FAIRNESS OF THE TCP-BASED NEW AIMD CONGESTION CONTROL ALGORITHM

HAYDER NATIQ JASEM, ZURIATI AHMAD ZUKARNAIN, MOHAMED OTHMAN, SHAMALA SUBRAMANIAM

Faculty of Computer Science and Information Technology

University Putra Malaysia, 43400 UPM, Serdang, Selangor, Malaysia

E-mail: hayder_n@yahoo.com , zuriati, mothman , shamala@fsktm.upm.edu.my

ABSTRACT

Congestion control is one of the fundamental issues in computer networks. In transport control protocol (TCP) the performance of protocol is being measured based on fairness and efficiency. The fairness on round trip time (RTT) can be measured using the algorithm based on two flows or more than that. Congestion control is an effort to adapt the performance of a network to changes in the traffic load without adversely affecting users perceived utilities. The traffic load in a network will effect the performance of a network and the fairness of algorithm. Congestion control is an introduced effect that adapts the performance. AIMD (Additive Increase Multiplicative Decrease) is an established algorithm in a set of liner algorithms that it reflects good efficiency as well as good fairness. In this paper we propose an evaluation method of fairness for New AIMD congestion control algorithm. The evaluation of fairness has been done by using multiple flows start at the same time and also by considering each flow start at a different time in other way.

Keyword: *TCP, AIMD, Congestion control, Fairness.*

1. INTRODUCTION

The Transmission Control Protocol (TCP) has been the dominant reliable transport layer protocol ever since the appearance of its original version in 1981 [1]. The motivation behind TCP was to add reliability on top of an inherently unreliable IP network. The original TCP incorporated a "sliding window" mechanism, which, in conjunction with packet acknowledgments and segment sequence numbers, guaranteed a reliable data transmission as well as flow control.

2. RELATED WORK

In the early 1980's, network congestion did not constitute a focus of concern due to the limited number of interconnected hosts, and TCP's original version was deemed adequate. As the number of hosts that joined the Internet increased, congestion problems, caused by lack of available bandwidth, became more and more evident. The deficiency of the original TCP was the absence of a mechanism that would adjust the sending rate responding to changes in the network load, namely congestion control. As a result, the network would flood and its overall performance would be severely degraded, leading to a series of 'congestion collapses' in the mid 1980's [2].

2.1. Congestion control

It was not until 1988 that a widely accepted congestion control algorithm was finally suggested [3]. This algorithm employed the Additive Increase Multiplicative Decrease (AIMD) principle. According to the AIMD, a protocol should increase its sending rate by a constant amount and decrease it by a fraction of its original value, each time an adjustment is necessary. This mechanism is the base of virtually all TCP implementations used in today's Internet, since it is proven to converge to both a desirable level of efficiency as well as a desirable level of fairness among competing flows [4].
In the years that followed the establishment of AIMD as the standard algorithm to be used in TCP, Internet underwent numerous changes and rapidly

increasing popularity. With the availability of widespread services such as e-mail and the World Wide Web (WWW), the Internet became accessible to a broader range of people, including users lacking any particular familiarity with computers. Although new competing technologies emerged and the demands from a transport layer protocol were highly increased, TCP not only survived but also became an integral ingredient of the Internet, experiencing only minor modifications. These modifications reflect to the different in-use TCP versions (TCP-Tahoe, TCP-Reno, TCP-NewReno) [3, 5, 6], experimental TCP versions (TCP-SACK, TCP-Vegas) [7, 8], as well as special-purpose TCP versions (T/TCP) [9].

2.2. The AIMD principle

As mentioned earlier, the basic concept of AIMD was proven to yield satisfactory results when the network infrastructure consisted of hard-wire connected components. One year after the appearance of AIMD in 1988, the authors in [11] provided a detailed analysis of different congestion control strategies, as well as what renders the existence of such a strategy in a transport protocol crucial. Below we give a few important points made in this work.

