Proceedings Paper:

Aina, J, Mhamdi, L and Hamdi, H (2019) F-DCTCP: Fair Congestion Control for SDN-Based Data Center Networks. In: 2019 International Symposium on Networks, Computers and Communications (ISNCC). 2019 International Symposium on Networks, Computers and Communications (ISNCC), 18-20 Jun 2019, Istanbul, Turkey. IEEE . ISBN 9781728112442

https://doi.org/10.1109/isncc.2019.8909171

F-DCTCP: Fair Congestion Control for SDN-Based Data Center Networks

Jonathan Aina and Lotfi Mhamdi
School of Electronic and Electrical Engineering
The University of Leeds
Leeds, UK
L.Mhamdi@leeds.ac.uk

Hédi HAMDI
ESCT-Tunis
University of Manouba
Manouba, Tunisia
hamdi.h@gmail.com

***Abstract*—Network congestion control and management has long been a major issue in providing low-latency, high throughput and high link-utilisation. In particular, Data Center Network (DCN) environments, often dominated by partition-aggregate workloads, suffer performance collapse due to TCP Incast caused by inadequate congestion control parameters. This paper proposes a step in solving this problem. In particular, we describe a novel framework to overcome and control congestion in DCNs based on Software Defined Networking (SDN). We propose a native SDN-based congestion control mechanism, termed Fair Data Center TCP (F-DCTCP). As we shall see, the experimental results show that F-DCTCP outperforms all previous proposals by providing the best combined overall performance in terms of throughout, fairness and flow completion times, making it highly attractive for next generation networks.**

***Index Terms*—SDN, TCP, Congestion, DCTCP, FDCTCP**

I. Introduction

Traditional approaches to manage network resources are based on manual configuration of proprietary devices; which are cumbersome, error-prone and fail to fully utilize the capacity of physical network infrastructures. They also lack flexibility as they tend to be optimised only in a specific subnet of a larger network, this limitation significantly reduces the overall performance of the network [1]. Recently, SDN (Software Defined Network) [2] [3] [4] [6] has emerged as a promising solution for a more robust and efficient means of Network management. SDN is characterized by its two distinguished features; decoupling the control plane from the data plane in switching devices and allowing network control to be directly programmable. With respect to this advancement, SDN has proved largely successful in solving most of the problems and limitations in traditional networks [7].

Conversely, congestion control is a perplexing issue in the area of network management. As traditional Transmission Control Protocol (TCP) congestion control mechanisms are designed to operate in diverse networks, their control measures are based on estimation of network traffic patterns; as such they are menaced with inconsistencies resulting in the throttling of network performance [8]. In light of this problem, SDNs inherent capability of a centralised control and global view of the entire network is one which brings a glimpse of hope to the eradication of this dominant predicament. The level of abstraction added by SDN to the functionalities of the network equipment (switches, routers,...) and the possibility to manage them all globally requires rethinking congestion control and management paradigms in order to develop a revolutionary congestion control mechanism and also provide performance enhancing capabilities of the inherent features of SDN.

Motivated by the lack of adequate congestion control mechanism able to keep up with current and next generation networks traffic, this paper proposes a novel SDN-based congestion control mechanism that addresses the above-mentioned shortcomings. In particular, the main contributions of this paper are as follows:

- We propose a novel and efficient SDN-based congestion control mechanism, termed Fair Data Center TCP (F-DCTCP), to overcome and control congestion in DCNs. F-DCTCP is natively designed for SDN-based controllers, allowing for a new class of pure SDN-based congestion control solutions.
- We demonstrate that F-DCTCP outperforms all previous proposals by providing the best *combined-overall-performance* in terms of throughout, fairness and flow completion times respectively, making it highly attractive for next generation networks.

The remainder of this paper is structured as follows: Section 2 states the problem in detail with a brief study of two scenarios from our performance evaluation of traditional TCP congestion approaches. Section 3 gives a succinct critical review of similar works in the area of adapting TCP to different domains. Section 4 describes our approach in details and expatiates on how the envisaged result can be achieved by describing the algorithm in stages. Section 5 presents a performance evaluation and analysis of our solution; further comparing this with existing works. Section 6 concludes the work and gives recommendation for future work.

