Telecommun Syst (2017) 64:585–598
DOI 10.1007/s11235-016-0191-1

Fixed-point analysis of a network of routers with persistent TCP/UDP flows and class-based weighted fair queuing

Caglar Tunc[1] · **Nail Akar**[1]

Published online: 8 July 2016

Abstract Fixed-point models have already been successfully used to analytically study networks consisting of persistent TCP flows only, or mixed TCP/UDP flows with a single queue per link and differentiated buffer management for these two types of flows. In the current study, we propose a nested fixed-point analytical method to obtain the throughput of persistent TCP and UDP flows in a network of routers supporting class-based weighted fair queuing allowing the use of separate queues for each class. In particular, we study the case of two classes where one of the classes uses drop-tail queue management and is intended for only UDP traffic. The other class targeting TCP, but also allowing UDP traffic for the purpose of generality, is assumed to employ active queue management. The effectiveness of the proposed analytical method is validated in terms of accuracy using ns-3 simulations and the required computational effort.

Keywords TCP · Active queue management · UDP · Class-based weighted fair queuing · Fixed-point analysis

This work is supported by the Science and Research Council of Turkey (Tubitak) under projects no. 111E106 and 115E360.

✉ Nail Akar
akar@ee.bilkent.edu.tr

Caglar Tunc
caglar@ee.bilkent.edu.tr

[1] Electrical and Electronics Engineering Department, Bilkent University, Ankara, Turkey

1 Introduction

In today's Internet, transmission control protocol (TCP) and user datagram protocol (UDP) are the most dominant protocols for the transport layer. Using 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 [8]. The full queues problem can be described as the buffer being occupied most of the time which leads to large queuing delays and thus 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 have been proposed which drop packets without waiting for the queue to be full [8]. The AQM drop decision is generally probabilistic on certain queue parameters to mitigate the lock-out problem [8]. In this paper, we do not delve into the problem of parameter optimization for AQM but rather assume that the parametrization of a given AQM scheme is given.

Increasing use of real-time voice and video applications has led to changes in the "UDP to TCP ratio" trend in today's Internet which is further discussed in [28]. Since UDP does not have a congestion control mechanism as TCP, TCP flows sharing the same link with UDP flows may starve because of UDP flows' unresponsive behavior, known as congestion collapse [16,34]. Obviously, this is not desirable for applications using TCP. In this paper, we envision class-based queuing at the network links to address the congestion collapse problem. We focus on the particular case of two classes, namely classes 1 and 2, where persistent UDP flows can join either Class 1 or Class 2 and TCP flows are only allowed to join Class 2. If all UDP flows join Class 1, then we have complete

isolation between UDP and TCP flows. On the other hand, if all UDP flows join Class 2, then we would not have any isolation, allowing us to study both the two extreme cases with a unifying framework. We envision the use of AQM for Class 2 queues and drop-tail as the buffer management mechanism for Class 1. We also assume Class-based Weighted Fair Queuing (CBWFQ)-based scheduling among per-class queues which refers to a set of link-based techniques in which each class receives a weighted fair share of link resources and the buffer management of a class is configured independently of others. CBWFQ is a term coined by Cisco and is elaborated in several references including [7,13,23]. Weighted Round Robin (WRR) and Deficit Round Robin (DRR) are specific schemes for weighted fair queuing [24,42] where the latter scheme is proposed for variable-sized packets [42]. The DRR algorithm is shown to achieve near-perfect weighted fairness in terms of throughput while requiring less computational complexity to process a packet when compared with other mechanisms. The main goal of this paper is to devise an analytical method to calculate the performance experienced by the persistent TCP and UDP flows using the network, in terms of the per-flow throughputs. Although it may be desirable to analyse cases with more than two classes of traffic, such scenarios in which TCP flows may further be segregated into further classes depending on their flow lengths, quality of service requirements, etc., as in [44], are left for future research.

In the literature, fixed-point iterative models have been used to study a network consisting of persistent TCP flows only in [9,18,19]. For the case of TCP and UDP flows queued in a single buffer at the network links, an extended fixed-point model is proposed in [39] to study the impact of differentiated buffer management on the performance of TCP/UDP flows. However, to the best of our knowledge, a specific method has not been proposed in the literature to study a network of routers carrying persistent TCP/UDP flows and per-class queuing at the network links. The goal of this paper is to fill this research gap and devise a fixed-point analytical model for this scenario.

The main contributions of this study are given as follows. Using the TCP send rate formula provided in [9], we propose a nested fixed-point iterative algorithm to study a network of routers of arbitrary topology using CBWFQ-based scheduling on inter-router links and which is offered with an arbitrary number of persistent TCP/UDP flows. The nested fixed-point model consists of one single outer loop with two inner loops, one loop per class. We validate the accuracy of the model by employing ns-3 simulations. Moreover, we demonstrate convergence statistics of the fixed-point iterations by providing the number of iterations and the computation time required until convergence for various scenarios. Although we do not have a formal proof for the convergence of the proposed algorithm, we observed that the algorithm always converged within plausible amount of time for all the network scenarios we studied.

The paper is organized as follows. In Sect. 2, we recapitulate the related work. We present the nested fixed-point iterations to obtain the per-flow throughputs in networks with per-class queuing offered with persistent TCP and UDP flows in Sect. 3. The proposed analytical method is validated by ns-3 simulations with various numerical examples in Sect. 4. Finally, we conclude.

2 Related work

In this section, we summarize the related work in the following three categories: AQM techniques, TCP-UDP interaction, and analytical models for TCP.

2.1 AQM techniques

In the literature, there are several AQM techniques proposed such as random early detection (RED) [17,22], early random drop (ERD) [21], random exponential marking (REM) [4]. Performance of various AQM mechanisms in terms of packet travel times and packet loss probabilities is studied in [20]. Optimization of AQM parameters for various traffic scenarios has also been an active area of research; see for example [45] that studies a RED-controlled router that automatically tunes its RED parameters, and also [43] for a self-tuning proportional and integral-type feedback controller extension of the basic RED. The reference [33] uses a traffic prediction technique to decide packet drops in AQM whereas in [25], another controller design is proposed to increase the performance of AQM for TCP traffic. As stated before, we assume a certain variant of RED as the particular AQM mechanism to be used in this study although the work can be extended to more general AQM mechanisms.

2.2 TCP-UDP interaction

There are several studies that focus on improving the performances of TCP's congestion control mechanism and UDP's unresponsive behavior. The reference [14] proposes a TCP variant which uses a novel AQM technique to increase TCP's end-to-end delay performance whereas a variant of UDP that employs congestion control as in TCP is studied in [10]. The references [37] and [40] aim at providing bandwidth guarantees to TCP flows in cloud networks with both TCP and UDP flows. To mitigate the TCP starvation problem, per-class queuing has been proposed in which flows using different transport layers are classified into separate service queues per transport layer [44]. On the other hand, a class-based buffer management approach is proposed in [3] in which packets

belonging to different classes experience different dropping policies to preserve TCP/UDP fairness.

2.3 Analytical models for TCP

In the literature, different analytical expressions have been proposed for characterizing the send rate of TCP's congestion control mechanism as a function of the packet loss and round trip delay [27,29,31,35,36]. In this study, we use the TCP send rate formula proposed in [36] 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 [29,31,35] ignore certain features of TCP and consequently over-estimate TCP throughput while proposing simpler expressions. Using fixed-point iterations using the analytical expression for TCP flows' send rates given in [36], one can study the behavior of a TCP flow in a network of AQM routers offered with persistent TCP flows [9]. A similar fixed-point analysis following an M/D/1 queuing model for each AQM link is given in [19]. An elaborate queuing analysis is proposed by [41] for a finite queue with its arrivals controlled by the random early detection algorithm. The authors of [41] study the exact dynamics of this queue and provide the stability and the rates of convergence to the stationary distribution and obtain the packet loss probability and the waiting time distributions for this queue. In [32], the authors have developed a methodology to model and obtain the expected transient behavior of networks with AQM routers supporting TCP flows. An analytical framework for modeling a network of RED queues with mixed UDP and TCP traffic is introduced in [1] which only allows one single queue for both traffic types but differentiated buffer management as opposed to per-class queuing. However, to the best of our knowledge, a method has not been proposed to analyse a network containing persistent TCP/UDP traffic and routers with per-class queuing. The goal of this paper is to devise an analytical model based on the work of [9] that enables us to analyse networks offered with persistent TCP and UDP flows in a network of routers using CBWFQ-based scheduling on inter-router links.