The major issue of concern to a transport protocol is its efficiency. On a network link crossed by a number of different flows running the same protocol, the ideal situation is to utilize as much of the available bandwidth without introducing congestion (i.e. packets queuing up on the router). In Figure 1, we see the achieved throughput as a function of the network load. It becomes clear that we need to avoid overloading the link, since the achieved throughput will diminish. For a protocol to operate in the area between the points labeled as Knee and Cliff, a congestion control mechanism is necessary. In [11] efficiency is defined as the closeness of the total load to the Knee, which is a good starting point.

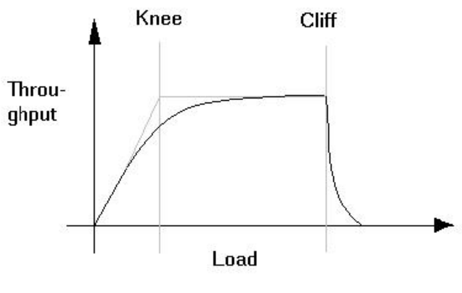

Figure 1: Throughput as a function of load.

Besides utilizing a high portion of the available bandwidth, a transport protocol must also be fair to the rest of the flows traversing the same part of the network. An efficient transport protocol does not necessarily mean that it is also fair. A single flow might take up the largest portion of the available bandwidth while the rest remain idle. Obviously, this is an undesirable behavior and in certain cases, gaining higher fairness is worthwhile even at the cost of reduced efficiency.

Intuitively, fairness is the closeness of the throughput achieved by each flow to its fair share. To measure fairness, the authors in [11] define a fairness index as: $F(x) = \frac{(\sum x_i)^2}{n(\sum x_i^2)}$

Where, x_i is the throughput of the i^{th} flow and n is the total number of flows. The fairness index of a system ranges from 0 to 1, with 0 being totally unfair and 1 being totally fair.

Along the lines of efficiency and fairness, as determined in [16], four different scenarios were tested: Additive Increase Additive Decrease, Additive Increase Multiplicative Decrease, Multiplicative Increase Additive Decrease, and Multiplicative Increase Multiplicative Decrease. These scenarios were evaluated in terms of how fast they converged to the desirable efficiency and fairness levels. The AIMD scheme was found to be the one that better matched the required characteristics. Recent studies [16, 17] provide a more in-depth analysis, regarding the impact of the AIMD parameters on the performance of TCP.

2.3. System model

Chiu and Jain [11] have formulated the congestion avoidance problem as a resource management problem and proposed a distributed congestion avoidance mechanism named 'additive increase/multiplicative decrease' (AIMD). In their work, as a network model they use a "binary feedback" scheme with one bottleneck router [12]. As shown in Figure 2. It consists of a set of m users each of which send data in the network at a rate ^{2} w_i. The data send by each user are aggregated in a single bottleneck and the network checks whether the total amount of data send by users exceeds some network or bandwidth threshold X_{goal} (we can assume that X_{goal} is a value between the knee and the cliff and is a characteristic of the network). The system sends a binary feedback to each user telling whether the flows exceed the network threshold. The system response is 1 when bandwidth is available and 0 when bandwidth is exhausted.

The feedback sent by the network arrives at the same time to all users. The signal is the same to all users and they take the same action when the signal arrives. The next signal is not send until the users have responded to the previous signal. Such a system is called synchronous feedback system or simply synchronous system. The time elapsed between the arrival of two consecutive signals is discrete and the same after every signal arrival. This time is referred also as RTT.

The system behavior can be defined the following time units:

A step (or round-trip time – RTT) is the time elapsed between the arrival of two consecutive signals.
A cycle or epoch is the time elapsed between two consecutive congestion events (i.e., the time immediately after a system response 0 and ending at the next event of congestion when the system response is again 0).