II. Problem Statement

While the demand for internet services continues to grow enormously, the resources to provide them are greatly limited in terms of cost, processing power, capacity and performance [4]. The TCP is the main Internet transport protocol that carries 90% of the Internet traffic [5]. Hence when incoming requests increases aggressively, in a network, to a point where it eclipses the outbound capacity of a router, a network congestion occurs. This causes delay, packet loss, low throughput and in worst cases result to network collapse. Hence, congestion control is paramount in any network [9]. TCP congestion control has been the standardised approach to solving this problem in traditional networks. It works by limiting the amount of data sent by a host every round-trip time (RTT), thus keeping the network stable. However, due to the heterogeneity of the internet and limited information on the instantaneous state of a network, TCP is designed to operate based on allusive network statistics; as a result of this and the diverse nature of different networks, a performance instability occurs whereby TCP intermittently increases and decreases its congestion window size to cater for changes in traffic. This limitation seriously degrades the throughput and utilisation of individual networks irrespective of the capacity of that network [10]. We will describe this problem in detail with a brief study of two scenarios from our performance evaluation of Traditional TCP congestion approach.

A. *Case 1: TCP in Long Fat Networks*

We performed a TCP throughput test in a high-speed network with long delay, known as long fat network where four flows are sharing a 250Mbps bottleneck link. Fig. 1 shows the result of the test in terms of achievable throughput. It was realised that regardless of the high capacity dedicated link (250Mbps) that was put in place for this network connection, its throughput was 75% lower than the theoretical performance of TCP, this is certainly a very poor performance and under-utilization of resources. This fact is strengthened by the result represented in the second graph of Fig. 2, which shows a maximum utilisation of 18% by TCP CUBIC [11] and just about 12% for the default TCP Reno [12].

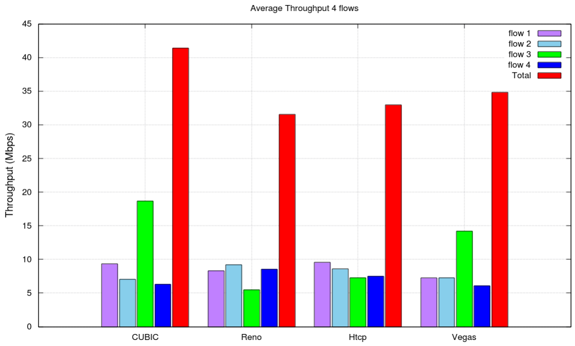

Fig. 1. Average TCP Throughput Analysis of four TCP flows.

B. *Case 2: TCP Incast*

We performed an another throughput analysis of TCP congestion control in Data Centre Networks, to determine the suitability of each TCP variant. Fig. 3 shows that there was a total throughput collapse of 88% witnessed while using different TCP congestion control variants; only the newly developed DCTCP was able to achieve higher throughput. However, its performance in this scenario remains unsatisfactory with respect to fairness among competing flows (flow 4 is starved using DCTCP). This also establishes that if the default TCP congestion control was used, there will be a significant reduction in throughput and a network collapse.

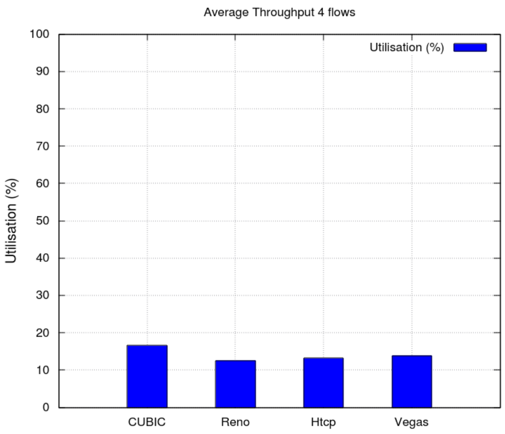

Fig. 2. Link utilisation of four TCP flow variants.

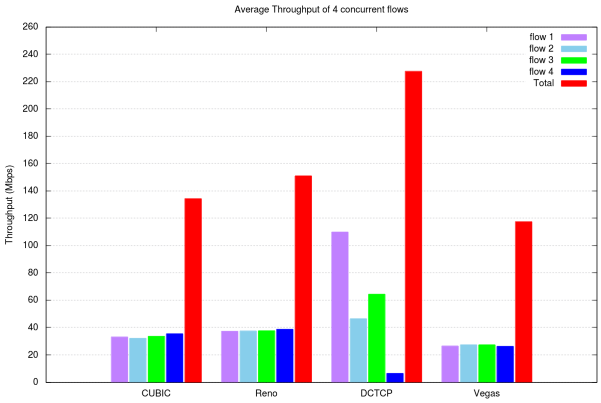

Fig. 3. Incast Throughput analysis of TCP variants.