3 Fixed-point analysis of a network of routers

We assume a network of routers that are offered with persistent UDP and TCP flows. Consider a link v that interconnects two routers, in a network with V links, with transmission capacity C_v (in units of bps). The link v employs a variant of WFQ among the two queues $Q_v^{(1)}$ and $Q_v^{(2)}$ which are assigned the scheduling weights $w_v^{(1)}, 0 \leq w_v^{(1)} \leq 1$, and $w_v^{(2)} = 1 - w_v^{(1)}$, respectively. Moreover, let $B_v^{(1)}$ and $B_v^{(2)}$ denote the queue sizes of $Q_v^{(1)}$ and $Q_v^{(2)}$, respectively. The throughput, i.e., the average queue drainage rate, of the queues $Q_v^{(1)}$ and $Q_v^{(2)}$, are denoted by $C_v^{(1)}$ and $C_v^{(2)}$, respectively. If both queues are non-empty, then $C_v^{(l)} = C_v w_v^{(l)}, l = 1, 2$. However, these quantities can not exceed the queue demands in which case we will have empty queues and the queue's throughput will be equal to its demand. Drop-tail queue management is envisioned for the queue $Q_v^{(1)}$ since TCP flows are not allowed to join the queue $Q_v^{(1)}$. Let the probability drop function at the queue $Q_v^{(1)}$ be denoted by $p_v^{(1)}(x_v^{(1)})$ where $x_v^{(1)}$ is the queue occupancy of $Q_v^{(1)}$. On the other hand, all TCP and optionally some UDP flows are allowed to join $Q_v^{(2)}$. Therefore, we propose to use RED with probability drop function $p_v^{(2)}(x_v^{(2)})$ where $x_v^{(2)}$ is the current queue occupancy of $Q_v^{(2)}$. For the fluid fixed-point analysis, we need to have an injective probability drop function which leads us to the generic expression given in Eq. (1) for $p_v^{(2)}(x_v^{(2)})$ that reduces to the gentle variant of RED (G-RED) of [15] when $p_v^{(2)min}$ and $t_v^{(2)min}$ are set to zero. In particular, the quantity $p_v^{(2)}(x_v^{(2)})$ equals

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

Note that the parameters $p_v^{(2)min}$ and $t_v^{(2)min}$ are to be set to values very close to zero in the numerical experiments to be compatible with the G-RED curve that is used in [9]. The generic G-RED curve we use in the analysis for all the links is depicted in Fig. 1. For the traffic demands, we assume $K_U^{(1)}$ persistent UDP flows, each flow labeled as $i_u^{(1)} = 1, \ldots, K_U^{(1)}$ that belong to class 1. Similarly, we assume $K_U^{(2)}$ persistent UDP and $K_T^{(2)}$ persistent TCP flows for class 2 where the UDP flows are labeled as $i_U^{(2)} = 1, \ldots, K_U^{(2)}$ and TCP flows are labeled as $i_T^{(2)} = 1, \ldots, K_T^{(2)}$. In case there is no isolation

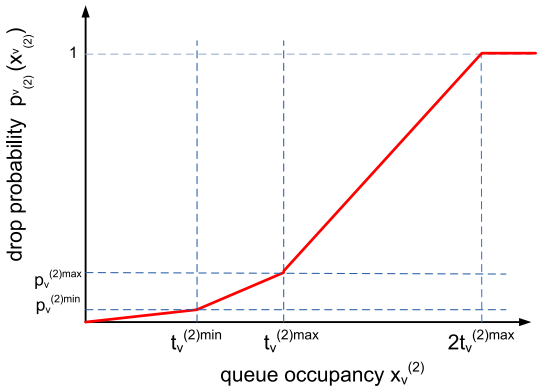


Fig. 1 The generic G-RED drop probability curve

between TCP and UDP flows, then we have $K_U^{(1)} = 0$. If we have complete isolation, then $K_U^{(2)} = 0$. Partial isolation between TCP and UDP flows can also be addressed by setting $K_U^{(l)} > 0, l = 1, 2$ in this general framework. Let $V_{i_T^{(2)}}$ be the ordered set of links for flow $i_T^{(2)}$ which can be found using Dijkstra's shortest path algorithm with all the link weights set to one, i.e., min-hop path [11]. In order to express the TCP send rate of each flow, the round-trip time and the end-to-end loss probability needs to be calculated. Denoting the one-way fixed propagation delay of a link v by P_v, the expected round-trip time for flow $i_T^{(2)}$, denoted by $R_{i_T^{(2)}}$, can be expressed as:

$$R_{i_T^{(2)}} = 2 \sum_{v \in V_{i_T^{(2)}}} P_v + \sum_{v \in V_{i_T^{(2)}}} \frac{x_v^{(2)}}{C_v^{(2)}}, \tag{2}$$

where the first term is the two-way fixed propagation delay for TCP flow $i_T^{(2)}$ and the second term is the queuing delay experienced by the flow $i_T^{(2)}$ on its 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:

$$q_{i_T^{(2)}} = 1 - \prod_{v \in V_{i_T^{(2)}}} (1 - p_v^{(2)}). \tag{3}$$

Similarly, let $V_{i_U^{(1)}}$ and $V_{i_U^{(2)}}$ be the ordered set of links used by UDP flows $i_U^{(1)}$ and $i_U^{(2)}$, respectively. Consequently, the end-to-end loss probabilities for these flows can be written as follows:

$$q_{i_U^{(l)}} = 1 - \prod_{v \in V_{i_U^{(l)}}} (1 - p_v^{(l)}), \quad l = 1, 2. \tag{4}$$

In the literature, several expressions in various complexities have been proposed for the TCP send rate of an individual flow [22]. In our proposed model, we use the TCP send rate expression suggested by [36] which is shown to capture TCP's fast retransmit and timeout mechanisms. The maximum congestion window size W_{max} is determined at the beginning of TCP flow establishment [36]. For the purpose of using this expression, we first define the expected value of the unconstrained window size denoted by $E[W_{i_T^{(2)}}]$ of flow $i_T^{(2)}$ based on [36]:

$$E\left[W_{i_T^{(2)}}\right] = \frac{2+b}{3b} + \sqrt{\frac{8(1-q_{i_T^{(2)}})}{3bq_{i_T^{(2)}}} + \left(\frac{2+b}{3b}\right)^2}, \tag{5}$$

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 set to two. We also have the probability that loss in a window of size w is detected by time-outs denoted by $Q_{i_T^{(2)}}(w)$=

$$min\left(1, \frac{\left(1-\left(1-q_{i_T^{(2)}}\right)^3\right)\left(1+\left(1-q_{i_T^{(2)}}\right)^3\right)\left(1-\left(1-q_{i_T^{(2)}}\right)^{w-3}\right)}{1-\left(1-q_{i_T^{(2)}}\right)^w}\right). \tag{6}$$

Using the identities (5) and (6), and on the basis of work in [36], one can write the TCP send rate of flow $i_T^{(2)}$, namely $T(i_T^{(2)}) =$

$$\frac{\frac{1-q_{i_T^{(2)}}}{q_{i_T^{(2)}}} + E\left[W_{i_T^{(2)}}\right] + Q_{i_T^{(2)}}\left(E\left[W_{i_T^{(2)}}\right]\right)\frac{1}{1-q_{i_T^{(2)}}}}{R_{i_T^{(2)}}\left(\frac{b}{2}E[W_{i_T^{(2)}}]+1\right) + Q_{i_T^{(2)}}\left(E[W_{i_T^{(2)}}]\right)T_0 F\left(i_T^{(2)}\right)\frac{1}{1-q_{i_T^{(2)}}}} \tag{7}$$

if $E[W_{i_T^{(2)}}] < W_{max}$. Otherwise, $T(i_T^{(2)}) =$

$$\frac{\frac{1-q_{i_T^{(2)}}}{q_{i_T^{(2)}}} + W_{max} + \frac{Q_{i_T^{(2)}}(W_{max})}{1-q_{i_T^{(2)}}}}{R_{i_T^{(2)}}\left(\frac{b}{8}W_{max} + \frac{1-q_{i_T^{(2)}}}{q_{i_T^{(2)}}W_{max}} + 2\right) + Q_{i_T^{(2)}}(W_{max})T_0 F(i_T^{(2)})\frac{1}{1-q_{i_T^{(2)}}}} \tag{8}$$

where T_0 denotes the timeout period and $F(i_T^{(2)}) =$

$$1 + q_{i_T^{(2)}} + 2q_{i_T^{(2)}}^2 + 4q_{i_T^{(2)}}^3 + 8q_{i_T^{(2)}}^4 + 16q_{i_T^{(2)}}^5 + 32q_{i_T^{(2)}}^6. \tag{9}$$

For detailed derivation of the TCP send rate, we refer the reader to [36]. Note that the TCP send rate formula in (8) assumes that TCP ACK traffic does not encounter losses on the way back, which is not typically the case. By TCP ACK traffic prioritization over other data traffic as in [26], not only

TCP performance can be improved in general but Eq. (8) provides a more precise characterization for the TCP send rate in real networks. This is the approach we take in the current study. Moreover, let $Y(i_T^{(2)})$ denote the throughput of the TCP flow $i_T^{(2)}$:

$$Y(i_T^{(2)}) = T\left(i_T^{(2)}\right)\left(1 - q_{i_T^{(2)}}\right), \tag{10}$$

which is the main performance metric of interest that we attempt to obtain analytically.

