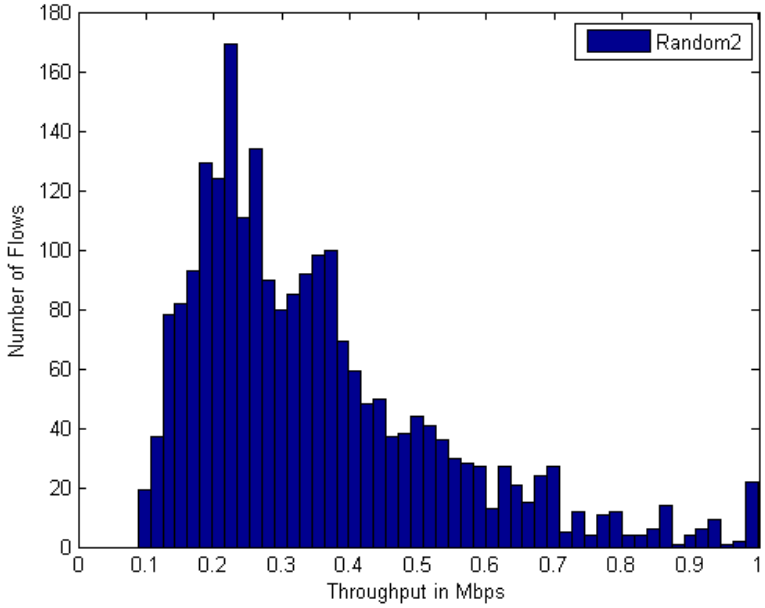

Figure 4.3: Random2 Policy Results for Ring Network

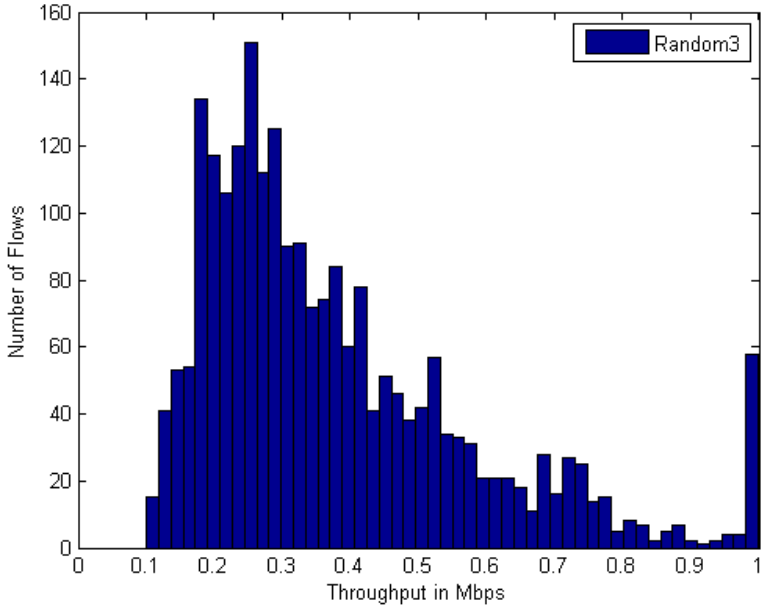

Figure 4.4: Random3 Policy Results for Ring Network

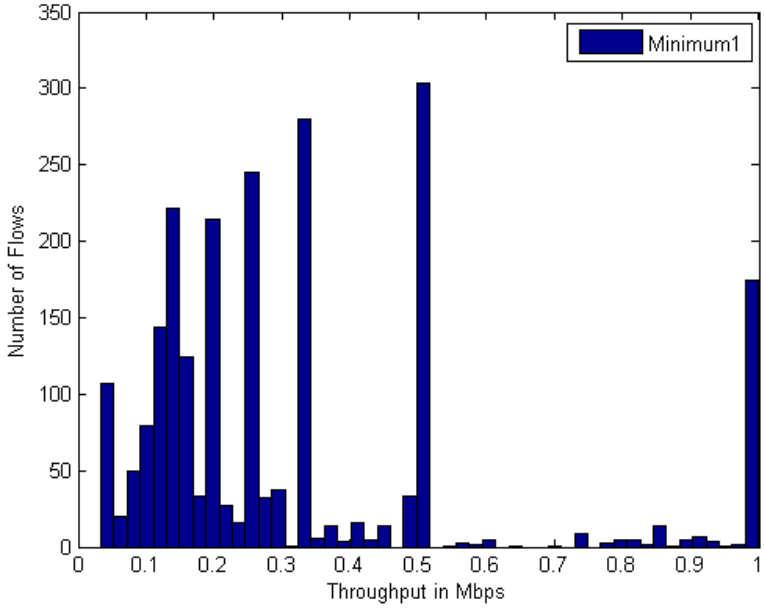

Figure 4.5: Minimum1 Policy Results for Ring Network

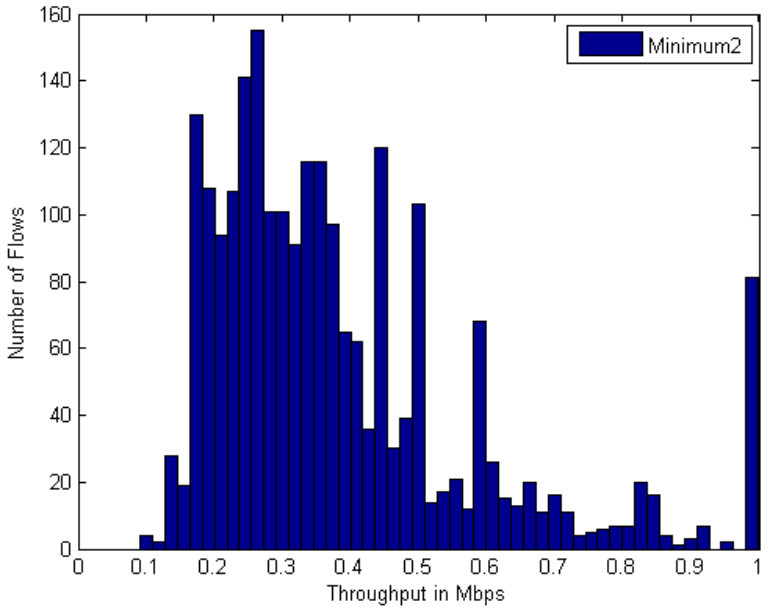

Figure 4.6: Minimum2 Policy Results for Ring Network

4.2 US Network

Assume, 4 client and 1 server positions are available to each node of US Network, Figure 3.5. Hence, we can have 14 servers and 56 clients at most in the scenario. Call the 14 nodes of the US Network as core nodes and we have servers and clients at edge nodes which are attached to each core node. Similar to the previous section, we have the following properties in this network:

1. n=4
2. k_n = U(4,14)
3. m_n = U(1,20)
4. C_v = 1 Mbps, v=1,2,...,91
5. RTT_v for core nodes are determined from Figure 3.6 and for edge links, we have 20 km of fibre length.

Policies	Mean Throughput in Mbps	Coefficient of Variation
Random1	0.2350	0.7732
Random2	0.2680	0.6149
Random3	0.2746	0.5582
Minimum1	0.2348	0.9628
Minimum2	0.2844	0.6356