From the above study, we clearly see the variations in performance of TCP congestion control algorithms in different network domains and we can appreciate that there is a need to adopt or optimise different TCP congestion algorithms in different networks. As SDN becomes increasingly popular, with its deployment in several high-speed networks [10], there exists two pressing needs:

- To adapt traditional TCP congestion control to SDN.
- To optimise these algorithms such that we can explore the features of SDN to achieve an optimal congestion control mechanism.

III. Related Work

Aside from the core (traditional) TCP congestion control mechanisms and its variants; Following the discovery of the TCP incast problem by Nagle et al. [13] a number of propositions have been made. In this section, we give a brief summary of relevant solutions based on the implementation domain. These domains can be grouped into these areas: Application based solutions, switch assisted solution, window based solution and SDN based solutions. The works are described below:

A. Application Based Solutions

Solutions in this category are based on utilising application information (e.g. knowing the amount of parallel connections to different servers), to ensure the avoidance of the scenarios necessary for incast to occur. The most notable works in this aspect include [14] and [15]. The practicability and stability of this solution becomes a major problem as it is difficult to get a timely global knowledge of application information and ensuring synchronisation in all connections. This makes the solution difficult to implement and deploy.

B. Switch Assisted Solutions

These solutions propose a modification to switches to predict the occurrence of TCP incast and pro-actively avoid it. A notable implementation of this is IQM; which works by setting the transfer rates of elephant flows to 1 MSS upon detection of TCP incast. Aside from the unfairness of this approach a major drawback is in the need to modify the switches hardware to support this functionality.

C. SDN-Based Solutions

A number of solutions for congestion control in SDN have been proposed; the worthy ones are: OpenTCP [16] and SDTCP [17]. OpenTCP is not a new solution but just an SDN application that enables network operator implement different TCP variants in advance. For example, it allows the selection of either CUBIC or Vegas [18] as the default congestion control or tuning of pre-determined congestion parameters, which also lacks accuracy and timeliness. SDTCP [13], proposes an incast congestion control mechanism that utilises the SDN functionality and implements a network side congestion control. The mechanism involves prioritising bursty flows over elephant flows; subsequently updating the flow table to decrease the sending rate of elephant flows. This solution lacks fairness as it does not assure good throughput for elephant flows. Another major problem however is the proposed detection of the flows which is fairly inaccurate estimation, this could potentially result to a case of reduction of multiple flow sending rate which ultimately leads to poor utilisation.

D. Window Based Solutions

These solutions involve utilising various parameters and characteristics of Traditional congestion mechanism to adjust the congestion window of the sender or the receive window of the receiver as a means to proactively reduce transfer rate before congestion occurs. Two major approaches in this domain are DCTCP [19] and ICTCP [20]. DCTCP provides a congestion control scheme that utilizes the ECN feedback from congested switches to help the sender adjust the congestion window based on the network congestion; Whereas ICTCP estimates available bandwidth and per-flow throughput at the receivers gateway to adjust the senders transfer rate. These two implementations curbs timeouts and achieve fairly better throughput for TCP incast traffic. However, ICTCP is ineffective if the bottleneck is not at the receivers switch. This makes it a half-built solution that will prove ineffectual a large amount of the time; the incast problem could be evident at any switch in the network. DCTCP, on the other hand, is to be regarded as a well-designed approach which exhibits desirable performance and control characteristics when TCP incast is at its infant. However, DCTCP exhibits poor stability and convergence as a result of inaccuracies in its congestion detection and notification mechanism [17].

IV. F-DCTCP: Fair SDN-Based DCTCP

Having described existing solutions and their major drawbacks; in this section, we describe the idea of our F-DCTCP SDN based solution and explain the working mechanism clearly. Our design is motivated by the SDN and window based solution approach; utilising the best of both worlds to achieve an optimal all-purpose congestion control. Based on an extensive performance evaluation of selected TCP congestion control algorithms, the goal of F-DCTCP is to improve upon the implementation of DCTCP using the centralised control advantage of an SDN controller, to achieve a much accurate detection of congestion and fair control of all senders transfer rates.