Let the send rates of UDP flows $i_U^{(1)}$ and $i_U^{(2)}$ be denoted by $T(i_U^{(1)})$ and $T(i_U^{(2)})$, respectively. Recall that the send rate of TCP flows can be calculated using (8). Given the loss probabilities for each link, the total traffic demand on link v's two queues, denoted by $D_v^{(1)}$ and $D_v^{(2)}$, can be calculated. For this purpose, let $S_U(v,l)$ be the set of UDP flows which use queue $Q_v^{(l)}, l = 1,2$. For $i_U^{(l)} \in S_U(v,l)$, let $Z_{v,i_U^{(l)}}$ be the set of ordered links ordered as the route of the flow $i_U^{(l)}$ until it reaches the link v again for $l = 1,2$. Similarly, let $S_T(v)$ be the set of TCP flows which use queue $Q_v^{(2)}$. For $i_T^{(2)} \in S_T(v)$, let $Z_{v,i_T^{(2)}}$ be the set of ordered links that constitute the route of the flow $i_T^{(2)}$ until it reaches the link v. With these definitions, it is not difficult to write demands $D_v^{(1)}$ and $D_v^{(2)}$ as in Eqs. (11) and (12), respectively.

$$D_v^{(1)} = \sum_{i_U^{(1)} \in S_U(v,1)} T(i_U^{(1)}) \prod_{u \in Z_{v,i_U^{(1)}}} (1 - p_u^{(1)}), \tag{11}$$

$$\begin{aligned} D_v^{(2)} = & \sum_{i_U^{(2)} \in S_U(v,2)} T(i_U^{(2)}) \prod_{u \in Z_{v,i_U^{(2)}}} (1 - p_u^{(2)}) \\ & + \sum_{i_T^{(2)} \in S_T(v)} T(i_T^{(2)}) \prod_{u \in Z_{v,i_T^{(2)}}} (1 - p_u^{(2)}). \end{aligned} \tag{12}$$

The link loss probabilities can then be written as follows:

$$p_v^{(l)} = \begin{cases} 1 - \frac{C_v^{(l)}}{D_v^{(l)}}, & C_v^{(l)} \le D_v^{(l)}, l = 1,2. \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

Once we know $p_v^{(l)}, l = 1,2$, the per-flow throughputs for UDP flows, denoted by $Y(i_U^{(l)})$, of the flow $i_U^{(l)}, l = 1,2$, can be found by using the following identity:

$$Y(i_U^{(l)}) = T(i_U^{(l)})(1 - q_{i_U^{(l)}}), \quad l = 1,2. \tag{14}$$

We propose a nested fixed-Point Iterations (NFPI) algorithm to solve the per-flow TCP and UDP throughputs in Algorithm 1 whose nomenclature is provided in Table 1. The algorithm consists of an outer loop and two inner loops (one inner loop per class). We initially set $C_v^{(l)} = C_v w_v^{(l)}, l = 1,2, \forall v \in V$. Then, given $C_v^{(1)}$, we use fixed-point iterations to solve $D_v^{(1)}$ and $p_v^{(1)}$ for all $v \in V$. Once the demands $D_v^{(1)}$ are found then we decide which of the class 1 queues are empty by comparing $D_v^{(1)}$ against $C_v^{(1)}$. Having obtained $D_v^{(1)}$, we use fixed-point iterations to solve for per-flow throughputs of UDP and TCP flows sharing class 2 queues using a similar scheme as in [9] which is detailed in Algorithm 1. One of the differences of our scheme than that of [9] in this step is the combined treatment of both UDP and TCP flows. Moreover, we solve class 2 queues one link at a time using binary search. To explain, we fix a link v. For the most recent values of $T(i_T^{(l)})$ for $l = 1,2$, we solve for $q_{i_T^{(2)}}$, and new send rates $\bar{T}(i_T^{(l)})$ for TCP flows using link v. After all new send rates are obtained, we equate $T(i_T^{(l)})$ to $\bar{T}(i_T^{(l)})$. Then, after calculating $D_v^{(2)}$, we decide whether $x_v^{(2)}$ is empty or not by comparing $D_v^{(2)}$ against $C_v^{(2)}$. If $C_v^{(2)}$ is

Table 1 Nomenclature of Algorithm 1

Input	
$Q_v^{(j)}$	Queue of class j on link v
C_v	Capacity of link v (bps)
$w_v^{(j)}$	Scheduling weight of class j on link v
$T(i_U^{(j)})$	Send rate of UDP flow $i_U^{(j)}$ (bps)
x_v^l	Binary search thresh. for $x_v^{(2)}, l \in \{-,+\}$
ε_j	Tolerance parameters, $1 \le j \le 3$ (bps)
ζ	Threshold convergence parameter (bits)
$t_v^{(2)min}$	G-RED parameters
$t_v^{(2)max}$	
$p_v^{(2)min}$	
$p_v^{(2)max}$	
Output	
$x_v^{(j)}$	Queue occupancy of $Q_v^{(j)}$ (bits)
$p_v^{(j)}$	Loss probability of link v for class j
$C_v^{(j)}$	Queue drain rate of $Q_v^{(j)}$ (bps)
$D_v^{(j)}$	Demand on link v for class j (bps)
$q_{i_U^{(j)}}$	Loss prob. for UDP flow $i_U^{(j)}$ in class j
$q_{i_T^{(2)}}$	Loss prob. of TCP flow $i_T^{(2)}$
$T(i_T^{(2)})$	Send rate of TCP flow $i_T^{(2)}$ (bps)
$Y(i_T^{(2)})$	Throughput of TCP flow $i_T^{(2)}$ (bps)
$Y(i_U^{(j)})$	Throughput of UDP flow $i_U^{(j)}$ (bps)