Table 4.2: Simulation Results for US Network

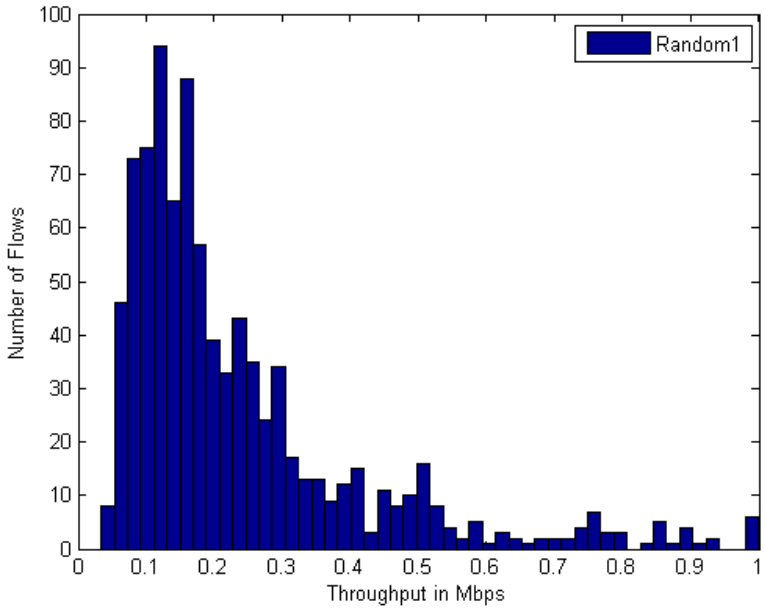

Figure 4.7: Random1 Policy Results for US Network

In MATLAB, 20 scenarios are created with 910 clients for each policy. On Table 4.2, mean throughput values for each policy is presented along with coefficient of variation of each policy. Histogram of each policy can be observed on Figure 4.7, 4.8, 4.9, 4.10, 4.11. It can be observed that significant amount of gain is achieved when using more than one server. In particular, % 14 gain is achieved between "Random1" and "Random2" and % 2 gain between "Random2" and "Random3". Moreover, we observe % 21 gain between "Minimum1" and "Minimum2". Similar as before, connecting more servers decreases coefficient of variations which indicates better fairness in terms of throughput.