A. F-DCTCP system design

The basic idea of F-DCTCP is to monitor the possibility of congestion occurrence in all the switches under the domain of a controller and proactively reduce the sending rate of participating flows to avoid it. We believe that at a time of imminent congestion, it is only optimal to limit all flow rate fairly to a maximum that prevents overloading of the switches buffers.

In doing this, we developed 3 mechanisms; Congestion Detection, Fair Adaptive Transfer Rating and Enforcing Adaptive Transfer Rate. Fig. 4 represents the interaction between the 3 modules our system.

Our robust congestion detection is achieved by monitoring the switch buffers' queue occupancy for medium congestion monitoring and receipt of multiple *SYN* request from the same host to different servers for incast mitigation. Whenever we discover either of this, we trigger the congestion avoidance state; which is implemented as a max-min fair allocation of the switches non-congestible bandwidth. The optimal rate is enforced by rewriting the receive window of the *ACK* packet from the receiving hosts, which we regard as the network bottleneck receive window (*NetWin*). Subsequently, this rate is adapted until the impending congestion is annulled. However, in a high-speed oversubscribed network, our solution ensures that the offered load at any time is exactly what the network can handle at that time, thus keeping the network in a stable state.

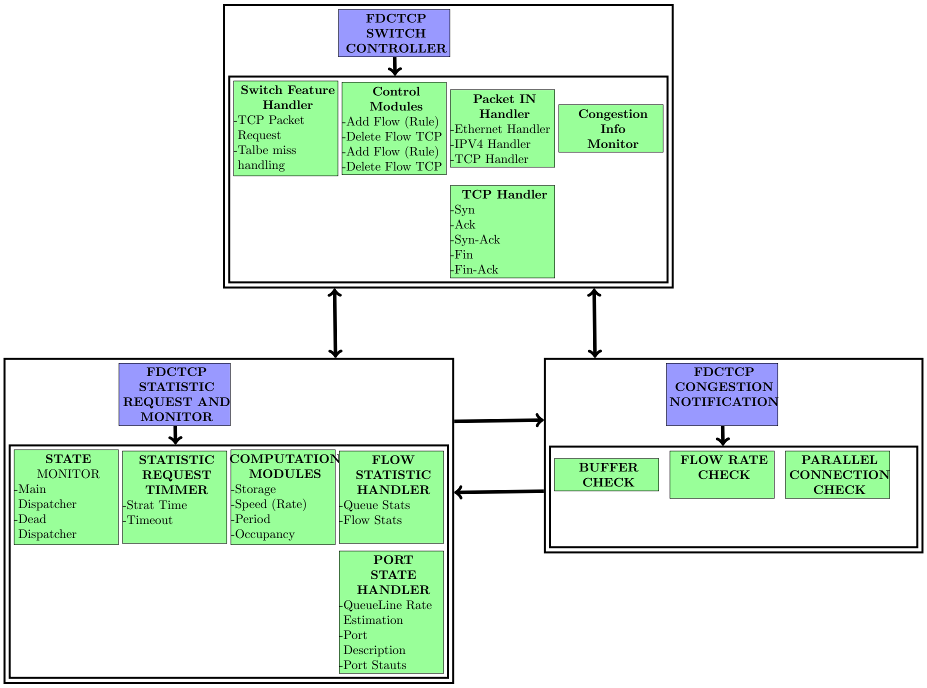

Fig. 4. System Architecture Diagram showing the interaction of the different modules

B. F-DCTCP Blocks

We give a detailed description of the three stages of our proposed solution below.

1) Congestion Detection: We designed a network congestion detection module at the OF-switch to monitor two congestion scenarios: monitoring Queue occupancy for simple-medium congestion and monitoring parallel connections for Complicated Congestion (notably TCP incast). A congestion flag is defined in this module, which takes a value **'1'** signifying imminent congestion and **'0'** signifying no congestion. The congestion avoidance solutions are triggered when the value of this flag is **'1'** and a return to normal state is triggered when the value is **'0'**.