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 *Maximum throughput difference between two subsequent iterations is larger than* ε_1 **do**
  Start with Class 1 first;
  while *Max. throughput difference between two subsequent Class 1 iterations is larger than* ε_2 **do**
    Find the loss probability $p_v^{(1)}$ for each link v in accordance with Eqn. (13);
    Find end-to-end drop probabilities $q_{i_U^{(1)}}$ of flows from Eqn. (4);
    Find the demand $D_v^{(1)}$ on each link v on the basis of Eqn. (11);
  end
  Continue with Class 2;
  while *Max. throughput difference between two subsequent Class 2 iterations is larger than* ε_3 **do**
    Initialization: $p_v^{(2)} \leftarrow 0$; $x_v^{(2)} \leftarrow 0$; $x_v^{+} \leftarrow 2t_v^{(2)max}$; $x_v^{-} \leftarrow 0$; $\forall v \in V$;
    for *v = 1: total number of links TCP flows utilize* **do**
      $is\_solved \leftarrow 0$;
      while *is_solved* $\neq 1$ **do**
        For flows using v, find end-to-end drop probabilities $q_{i_U^{(2)}}$ and $q_{i_T^{(2)}}$ from Eqn. (3) and Eqn. (4) ;
        For TCP flows using v, find new send rates $\bar{T}(i_T^{(2)})$ from Eqn. (8) ;
        After solving $\bar{T}(i_T^{(2)})$ for all flows, $T(i_T^{(2)}) \leftarrow \bar{T}(i_T^{(2)})$;
        Find the demand $D_v^{(2)}$ based on Eqn. (12);
        if $\left(D_v^{(2)}(1 - p_v^{(2)}) \leq C_v^{(2)}$ *and* $x_v^{(2)} = 0\right)$ *or* $D_v^{(2)}(1 - p_v^{(2)}) = C_v^{(2)}$ *or* $x_v^{+} - x_v^{-} \leq \zeta$ **then**
          $is\_solved \leftarrow 1$;
        else if $D_v^{(2)}(1 - p_v^{(2)}) > C_v^{(2)}$ **then**
          $x_v^{-} \leftarrow x_v^{(2)}$;
          $x_v^{(2)} \leftarrow (x_v^{(2)} + x_v^{+})/2$;
        else if $D_v^{(2)}(1 - p_v^{(2)}) < C_v^{(2)}$ **then**
          $x_v^{+} \leftarrow x_v^{(2)}$;
          $x_v^{(2)} \leftarrow (x_v^{(2)} + x_v^{-})/2$;
        end
        Find $p_v^{(2)}$ from $x_v^{(2)}$ in accordance with Eqn. (1) ;
      end
    end
  end
  Start capacity update process;
  if $D_v^{(1)} > C_v^{(1)}$ *and* $D_v^{(2)} < C_v^{(2)}$ **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 $D_v^{(1)} < C_v^{(1)}$ *and* $D_v^{(2)} > C_v^{(2)}$ **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 $D_v^{(1)} > C_v^{(1)}$ *and* $D_v^{(2)} > C_v^{(2)}$ **then**
    $C_v^{(1)} \longleftarrow C_v w_v^{(1)}$;
    $C_v^{(2)} \longleftarrow C_v(1 - w_v^{(1)})$;
  end
end

Algorithm 1: Nested Fixed-Point Iterative (NFPI) Algorithm

greater than $D_v^{(2)}$, we decide it is empty. Otherwise, we find $x_v^{(2)}$ with a binary search between 0 and $2t_v^{(2)max}$. We repeat this binary search method to solve for all links carrying TCP flows. Subsequent to the fixed-point iterations, the capacity update process is established in the following manner. If the demand $D_v^{(1)}$ is less than $C_v^{(1)}$ and $D_v^{(2)}$ is greater than $C_v^{(2)}$, excess capacity in $C_v^{(1)}$ is handed to class 2. Similarly, excess capacity in $C_v^{(2)}$ can be given to class 1. Finally, if both $D_v^{(1)}$ and $D_v^{(2)}$ are larger than $C_v^{(1)}$ and $C_v^{(2)}$, respectively, we set both capacities to their guaranteed values, $C_v^{(1)}$ to $C_v w_v^{(1)}$ and $C_v^{(2)}$ to $C_v(1 - w_v^{(1)})$. Obtaining the class 1 and 2 per-flow throughputs in this manner concludes one iteration of the outer loop. Subsequently, this process repeats itself until convergence on the quantities $p_v^{(l)}$ and $x_v^{(l)}$ for all $v \in V$ and $l = 1, 2$. Although we do not have a formal proof, we reached convergence in all the numerical examples we studied.

4 Numerical examples

In the first two numerical examples, we provide two different network topologies to assess the accuracy of the proposed NFPI algorithm implemented using MATLAB by comparing the results with that of simulations using ns-3. We based our ns-3 simulations on the study presented in [38] which consists of an ns-3 implementation of IETF differentiated services (Diffserv) architecture [6]. The Expedited Forwarding (EF) class in this implementation has strict priority over other classes and uses drop-tail queue management and is dedicated to TCP acknowledgment packets in our study. The Assured Forwarding (AF) class AF1 is mapped to class 1 in our study whereas AF2 is mapped to class 2. On the other hand, AF1 uses drop-tail whereas AF2 uses G-RED buffer management. A DRR scheduler serves the two AF queues with configurable scheduling weights. The routes for the flows are obtained using Dijkstra's shortest path algorithm for both analysis and simulations with all link weights set to unity. The network parameters used throughout the numerical examples of this study are listed in Table 2. We set $p_v^{(2)min}$ to zero in ns-3 simulations; however, in NFPI, the parameter

Table 2 Network parameters used in the numerical examples

Packet size P	1000 Bytes
$t_v^{(2)min}$	$30P$
$t_v^{(2)max}$	$90P$
$B_v^{(2)}$	$180P$
$p_v^{(2)min}$	0
$p_v^{(2)max}$	0.1
Time-out parameter T_0	0.2 s

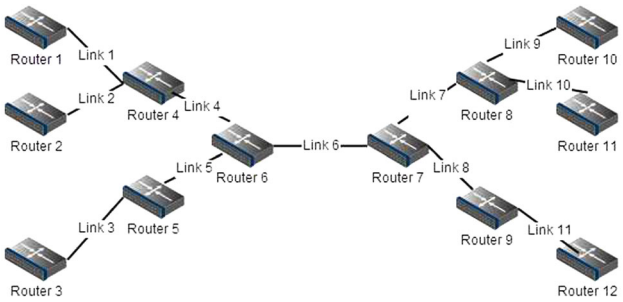


Fig. 2 The network topology for Example I.

Table 3 Simulation results for Example I in terms of per-flow throughputs

Flow	Throughput (Mbps)					
	$w_v^{(1)} = 0.25$		$w_v^{(1)} = 0.50$		$w_v^{(1)} = 0.75$	
	ns-3	NFPI	ns-3	NFPI	ns-3	NFPI
TCP 1-10	3.017	2.979	2.022	1.941	0.986	0.929
TCP 2-11	1.539	1.543	1.125	1.118	0.676	0.643
TCP 3-12	3.069	2.979	2.061	1.941	1.004	0.929
UDP 1-10	0.601	0.639	1.351	1.426	2.263	2.327
UDP 2-11	0.439	0.456	0.852	0.886	1.255	1.284
UDP 3-12	1.304	1.405	2.561	2.688	3.793	3.889

$p_v^{(2)min}$ is set to 0.001 for all links as explained before. UDP traffic is assumed to be Poisson for simulation purposes.

4.1 Example I

We first study the network topology given in Fig. 2. We assume full isolation between UDP and TCP flows; only TCP flows join class 2. The network includes one TCP and one UDP flow from Router i to Router $i + 9$ for $1 \le i \le 3$, which amounts to six overall flows. All links in this simple network have the same capacity $C_v = 10$ Mbps and propagation delay $P_v = 2$ ms for $1 \le v \le 11$ except for link 2 whose propagation delay P_2 is set to 20 ms. We study three different scenarios concerning $w_v^{(1)} = 0.25, 0.50, 0.75$ for all v. The send rates are set to 10 Mbps for UDP flows between routers 1–10 and 3–12, and it is set to 5 Mbps for the flow between routers 2–11. Finally, five ns-3 simulations are carried out each having a duration of 500 sec. and average throughput results are reported. The results obtained by ns-3 and the NFPI analytical method are presented in Table 3. The results obtained by NFPI match the simulation results acceptably well in all three scenarios for all the six flows. The number of inner loop iterations required until convergence at each outer loop for the three different $w_v^{(1)}$ values are presented in Table 4 demonstrating the rapid convergence of NFPI.

Table 4 Number of required iterations for the two inner loops at each turn of the outer loop for Example I

$w_v^{(1)}$:	0.25	0.5	0.75
Outer 1:			
Class 1:	5	5	5
Class 2:	4	4	4
Outer 2:			
Class 1:	4	4	4
Class 2:	1	1	1
Outer 3:			
Class 1:	3	3	3
Class 2:	1	1	1
Outer 4:			
Class 1:	1	1	1
Class 2:	1	1	1
Total			
Outer:	4	4	4
Class 1:	13	13	13
Class 2:	7	7	7

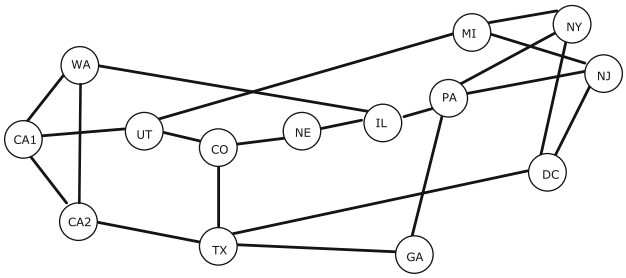


Fig. 3 NSF network topology of Example II

4.2 Example II

In Example II, we employ the NSF network topology given in Fig. 3 which consists of 14 nodes with 21 links [5,12]. The propagation delays of the network links can be obtained from the inter-nodal distances given in [5]. The link capacities C_v are set to 10 Mbps for all the links. Furthermore, 20 UDP and 20 TCP flows are assumed in Example II where all UDP flows have a network send rate of 3 Mbps. For reproducibility purposes, the source and destination nodes for each flow are given in Table 5. Recall that minimum hop routing is employed in this study. In case of two or more minimum hop paths for a given flow, one of such paths is used for the flow, which is explicitly given at the end of Table 5 again for reproducibility.