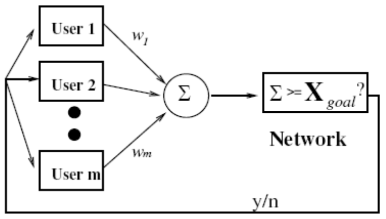

Figure 2: A control system model of m users sharing a network [10].

This network model is quite simple and its assumptions have been evaluated in the Internet for several years. In practice the parameter X_{goal} is the network capacity (i.e. the number of packets that the link and the routers' buffer can hold – or in-the-fly packets). When the aggregate flows' rate exceeds the network capacity the flows start to lose packets. If the transport protocol provides reliability mechanisms (e.g. as in TCP) it can detect the packet loss or congestion event. Since the majority of the applications use reliable transport protocols (e.g. TCP), the binary feedback mechanism has an implicit presence: a successful data transmission is interpreted as available bandwidth, and a packet loss is interpreted as congestion event [3].

Although the system had a strong impact on the evaluation of congestion avoidance mechanisms (e.g. AIMD), there are some limitations. First, the system considers the responses to be synchronous, which, in terms of real networks means that all flows have the same RTT. This assumption is not real. A second assumption and limitation is that the network response arrives at the same time to all users, even when they have the same RTT. This is disputed in [13]. The above assumption is supported by Jacobson experimentally in a low bandwidth network with congestion avoidance mechanisms (TCP-Tahoe) and where flows have the same RTT [14, 15]. Whatever the argument, this assumption is not true for a reason which is the third limitation of the system. The system has only one bottleneck. In reality a connection might go through none, one, or more than one router or bottlenecks. If a flow traverses more than one bottleneck, then it is not guaranteed that at each bottleneck congestion will happen at the same time. Nevertheless, these limitations do not prevent the mechanisms from controlling flows' data rate and avoid congestion which was the major concern in the early stages of the Internet [10].

2.4. Additive Increase / Multiplicative Decrease Control algorithm (AIMD)

The Additive Increase/Multiplicative Decrease (AIMD) algorithms described in detail in [10] and are referred as "dynamic window adjustment" in [18]. The basic idea of the algorithms to reduce the sending rate/window of the flows when the system bandwidth is exhausted and to increase the sending rates/windows when bandwidth is available. As mentioned in the previous section, when bandwidth is available (i.e. the aggregate rates of the flows do not exceed the network threshold: $\Sigma\ w_i < X_{goal}$) the system attaches the signal 1 to the acknowledgment of each packet. In response, flows increase by one (packet) their windows. A continuous series of positive signals will cause a linear increase in the flows' rate. Obviously, the increase is not unlimited because the bandwidth is fixed. When flows' rate exceed the bandwidth limit (i.e. $\Sigma\ w_i \geq X_{goal}$) the system attaches the 0 signal to the acknowledgment of each packet and flows respond to congestion by a decrease in their sending rates/windows.

A. Lahanas and V. Tsaoussidis in [10] prove that a linear increase/exponential decrease policy is a condition for the increase/decrease algorithms to set (or converge) quickly the system in a fair state where the load oscillates around some equilibrium. The equilibrium state determines also the fairness and efficiency of the mechanism.
The convergence behavior of a two flow AIMD system is depicted by vectors in a 2-dimensional space oscillating around the efficiency line (or equilibrium) in Figure 3. Upon each multiplicative decrease, the two windows x_1 and x_2 move closer to the fairness line ($x_1 = x_2$). More details on the convergence of AIMD can be found in [10].

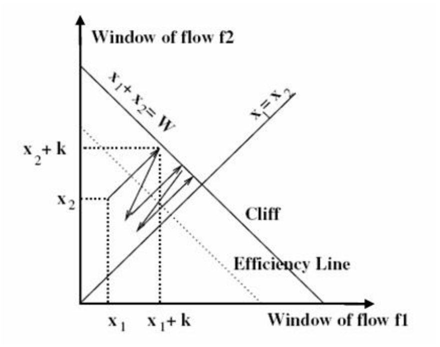

Figure 3: Vectorial representation of two-flow convergence to fairness. Figure is based on [10].