a) Queue Occupancy Monitoring: The queue occupancy section of the congestion detection module employs the openFlow statistics request message to get real time information of the switch activities. A stats query is issued to all output ports every **1ns** and we provide a means for tuning the interval for different network scenarios. In our queue monitoring design, we consider all the contributing factors to the over filling of a switch buffer; the current queue capacity, the sending rate of flows over the port and the bandwidth of the port. This enables us to characterise the arrival rate, service rate and capacity of the switch. We established that the outgoing port is under the danger of congestion if the queue length is beyond a defined threshold and the packet arrival rate is eclipsing the service rate of the port.

b) Parallel connections monitoring: The most crucial part of the detection module is monitoring parallel connections from a single client to multiple servers, which is a major indication of the likely occurrence of TCP incast. We achieve this by keeping track of *SYN-ACK* packets to record the source and destination address pair of a TCP connection. If multiple *SYN-ACK* packets are detected to have come from a single host to different senders, the number of senders (destination address of *SYN-ACK* packets) is recorded as the number of parallel connection to that host, hence the Second stage of our Algorithm is triggered when any one host has N parallel connections which is more than 30% of the total number of flows over an outgoing port. This condition only holds for $N > 2$ and not equal to the total number of flows.

2) Fair Adaptive Transfer Rate: This stage is triggered when congestion is imminent, which is indicated by a congestion flag set to **1**. In our Fair rate module we implemented the Max-Min fairness algorithm to compute the optimal transfer rate for all the flows over the congested port. The threshold bbthresh for all the flows is computed as follow:

$$bbthresh = (\frac{free\ buffer\ space}{total\ buffer\ size}) \times Available\ bandwidth$$

Using max-min fairness, we considered the congestion window of each flow as the requested rate and a fair allocation of the bottleneck threshold capacity is made on a per flow basis.

3) Enforcing Adaptive Transfer Rate: In order to enforce the adaptive transfer rate of participating flows, we modify the advertised received window of TCP ACK packets from the receivers at the OpenFlow switch. The basis of this approach is in the congestion window implementation in TCP; the implementation limits a TCP senders send window to the minimum between the senders congestion window and the receivers receive window. The formula is given below:

$$swnd = min(cwnd, rcvwnd)$$

This implementation forces the participating senders to always transfer at a rate which is less or equal to our predetermined adaptive transfer rate. Ensuring this is a major performance achievement in our implementation as this rate allows for maximal fair allocation of the available resource of the switches in the network.

4) Ensuring stability: To ensure the stability of our solution, the RCV window of the ACK packet is only set to the adaptive transfer rate if the current rcvwin $>$ adaptive transfer rate; otherwise it is left unchanged. This little modification is crucial as it ensures that the final transfer rate of witch the participating hosts are allowed to utilise will be the minimum non-congestible rate of the entire path, which is equivalent to the transfer rate allowed on the most congested switch. This achieves a network wide congestion control that will always achieve optimal performance regardless of the traffic characteristics of the network.

5) Recovering: The recovery process is simply a fall back to the congestion monitoring state, which occurs when the congestion flag is set to **0**. This state is closely linked to the congestion detection state. As at this state, an ongoing monitoring of the switch statistics continues until a possibility of congestion is noticed. However, in certain cases a network administrator may prefer a re-iteration of the Fair adaptive transfer rate stage; such that an optimal and fair rate keeps getting allocated to participating flows, hence forever preventing congestion while allowing for the maximum utilisation of the available resources. An external parameter is added to enable this functionality.

6) Alternative Implementation - Enforcing Adaptive Transfer Rate: As a result of possible limitations in the widespread deployment of a scheme that involves the receiver window rewriting due to OpenFlow protocol ver 1 not supporting the SetField functionality. We propose another alternative. But it has to be mentioned that the option should only be used if the OpenFlow protocol used by the switch is version 1 as it is an aggressive way of enforcing the controlled rate. It has to be stated further that this add-on simple ensures fairness hence increasing average flow completion time, but provides no additional performance improvement to throughput.

a) Preventive Thresholding: The alternative approach involves setting the Adaptive Transfer rate as an implicit congestion threshold for participating flows in the OpenFlow switch. Setting a threshold involves using the metering functionality of the OpenFlow protocol. Metering involves setting a maximum transfer rate for each group of flows in the flow table, such that a received packet is dropped if its sending rate is higher than this metered value. In this approach, a set of adjustable metered groups are created and based on the computed Adaptive transfer rate, flows are written into corresponding groups. This implementation causes packets transferred at rates higher than the required rate to be dropped, consequently resulting in the trigger of the congestion avoidance phase at sending host. This implementation requires a congestion cleared state, which is a simple additional variable at the congestion detection module to indicate that the switch is no longer congested and the flows removed from metered groups. This allows for the participating flow enter into the AIMD phase, consequently preventing a quick I/O saturation of the link.