Three different scenarios corresponding to the choice of $w_v^{(2)} = 0.25, 0.50, 0.75$ are used for comparison between NFPI and ns-3 simulations. The average of two simulation runs each with a duration of 100 s are reported for each scenario. TCP throughput values obtained by ns-3 simulations and the NFPI analytical method are presented in Figs. 4a, 5a, and 6a, whereas Figs. 4b, 5b, and 6b present the UDP throughputs for each scenario. For the sake of comparison, we also present the TCP and UDP flow-level throughputs when all UDP flows join Class 2 for which there is no isolation between UDP and TCP traffic in Fig. 7.

In all scenarios we tried, we have been able to accurately estimate the TCP and UDP per-flow throughput by the NFPI algorithm when compared with ns-3 simulations. The mean and maximum per-flow percentage throughput errors while using NFPI in comparison with ns-3 simulations are provided in Table 6. The number of inner loop iterations required until

Table 5 Source-destination node pairs for all the forty flows used in Example II

Flow	Source	Dest.	Flow	Source	Dest.
TCP1	CO	DC	UDP1	DC	GA
TCP2	TX	CA1	UDP2	CA2	CO
TCP3[a]	MI	TX	UDP3	NE	UT
TCP4	UT	TX	UDP4	NJ	CA2
TCP5	IL	WA	UDP5	UT	CA2
TCP6	UT	CO	UDP6	UT	NE
TCP7	NJ	WA	UDP7	CO	DC
TCP8	WA	CA2	UDP8[f]	NE	CA2
TCP9[b]	PA	MI	UDP9	DC	NY
TCP10	PA	NE	UDP10[g]	NE	CA1
TCP11	GA	CO	UDP11	UT	TX
TCP12	MI	CA2	UDP12[h]	IL	UT
TCP13[c]	PA	CA2	UDP13	IL	PA
TCP14[d]	TX	IL	UDP14	UT	CA1
TCP15	MI	CA1	UDP15	TX	GA
TCP16	NY	MI	UDP16	CA1	UT
TCP17	NE	CO	UDP17	NJ	WA
TCP18	PA	TX	UDP18	NY	MI
TCP19	NJ	GA	UDP19	NE	IL
TCP20[e]	CO	NY	UDP20	UT	NE

[a] MI-UT-CO-TX path is used
[b] PA-NJ-MI path is used
[c] PA-GA-TX-CA2 path is used
[d] TX-CA2-WA-IL path is used
[e] CO-UT-MI-NY path is used
[f] NE-CO-TX-CA2 path is used
[g] NE-CO-UT-CA1 path is used
[h] IL-WA-CA1-UT path is used

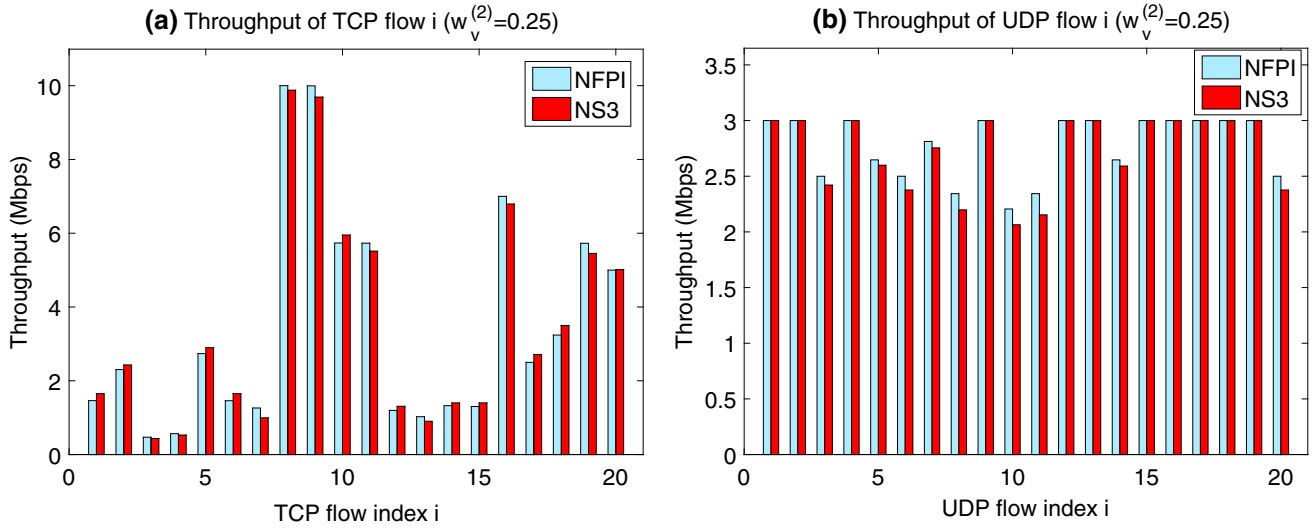


Fig. 4 **a** TCP and **b** UDP per-flow throughputs when $w_v^{(2)} = 0.25,\ \forall v \in V$

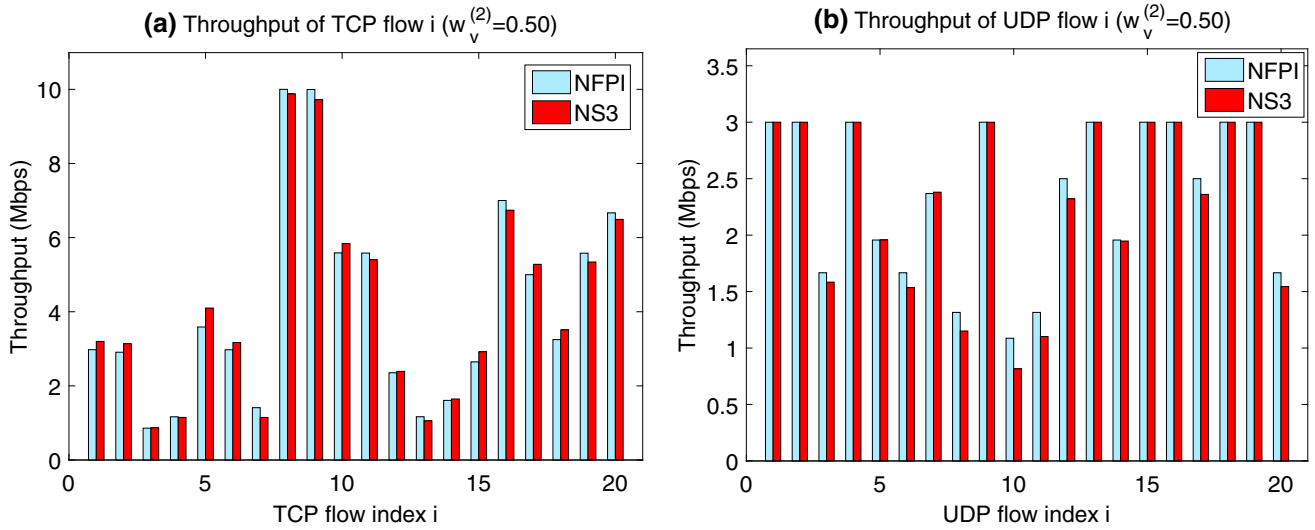


Fig. 5 **a** TCP and **b** UDP per-flow throughputs when $w_v^{(2)} = 0.50,\ \forall v \in V$

convergence for each outer loop for three different $w_v^{(1)}$ values and also the no isolation case are presented in Table 7.