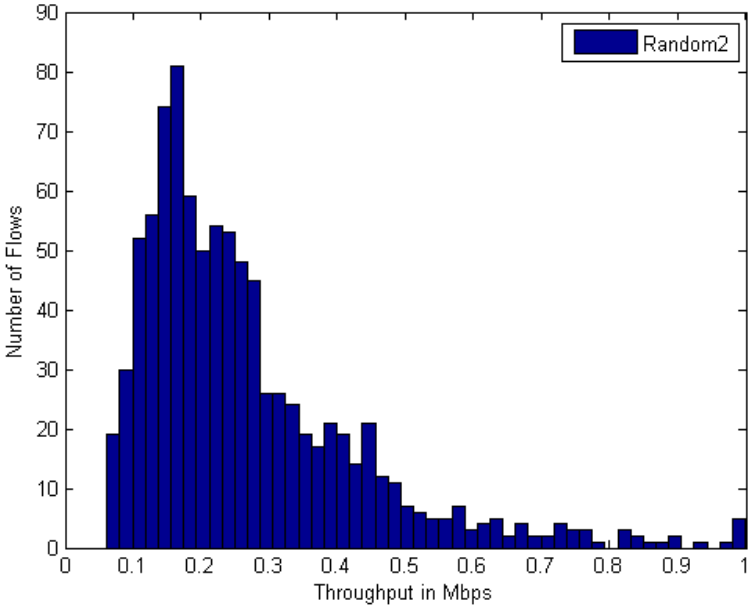

Figure 4.8: Random2 Policy Results for US Network

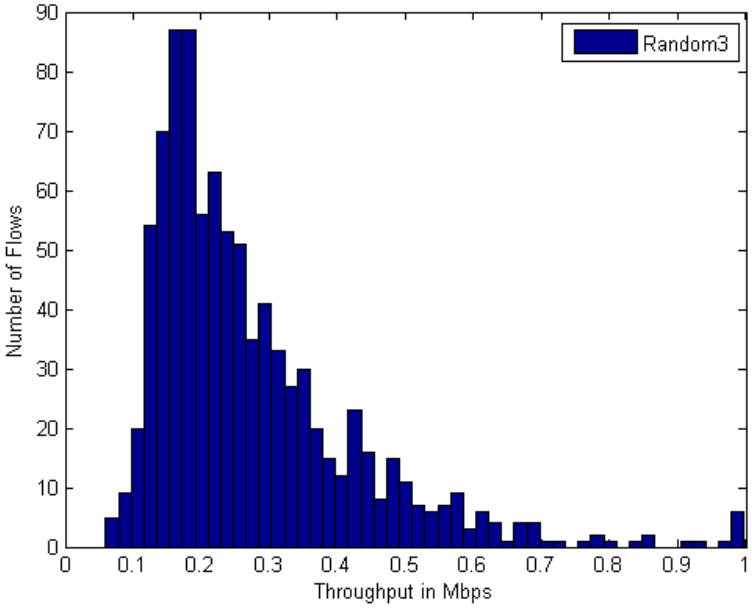

Figure 4.9: Random3 Policy Results for US Network

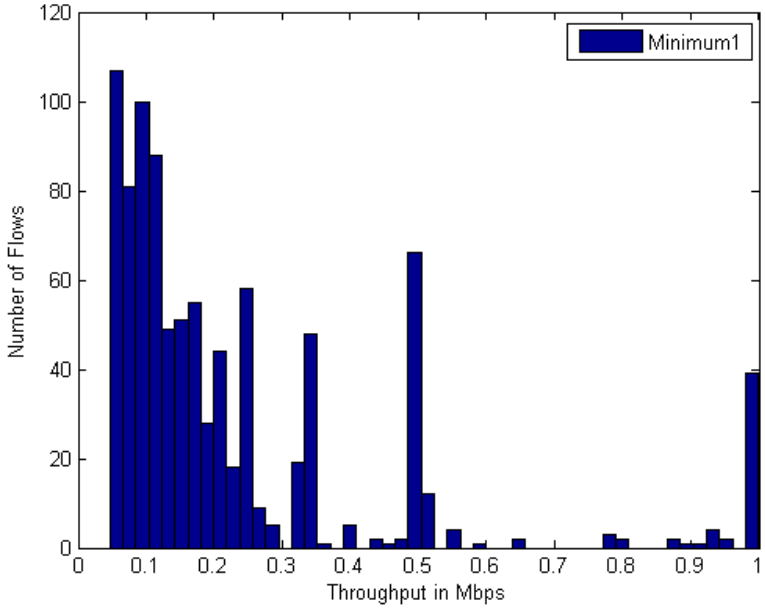

Figure 4.10: Minimum1 Policy Results for US Network

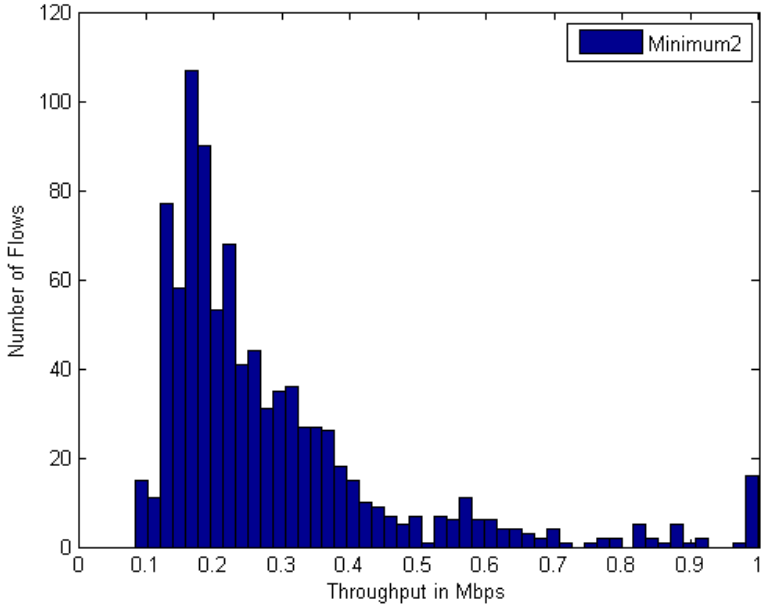

Figure 4.11: Minimum2 Policy Results for US Network

Chapter 5

Conclusion and Future Work

In this thesis, a fixed-point algorithm-based method is proposed to characterize the throughput of networks offered with a mixture of persistent TCP and UDP flows. The network is assumed to include router links which support per-class queuing and DRR scheduling. In particular, the case of two classes is studied where one of the classes goes through drop-tail queues and intended for UDP traffic whereas the other class goes through queues which implement a specific AQM mechanism. By using per-class queuing, TCP/UDP isolation is achieved for better TCP application performance. The proposed algorithm is validated by various NS3 simulations and the results indicate that the analysis tool captures the behavior of such networks in all cases. There is less than %5 difference in the results between the analysis tool and NS3 simulations for the mean throughput values. Moreover, at most % 25 error is observed for one flow's results. The difference in the results can partially be explained by not taking ACK packet traffic into account in the analysis. As a future work, the algorithm can be modified to capture the effect of the ACK traffic. Moreover, the proposed method is used to analyze the potential gain of employing many-to-one communication over classical one-to-one communication. We observed significant amount of gains, more than % 14, when using 2 random servers as opposed to 1 random server in both scenarios. Similarly, more than % 17 gain is achieved when using 2 closest servers as opposed to 1. Furthermore, coefficient of variation is decreased when using more