V. Performance Evaluation

In this section, we present a performance evaluation and comparison of our proposed F-DCTCP algorithm with other TCP congestion control algorithms. The performance metrics of interest are throughput analysis with link utilisation, fairness and bandwidth share amongst competing flows and finally flow completion time. The experimental environment is set up to realise an Incast scenario. Table I illustrates the parameters used for the experiments.

TABLE I
Experimental Parameters

Parameters	Values
Link Latency	5ms
Round Trip Time	10ms
Switch Buffer size	0.1MB
Bottleneck Link Speed	1Gbps
Client Link Speed	1Gbps
Number of Concurrent Flows	4

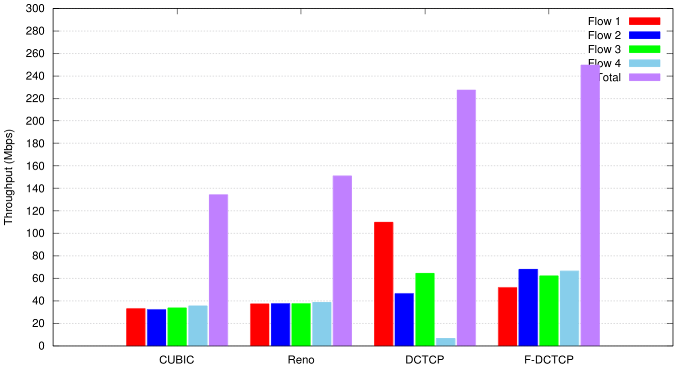

Fig. 5. Average and Total throughput result

The first performance metric we tested is the throughput of F-DCTCP as compared to its competitors. Fig. 5 depicts the average throughput of each of the algorithms under the given settings. We can observe from the Figure that F-DCTCP exhibits the best overall performance amongst all the schemes. Not only in terms of throughput, but also in terms of fairness amongst the competing flows. DCTCP, for example, while has high average throughput, does not have good fairness (flow 4 is almost starved). We can see that the flows are treated unequally/unfairly, as opposed to F-DCTCP that gives a fair share of the bottleneck link. Traditional schemes (CUBIC Reno) have low performance, which is attributed to their inability to cope in partition-aggregate workloads.

Fig. 6 plots the aggregate throughput per flow over time for each of the algorithms. Again, F-DCTCP maintains a high-overall aggregate per-flow throughput.

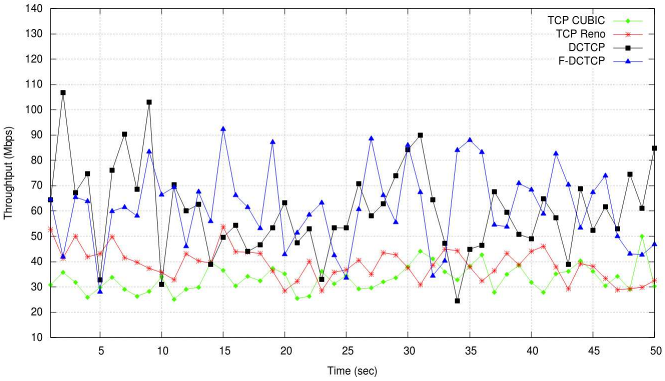

Fig. 6. Aggregate Throughput over time

One of the important features of F-DCTCP is its combined throughput-fairness performance. In order to study the fairness performance of F-DCTCP, we used Jains fairness index [21] which provides a fairness metric that takes all participating flows into consideration. It is bounded between 0 and 1 and provides a direct relationship in percentage scale. It also represents the users perception of the resource sharing clearly. An optimal congestion control solution should maximize this index while ensuring maximum utilisation of the available bandwidth [22]. As depicted in Fig. 7, we can observe that F-DCTCP's fairness index is much better than that of DCTCP. While F-DCTCP's average index is 0.82, the average fairness index of DCTCP is 0.49. Notably, the best fairness is exhibited by traditional schemes (CUBIC and Reno), however they suffer low throughput and link utilisation. This makes F-DCTCP the best overall solution in terms of combined performance. Ensuring fairness here brings a major performance optimisation that requires further attention and development. With the possibility that OVS will include more functionality in its design, this solution can be further improved to achieve ground breaking results.