We have the following observations. Although the general throughput figures are captured for all scenarios we tested for both TCP and UDP, approximation errors are inevitable caused by several factors including the shortcoming of the TCP send rate formula in describing each and every detail of the TCP protocol, overall carried ACK traffic which is neglected in the analysis, the fluid analysis framework that does not perfectly describe the packet-by-packet transmission aspect of each link, etc. When no isolation takes place between TCP and UDP flows, some of the TCP flows (that contend with UDP flows on their way) get hampered, for example TCP flows 3 and 4. Such starvation of TCP flows can be avoided with per-class queuing and the choice of a relatively larger scheduling weight for class 2. Note that this starvation mitigation effect is almost perfectly captured by NFPI for these two flows. We also observe that the outer loop takes at most a few iterations to converge. The inner loop corresponding to class 2 flows is relatively slower requiring more iterations.

4.3 Example III

In the third example, we study the convergence time of the NFPI algorithm implemented with MATLAB using a ring network topology with N nodes. Computations for NFPI are carried out on a mobile workstation equipped with Intel Quad Core i7-4712HQ processor and we used MATLAB *tic* and *toc* commands to obtain the computational run-times for

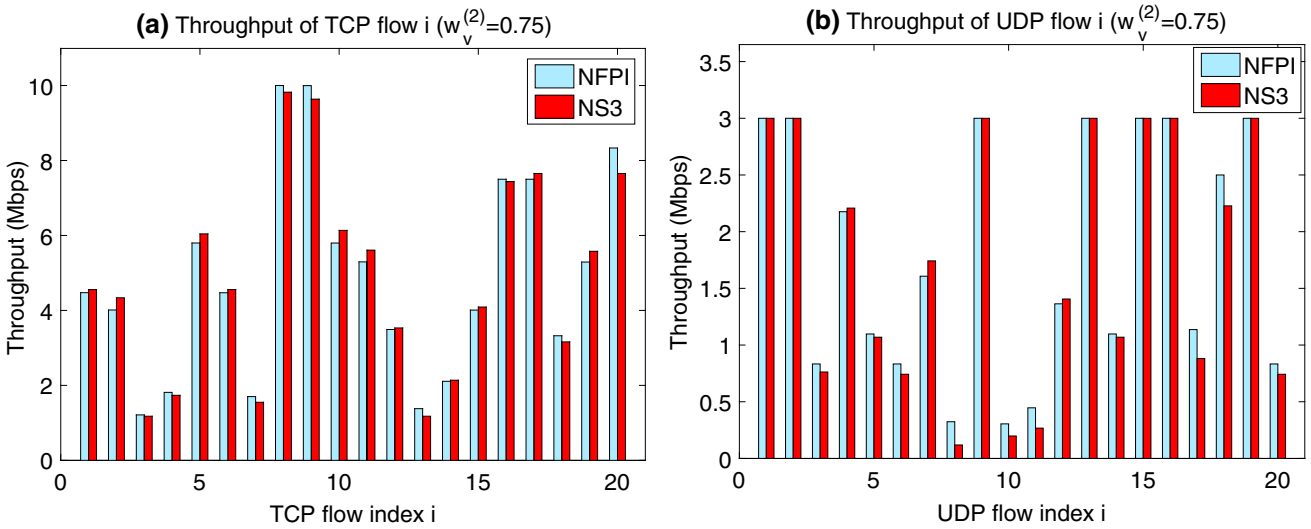


Fig. 6 **a** TCP and **b** UDP per-flow throughputs when $w_v^{(2)} = 0.75,\ \forall v \in V$

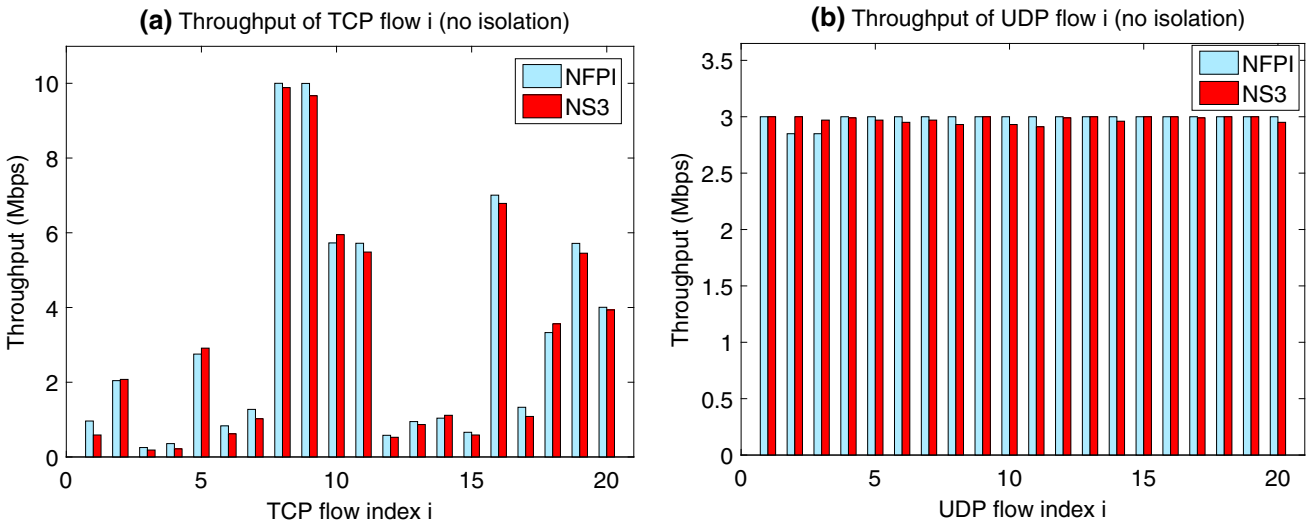


Fig. 7 **a** TCP and **b** UDP per-flow throughputs when no isolation is employed between TCP and UDP

NFPI. Link capacities and propagation delays are set to 100 Mbps and 1 ms, respectively. The parameter $w_v^{(1)}$ is set to 0.5 for all links. Other parameters are the same as in the first two examples. Fig. 8 depicts the behavior of the convergence time of NFPI with respect to the number of nodes N in the ring for three different UDP send rates. In order to fix the UDP send rates, we define a new parameter L_v as follows:

$$L_v = \sum_{i_U^{(1)} \in S_U(v,1)} T(i_U^{(1)}). \tag{15}$$

Note that the total UDP demand on link v sums up to L_v and all links in a ring network with an odd number of nodes have the same L_v value. We obtain numerical results for $L_v = 30, 50, 70$ Mbps.

To demonstrate the possible reasons of the differences between convergence times for different UDP send rates, the number of inner loop iterations required at each turn of the outer loop for five different UDP send rates are given in Table 8 for the case N is set to 13. We have the following observations related to NFPI computational run-times. As expected, the convergence time of the proposed algorithm is quadratic in the number of nodes N since there are $\mathcal{O}(N^2)$ flows in this example. Even for a 11-node ring network where we have 110 TCP and 110 UDP flows, NFPI converges within 45 seconds in all studied cases. Moreover, we observe that convergence times appear to be slightly dependent on the amount of overall UDP demand.

Table 6 Mean and maximum per-flow percentage throughput errors while using NFPI in comparison with ns-3 simulations

$w_v^{(1)}$	0.25	0.50	0.75	TCP/UDP in one queue
Mean % error (TCP)	4.40	5.58	6.83	11.95
Max. % error (TCP)	14.63	18.59	21.02	39.33
Mean % error (UDP)	11.06	4.42	2.33	1.29
Max. % error (UDP)	63.38	24.72	8.44	4.93

Table 7 Number of inner loop iterations required until convergence for each outer loop for three different $w_v^{(1)}$ values and also for the no isolation case

$w_v^{(1)}$:	0.25	0.5	0.75	No isolation
Outer 1				
Class 1:	6	5	4	–
Class 2:	7	6	7	25
Outer 2				
Class 1:	1	1	1	–
Class 2:	29	14	20	1
Outer 3				
Class 1:	4	3	3	–
Class 2:	32	67	32	–
Outer 4				
Class 1:	1	1	1	–
Class 2:	132	35	1	–
Outer 5				
Class 1:	2	2	––	–
Class 2:	5	17	––	–
Outer 6				
Class 1:	1	1	––	–
Class 2:	1	1	––	–
Total				
Outer:	6	6	4	2
Class 1:	15	13	9	–
Class 2:	226	140	60	26