A point on the quadrant between axis represent the sum of the windows. The vectors trace the sum of the windows as they increase or decrease. 'k' denotes the length of the projection of the linear increase vector on the x and y axis. **W** is the value of X_{goal} in terms of packets.

AIMD is also designed to be responsive to fluctuations of bandwidth availability due to varying contention; this is managed by a continuous probing mechanism through additive increase of resource consumption. Chiu and Jain [4] showed that AIMD guarantees convergence to fairness: all flows eventually converge to a fair-share, i.e., an equal allocation of resources. Convergence to fairness is faster when the multiplicative decrease is larger, but then, bandwidth is further underutilized, and applications experience severe transmission gaps. Hence, although smoothness is desirable, it works against fairness: the smoother the adjustment, the longer convergence to fairness takes [19].
And the mathematical formula for AIMD is:

$w \leftarrow w - aw$ *when loss is detected*
$w \leftarrow w + b/w$ *when an ACK arrives*

Efficiency: Controlling the rates of the flows in order to avoid congestion might leave the network resources (i.e., bandwidth) underutilized. So the algorithm that controls the sending rate of the flows should also utilize the resources to the maximum.
Fairness: Since the system is distributed in nature, any attempt to control the sending rate might end up in unfair resource allocation to the flows (e.g., some flows might use more bandwidth than other flows). A mechanism that controls the sending
rate of a flow must use the same bandwidth as the rest of the flows. Two generally used fairness criteria are **max-min fairness** [20, 21] and **proportional fairness** [22]. Max-min fairness concept has emerged from 'fair queuing' models: routers have a separate queue per flow and use the round-robin model to serve the flows. With this model flows that send few data receive higher priority than flows that send bulk data. Proportional fairness favors the flows that have more packets in the routers rather than flows that have fewer packets. This kind of fairness is useful for designing pricing models for the Internet. In this work the network is considered as a black box. The end-to-end protocols do not have any knowledge of the queuing model. However, they try to achieve some kind of fairness. If a flow is a *continuous* transmission of data between two hosts then a desired property of the flows (or the network) would be to have equal rates at a common bottleneck/router. With the above assumption this property matches with the max-min concept of fairness. This fairness is captured by an index or formula that compares the rates/throughputs of the flows.
Distributedness: Congestion can be controlled at the network layer (where it happens) or controlled at the end-nodes. The first (or centralized approach) would be optimal since the network knows better its load at any time. However, this technique
requires processing power from the routers and absolute knowledge of the number of flows and their sending rate. This approach would cause also some overhead since, in addition to data payload flows have to convey also information needed from the routers. Obviously this approach is not consistent with the layered architecture of the network because cooperation and transparency between layers is required to apply these techniques. The burden of control mechanism is delegated on the routers, hence, the name *centralized* or *router-centric* approach. The second approach (referred also as *distributed* or *end-to-end*

approach [23]) attempts to solve the congestion problem from the end-nodes. The routers are kept busy with forwarding packets and need not have extra information about the number of flows and their sending rates. This approach causes less overhead than the centralized approach and relies on the end-nodes' ability to grasp the load of the network. The end-to-end mechanisms can easily be applied to today's packet networks (e.g. Internet), without changing the functionality of any other
layer.