Finally, we wanted to study the flow completion time of each of the algorithms. Fig. 8 depicts the average flow completion time of four concurrent flows under each of the tested algorithms. As shown in Fig. 8, F-DCTCP has the shortest average completion time amongst all others. Thanks to the global view brought about by SDN that allowed F-DCTCP to optimise flow processing and act on very fine-grained scale.

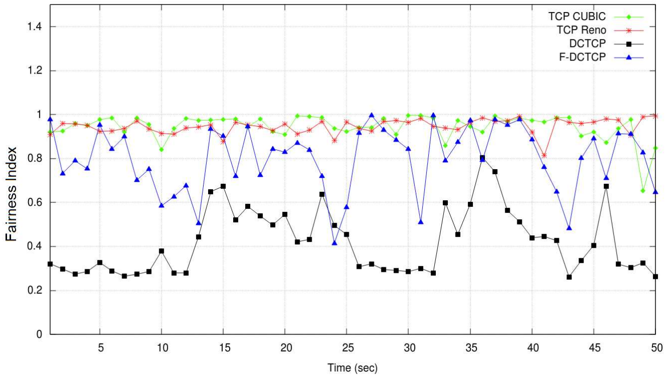

Fig. 7. Jains fairness comparison of F-DCTP to traditional algorithms.

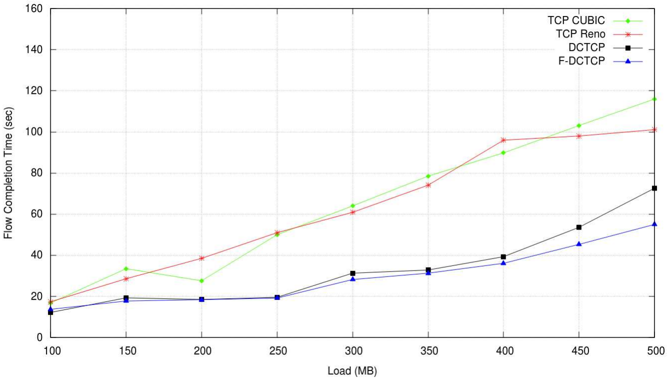

Fig. 8. Average Flow completion time of four concurrent flows.

VI. Conclusion

In this paper, we developed a solution-based framework for adapting traditional congestion control to SDN environment, practically assessing the features of SDN that can be used to optimise congestion control. This work does not converge at just describing a way of adapting traditional algorithms to SDN, we proceeded with implementing an innovative congestion control in SDN, tagged F-DCTCP, this solution majorly focussed on a menacing congestion scenario native to Data Centre Networks (TCP Incast). Following this, we performed a detailed analysis of the benefits of our solution in comparison with existing solutions, establishing that adapting congestion control algorithms to SDN encompasses a distinct performance improvement.

In light of the recommendation above and the proposed F-DCTCP,future work can be tailored along this line especially in creating a flexible and autonomous controller functionality for congestion control and performance optimisation.

References

[1] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, A Survey on Software-Defined Networking, IEEE Commun. Surv. Tutorials, vol. 17, no. 1, pp. 2751, 2015.

[2] D. Kreutz, F. M. V. Ramos, P. E. Verssimo, C. E. Rothenberg, S. Azodolmolky, S. Uhlig, "Software-Defined Networking: A Comprehensive Survey", Proceedings of the IEEE, vol. 103, no. 1, pp. 14-76, Jan. 2015.

[3] Thomas D. Nadeau and Ken Gray, SDN: Software Defined Networks, OREILLY, 2013.

[4] W. Xia, Y. Wen, C. H. Foh, D. Niyato and H. Xie, ”A Survey on Software-Defined Networking,” in IEEE Communications Surveys & Tutorials, vol. 17, no. 1, pp. 27-51, 2015.

[5] D. Lee, B. E. Carpenter, and N. Brownlee, Media streaming observations: Trends in udp to tcp ratio, Fifth International Conference on Internet Monitoring and Protection, pp.99-104, 2010.