4.4 Example IV

In the final example, we study the effect of the number of TCP and UDP flows on the convergence time for a different topology given in Fig. 9 which is a 10-node Italian network [30]. Inter-nodal distances for this topology is obtained from the study in [2]. We also compare the convergence time and throughput performance of NFPI with the *Static Sharing* (SS) approximation for which the outer loop runs only

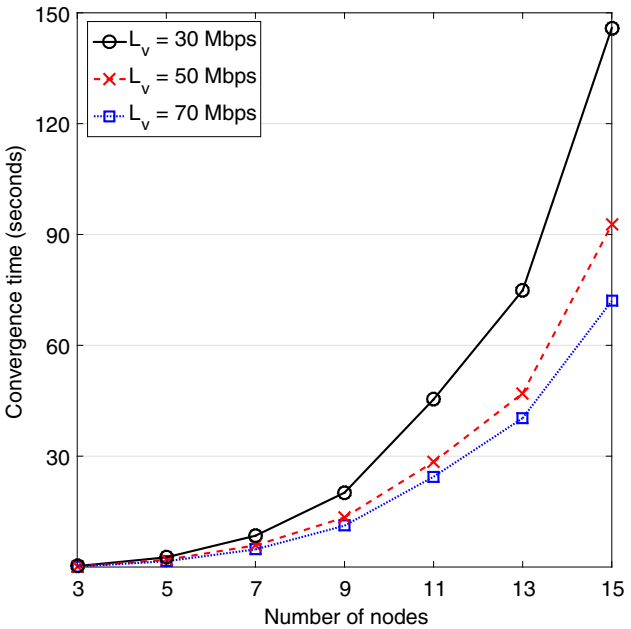


Fig. 8 Convergence time of the algorithm in a ring network as a function of the number of nodes

Table 8 Number of inner loop iterations for each turn of the outer loop for five different UDP send rates in the 13-node ring network

L_v:	30	40	50	60	70
Outer 1					
Class 1:	67	69	71	45	49
Class 2:	11	11	11	11	11
Outer 2					
Class 1:	1	1	1	1	1
Class 2:	10	10	2	1	1
Outer 3					
Class 1:	1	1	1	–	–
Class 2:	1	1	1	–	–
Total					
Outer:	3	3	3	2	2
Class 1:	69	71	73	46	50
Class 2:	22	22	14	12	12

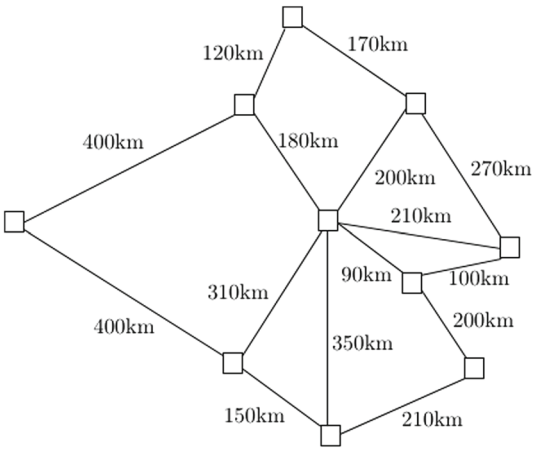


Fig. 9 10-node Italian network topology of Example IV

Table 9 The computation times and the mean UDP and TCP throughput for the 10-node Italian network for three different values of α and K

K	α	T(sec)		Mean throughput (Mbps)			
		NFPI	SS	TCP		UDP	
				NFPI	SS	NFPI	SS
10	0.3	6.41	2.15	22.89	13.90	20.05	18.17
	0.5	33.69	4.81	34.20	19.75	14.55	13.65
	0.7	44.25	6.01	45.17	25.00	8.81	8.62
50	0.3	46.28	5.89	51.42	41.47	64.29	52.36
	0.5	54.00	6.67	75.82	55.25	48.59	43.86
	0.7	57.36	8.14	97.16	63.06	34.34	32.65
250	0.3	77.21	19.95	87.65	85.70	86.06	76.72
	0.5	86.40	23.43	102.56	97.06	80.00	76.07
	0.7	107.41	26.82	116.87	101.29	66.85	65.54

once and the capacity sharing step is skipped that is used at the end of each outer loop. The SS scheme refers to one in which the TCP and UDP flows are completely isolated using static time sharing and methods to compute the TCP and UDP throughputs are already available in the literature. For this study, we construct fifty instances at each of which the source-destination pairs are assigned randomly and NFPI and SS results are computed. All UDP send rates are fixed to 3 Mbps. The link capacities are set to 10 Mbps for all links, and the propagation delays of the links are obtained from the inter-nodal distances. Scheduling weights $w_v^{(i)}$, $i = 1, 2$ are fixed to 0.5. Let K denote the total number of flows in the network. We assign αK of these flows to TCP and the remaining $(1 - \alpha)K$ to UDP, for any $\alpha \geq 0$ satisfying $\alpha K \in \mathbb{Z}$. Mean convergence time T and mean per-class throughput for nine different cases corresponding to $\alpha = 0.3, 0.5, 0.7$ and $K = 10, 50, 250$ are presented in Table 9. We observe that the throughput performances of both TCP and UDP flows are improved because of the dynamic capacity sharing in CBWFQ compared to SS. However, NFPI requires much longer convergence times than SS due to the few turns the outer loop requires until convergence. Note that SS corresponds to the outcome of one single turn of the outer loop. When the number of TCP flows increases, the convergence times also appear to increase stemming from the fact that the inner loop for TCP flows generally tends to require more iterations than the other inner loop for class 1. When the topology is fixed, the required computation time increases with increased number of flows but at a less than linear rate.

5 Conclusions

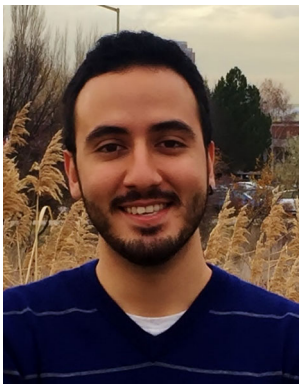
We propose a nested fixed-point iterative method for finding the throughput of persistent UDP and TCP flows in a network of routers supporting per-class queuing with two classes when TCP ACK traffic is given strict priority over all other types of traffic. With ns-3 simulations, we have been able to demonstrate the validity of the proposed analysis method for different per-class scheduling weights. The findings of this research can be used for provisioning Diffserv links in IP networks. Future work consists of modeling ACK traffic more accurately extending to scenarios where TCP ACK traffic does not possess strict priority, and also the study of cases with more than two traffic classes for improved traffic management.

Acknowledgements We would like to thank Mr. Gokhan Calis for the ns-3 code he produced as part of his MS thesis which is then used to validate the analytical model of the paper.

References

1. Abouzeid, A. A., & Roy, S. (2002). Modeling random early detection in a differentiated services network. *Computer Networks*, *40*(4), 537–556.
2. Ali, M. (2005). Routing of 40-gb/s streams in wavelength-routed heterogeneous optical networks. *IEEE Journal on Selected Areas in Communications*, *23*(8), 1632–1642.
3. Alvarez-Flores, E., Ramos-Munoz, J., Ameigeiras, P., & Lopez-Soler, J. (2011). Selective packet dropping for VoIP and TCP flows. *Telecommunication Systems*, *46*(1), 1–16.
4. Athuraliya, S., Low, S., Li, V., & Yin, Q. (2001). REM: Active queue management. *IEEE Network*, *15*(3), 48–53.
5. Betker, A., Gerlach, C., Hülsermann, R., Jäger, M., Barry, M., Bodamer, S., Späth, J., Gauger, C., & Köhn, M. (2003). Reference transport network scenarios. http://www.ikr.uni-stuttgart.de/INDSimLib/Usage/
6. Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., & Weiss, W. (1998). An architecture for differentiated service. RFC 2475 (Informational). http://www.ietf.org/rfc/rfc2475.txt. Updated by RFC 3260