than one server which indicates better fairness in terms of throughput. However, in order to collect more reliable data, more scenarios should be created. Also note that the analysis needs to be viewed as a proof of concept for now. In the future, one can employ specific protocols to study such scenarios in real internet traffic.

Bibliography

[1] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang, "Recommendations on queue management and congestion avoidance in the internet," *RFC 2309*, April 1998.

[2] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *Networking, IEEE/ACM Transactions on*, vol. 1, pp. 397 –413, aug 1993.

[3] M. Hassan and R. Jain, eds., *High performance TCP/IP networking: Concepts, issues, and solutions*. Pearson, 2004.

[4] E. S. Hashem, *Analysis of random drop for gateway congestion control*. PhD thesis, 1989. PHD.

[5] S. Athuraliya, S. Low, V. Li, and Q. Yin, "REM: Active queue management," *Network, IEEE*, vol. 15, no. 3, pp. 48–53, 2001.

[6] T. J. Ott, J. H. B. Kemperman, and M. Mathis, "The stationary behavior of ideal TCP congestion avoidance," 1996.

[7] J. Mahdavi, "Tcp-friendly unicast rate-based flow control." `http://www.psc.edu/networking/papers/tcp\_friendly.html`, 1997.

[8] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, "The macroscopic behavior of the TCP congestion avoidance algorithm," *SIGCOMM Comput. Commun. Rev.*, vol. 27, pp. 67–82, July 1997.

[9] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP throughput: a simple model and its empirical validation," *SIGCOMM Comput. Commun. Rev.*, vol. 28, pp. 303–314, Oct. 1998.

[10] R. Kooij, R. van der Mei, and R. Yang, "TCP and WEB browsing performance in case of bi-directional packet loss," *Computer Communications*, vol. 33, pp. S50–S57, 2010.

[11] T. Bu and D. Towsley, "Fixed point approximations for TCP behavior in an AQM network," *SIGMETRICS Perform. Eval. Rev.*, vol. 29, pp. 216–225, June 2001.

[12] L. Ji, T. Arvanitis, and S. Woolley, "Fair Weighted Round Robin scheduling scheme for DiffServ networks," *Electronics Letters*, vol. 39, no. 3, pp. 333–335, 2003.

[13] M. Shreedhar and G. Varghese, "Efficient fair queueing using Deficit Round Robin," *SIGCOMM Comput. Commun. Rev.*, vol. 25, pp. 231–242, Oct. 1995.

[14] A. Sharma, A. Venkataramani, and R. K. Sitaraman, "Distributing content simplifies ISP traffic engineering," *SIGMETRICS Perform. Eval. Rev.*, vol. 41, pp. 229–242, June 2013.

[15] S.Floyd, "Recommendation on using the gentle_ variant of RED." `http://www.icir.org/floyd/red/gentle.html`, Mar. 2000.

[16] J.-C. Chen, "Dijkstra's shortest path algorithm," *Journal of Formalized Mathematics*, vol. 15, 2003.

[17] S. Ramroop, "A diffserv model for the ns-3 simulator." `http://www.eng.uwi.tt/depts/elec/staff/rvadams/sramroop/`, 2011.

[18] A. Betker, C. Gerlach, R. Hülsermann, M. Jäger, M. Barry, S. Bodamer, J. Späth, C. Gauger, and M. Köhn, "Reference transport network scenarios," *MultiTeraNet Report, July*, 2003.

[19] B. Chinoy and H.-W. Braun, "The National Science Foundation Network," tech. rep., Technical Report GA-A21029, SDSC, 1992.

[20] M. A. Shokrollahi and M. Luby, "Raptor codes," *Foundations and Trends in Communications and Information Theory*, vol. 6, no. 3-4, pp. 213–322, 2009.