[6] D. Kreutz, F. M. V. Ramos, P. E. Verssimo, C. E. Rothenberg, S. Azodolmolky and S. Uhlig, ”Software-Defined Networking: A Comprehensive Survey,” in Proceedings of the IEEE, vol. 103, no. 1, pp. 14-76, Jan. 2015.

[7] I. T. Haque and N. Abu-Ghazaleh, Wireless Software Defined Networking: A Survey and Taxonomy, IEEE Commun. Surv. Tutorials, vol. 18, no. 4, pp. 27132737, 2016.

[8] H. Sandberg, H. Hjalmarsson, U. Jnsson, G. Karlsson, and K. H. Johansson, ”On performance limitations of congestion control,” in 48th IEEE Conference on Decision and Control, Shanghai, China, 2009, pp. 5869-5876.

[9] T. Hafeez, N. Ahmed, B. Ahmed and A. W. Malik, ”Detection and Mitigation of Congestion in SDN Enabled Data Center Networks: A Survey,” in IEEE Access, vol. 6, pp. 1730-1740, 2018.

[10] A. Ya , U. Gital, A. S. Ismail, and H. Chiroma, Performance Evaluation OF TCP Congestion Control Algorithms Throughput For CVE Based On Cloud Computing Model, J. Theor. Appl. Inf. Technol., vol. 1070, no. 1, 2014.

[11] Sangtae Ha, Injong Rhee, and Lisong Xu. 2008. CUBIC: a new TCP-friendly high-speed TCP variant. SIGOPS Oper. Syst. Rev. 42, 5 (July 2008).

[12] Kokshenev, Vladimir & Suschenko, Sergey. (2014). TCP Reno Congestion Window Size Distribution Analysis. Communications in Computer and Information Science. 487. 205-213. 2014.

[13] Y. Lu, Z. Ling, S. Zhu, and L. Tang, SDTCP: Towards Datacenter TCP Congestion Control with SDN for IoT Applications., Sensors (Basel)., vol. 17, no. 1, Jan. 2017.

[14] Elie Krevat, Vijay Vasudevan, Amar Phanishayee, David G. Andersen, Gregory R. Ganger, Garth A. Gibson, and Srinivasan Seshan. 2007. On application-level approaches to avoiding TCP throughput collapse in cluster-based storage systems. In Proceedings of the 2nd international workshop on Petascale data storage: held in conjunction with Supercomputing ’07 (PDSW ’07). ACM, New York, NY, USA, 1-4.

[15] M. Podlesny and C. Williamson, ”Solving the TCP-Incast Problem with Application-Level Scheduling,” 2012 IEEE 20th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, Washington, DC, 2012, pp. 99-106.

[16] S. Islam, M. Welzl, S. Gjessing and J. You, ”OpenTCP: Combining congestion controls of parallel TCP connections,” 2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), Xi’an, 2016, pp. 194-198.

[17] Mohanarangan, S., and D. Sivakumar. ”Limitations and Issues of Congestion Control in Heterogeneity Network.” An International Journal of Advanced Computer Technology COMPUSOFT (IJACT), Special Issue 4th National Conference on Advance Computing, Applications & Technologies by Easwari Engineering College, Chennai in May 2014.

[18] Zhou, Keren, Qian Yu, Zhenwei Zhu and Wenjia Liu. Dynamic Vegas: A Competitive Congestion Control Strategy. (2014).

[19] Mohammad Alizadeh, Albert Greenberg, David A. Maltz, Jitendra Padhye, Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, and Murari Sridharan. 2010. Data center TCP (DCTCP). In Proceedings of the ACM SIGCOMM 2010 conference (SIGCOMM ’10). ACM, New York, NY, USA.

[20] H. Wu, Z. Feng, C. Guo and Y. Zhang, ”ICTCP: Incast Congestion Control for TCP in Data-Center Networks,” in IEEE/ACM Transactions on Networking, vol. 21, no. 2, pp. 345-358, April 2013.

[21] Jain R., Chiu D.M. and Hawe, W., ”A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer Systems”, In DEC Research Report TR-301, September 1984.

[22] C. Joe-Wong, S. Sen, T. Lan and M. Chiang, ”Multiresource Allocation: FairnessEfficiency Tradeoffs in a Unifying Framework”, In Networking IEEE/ACM Transactions on, vol. 21, pp. 1785-1798, 2013, ISSN 1063-6692.