7. Boney, J. (2005). *Cisco IOS in a nutshell* (2nd ed.). Sebastopol: O'Reilly.
8. Braden, B., Clark, D., Crowcroft, J., Davie, B., Deering, S., Estrin, D., Floyd, S., Jacobson, V., Minshall, G., Partridge, C., Peterson, L., Ramakrishnan, K., Shenker, S., Wroclawski, J., & Zhang, L. (1998). Recommendations on queue management and congestion avoidance in the Internet. RFC 2309
9. Bu, T., & Towsley, D. (2001). Fixed point approximations for TCP behavior in an AQM network. *SIGMETRICS Performance Evaluation Review*, *29*(1), 216–225.
10. Chang, H. Y., Lee, W. T., & Wei, H. W. (2014). A novel protocol UDCP for improving fairness and maintaining the high-speed rate of UDP. *Intelligent Information Hiding and Multimedia Signal Processing (IIHMSP)*, *2014*, 698–701.
11. Chen, J. C. (2003). Dijkstra's shortest path algorithm. *Formalized Mathematics*, *11*(3), 237–247.
12. Chinoy, B., & Braun, H.W. (1992). The national science foundation network. Technical Report GA-A21029, SDSC
13. Cisco: QoS: Congestion Management Configuration Guide, Cisco IOS Release 15 (2013)
14. Farzaneh, N., Monsefi, R., Yaghmaee, M., & Mohajerzadeh, A. (2013). A novel congestion control protocol with AQM support for IP-based networks. *Telecommunication Systems*, *52*(1), 229–244.
15. Floyd, S. (2000). Recommendation on using the gentle_ variant of RED. URL: http://www.icir.org/floyd/red/gentle.html
16. Floyd, S., & Fall, K. (1999). Promoting the use of end-to-end congestion control in the Internet. *IEEE/ACM Transactions on Networking*, *7*(4), 458–472.
17. Floyd, S., & Jacobson, V. (1993). Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, *1*(4), 397–413.
18. Garetto, M., Cigno, R. L., Meo, M., & Marsan, M. A. (2004). Modeling short-lived TCP connections with open multiclass queuing networks. *Computer Networks*, *44*(2), 153–176.
19. Gibbens, R.J., Sargood, S.K., Eijl, C.V., Kelly, F.P., Azmoodeh, H., Macfadyen, R.N., & Macfadyen, N.W. (2000). Fixed-point models for the end-to-end performance analysis of IP networks. In: 13th ITC specialist seminar: IP traffic measurement, modeling and management. Monterey, CA
20. Grochla, K. (2008). Simulation comparison of active queue management algorithms in TCP/IP networks. *Telecommunication Systems*, *39*(2), 131–136.
21. Hashem, E.S. (1989). Analysis of random drop for gateway congestion control. Ph.D. thesis, Massachusetts Institute of Technology, Cambridge
22. Hassan, M., & Jain, R. (2004). *High performance TCP/IP networking: Concepts, issues, and solutions*. London: Pearson.
23. Held, G. (2002). *Quality of service in a Cisco networking environment*. Hoboken: Wiley.
24. Ji, L., Arvanitis, T., & Woolley, S. (2003). Fair weighted round robin scheduling scheme for DiffServ networks. *Electronics Letters*, *39*(3), 333–335.
25. Kahe, G., Jahangir, A., & Ebrahimi, B. (2014). AQM controller design for TCP networks based on a new control strategy. *Telecommunication Systems*, *57*(4), 295–311.
26. Kalampoukas, L., Varma, A., & Ramakrishnan, K. (1998). Improving TCP throughput over two-way asymmetric links: Analysis and solutions. *ACM SIGMETRICS Performance Evaluation Review*, *26*(1), 78–89.
27. Kooij, R., van der Mei, R., & Yang, R. (2010). TCP and WEB browsing performance in case of bi-directional packet loss. *Computer Communications*, *33*, S50–S57.
28. Lee, D., Carpenter, B. E., & Brownlee, N. (2010). Media streaming observations: Trends in UDP to TCP ratio. *International Journal on Advances in Systems and Measurements*, *3*(3 & 4), 147–162.
29. Mahdavi, J., & Floyd, S. (1997). TCP-friendly unicast rate-based flow control.Technical note sent to the end2end-interest mailing list
30. Mantelet, G., Cassidy, A., Tremblay, C., Plant, D. V., Littlewood, P., & Bélanger, M. P. (2013). Establishment of dynamic lightpaths in filterless optical networks. *Journal of Optical Communications and Networking*, *5*(9), 1057–1065.
31. Mathis, M., Semke, J., Mahdavi, J., & Ott, T. (1997). The macroscopic behavior of the TCP congestion avoidance algorithm. *SIGCOMM Computer Communication Review*, *27*(3), 67–82.
32. Misra, V., Gong, W. B., & Towsley, D. (2000). Fluid-based analysis of a network of aqm routers supporting tcp flows with an application to red. *SIGCOMM Computer Communication Review*, *30*(4), 151–160.
33. Na, Z., Guo, Q., Gao, Z., Zhen, J., & Wang, C. (2012). A novel adaptive traffic prediction AQM algorithm. *Telecommunication Systems*, *49*(1), 149–160.
34. Nagle, J. (1995). Congestion control in IP/TCP internetworks. *ACM SIGCOMM Computer Communication Review*, *25*(1), 61–65.
35. Ott, T.J., Kemperman, J.H.B., & Mathis, M. (1996). The stationary behavior of ideal TCP congestion avoidance
36. Padhye, J., Firoiu, V., Towsley, D., & Kurose, J. (1998). Modeling TCP throughput: A simple model and its empirical validation. *SIGCOMM Computer Communication Review*, *28*(4), 303–314.
37. Popa, L., Yalagandula, P., Banerjee, S., Mogul, J. C., Turner, Y., & Santos, J. R. (2013). ElasticSwitch: Practical work-conserving bandwidth guarantees for cloud computing. *SIGCOMM Computer Communication Review*, *43*(4), 351–362.
38. Ramroop, S. (2011). A diffserv model for the NS-3 simulator. http://www.eng.uwi.tt/depts/elec/staff/rvadams/sramroop/
39. Ranjan, P., La, R.J., & Abed, E.H. (2002). Bifurcations of TCP and UDP traffic under RED. Proc. MED 2002
40. Rodrigues, H., Santos, J. R., Turner, Y., Soares, P., & Guedes, D. (2011). Gatekeeper: Supporting bandwidth guarantees for multi-tenant datacenter networks. *Proceedings of the 3rd Conference on I/O Virtualization* (pp. 6–6)., WIOV'11 Berkeley, CA: USENIX Association.
41. Sharma, V., Virtamo, J., & Lassila, P. (2002). Performance analysis of the random early detection algorithm. *Probability in the Engineering and Informational Sciences*, *16*, 367–388.
42. Shreedhar, M., & Varghese, G. (1995). Efficient fair queueing using deficit round robin. *SIGCOMM Computer Communication Review*, *25*(4), 231–242.
43. Xiong, N., Pan, Y., Jia, X., Park, J. H., & Li, Y. (2009). Design and analysis of a self-tuning feedback controller for the internet. *Computer Networks*, *53*(11), 1784–1797.
44. Yilmaz, S., & Matta, I. (2001). On class-based isolation of UDP, short-lived and long-lived TCP flows. In: Proceedings of ninth international symposium on modeling, analysis and simulation of computer and telecommunication systems, 2001. pp. 415–422
45. Zhang, H., Towsley, D., Hollot, C. V., & Misra, V. (2003). A self-tuning structure for adaptation in TCP/AQM networks. *SIGMETRICS Performance Evaluation Review*, *31*(1), 302–303.



Caglar Tunc received the B.S. degree from Bilkent University in 2013 in electrical and electronics engineering. He is currently pursuing the M.S. degree in the same department. His current research interests are in energy harvesting wireless communications, green communications, stochastic modeling and algorithmic aspects of computer and communication systems.

Nail Akar received his B.S. degree from Middle East Technical University, Turkey, in 1987 and M.S. and Ph.D. degrees from Bilkent University, Ankara, Turkey, in 1989 and 1994, respectively, all in electrical and electronics engineering. From 1994 to 1996, he was a visiting scholar and a visiting assistant professor in the Computer Science Telecommunications program at the University of Missouri - Kansas City, USA. He joined the Technology Planning and Integration group at Long Distance Division, Sprint, Overland Park, Kansas, in 1996, where he held a senior member of technical staff position from 1999 to 2000. Since 2000, he has been with Bilkent University, Turkey, currently as a Professor, at the Electrical and Electronics Engineering Department. He visited the School of Computing, University of Missouri - Kansas City, as a Fulbright scholar in 2010 for a period of six months. His current research interests include performance analysis of computer and communication systems and networks, performance evaluation tools and methodologies, design and engineering of optical and wireless networks, queuing systems, and resource management.