Convergence: A requirement for the control mechanism is to maximize both efficiency and fairness (i.e. to set the network in a state where fairness and efficiency are optimal). Therefore, a desired property of the control algorithm is to converge fast to
this state. In a dynamic environment where flows join or leave the system quite often, the sending rate of each flow will oscillate (downwards) in order to leave the bandwidth available for new incoming flows or (upwards) to achieve the highest goodput when bandwidth is available. Because of these oscillations the system load might not be constantly at the optimal load. Rather, the load oscillates around some load or **equilibrium** point. The time it takes the control algorithm to converge the sending rates of the flows to an acceptable (or optimal) fair state (e.g. the rates are approximately equal) is called **responsiveness**. The oscillation size of the network load around the equilibrium is referred as **smoothness** [4] (see Figure 4). Obviously, an algorithm that has faster convergence time and smaller magnitude of oscillations is better than some
other algorithm that oscillates around the same equilibrium but its convergence time is longer, or its oscillating magnitude is bigger.

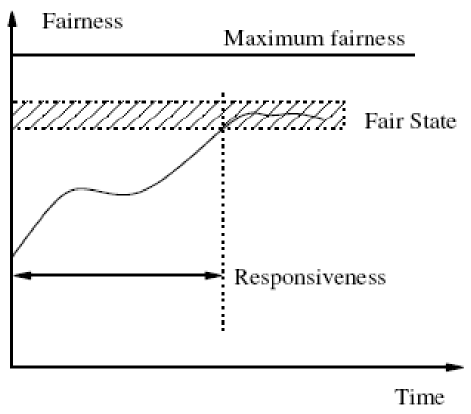

Fig. 4(a)Responsiveness.

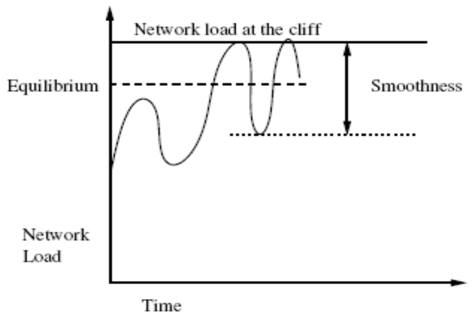

Fig. 4(b) Smoothness.
Figure 4: Responsiveness and Smoothness, figure based on [4].

3. FAIRNESS EVALUATION CRITERIA

In this paper we interest to analysis for the fairness factor it is one of the factors of Congestion Control and another factor such as efficiency was explained and published in [24].

3.1. Fairness

One of the interesting properties of AIMD algorithm that we introduce in the paper is ability of a scheme to approach to fairness monotonically, i.e. the fairness during interval 'i' is given

by $f_i = \frac{x_{1i}}{x_{2i}}, 0 \le f_i \le 1$ (Eq. 1)

(Initially let flows f1 and f2 contain x_1 and x_2 window respectively [24]), k means number Round Trip Time (RTTs) and we assuming system converges 'fair' in 'm' cycle ('m' means number of cycles need for equilibrium state). Then the following conditions should be satisfied.

$\forall i : f_i + 1 \ge f_i$ and $\lim_{i \to \infty} f_i = 1$

Without loss of generality we are assuming that $x_2 = x_1 + n$. At the end of 1st cycle, fairness ratio is given by:

$$\frac{(x_1 + k_1)}{(x_2 + k_1)} = \frac{(x_1 + k_1)}{(x_1 + n + k_1)} =$$

$$1 - \frac{n}{(x_1 + n + k_1)} \quad \text{(Eq. 2)}$$

Similarly at the end 2nd cycle, fairness ratio is given

by $1 - \frac{n}{2}\left(\frac{1}{(\frac{x_1}{2} + \frac{n}{2} + k_1 + k_2)}\right)$. (Eq. 3)

Clearly term $\frac{n}{2}\left(\frac{1}{(\frac{x_1}{2}+\frac{n}{2}+k_1+k_2)}\right)$

is smaller than $\frac{n}{(x_1+n+k_1)}$. (Eq. 4)

Similarly we can find fairness ratio for remaining cycle.

According to these result we can say that our system converge to monotonic fairness. There is one interested question here how much cycles are required for fairness. We have following reasoning for it. Since every time both x_1 and x_2 are divided by 2 of its previous value and equal constant are added in both flows. Thus system can never reach equilibrium if we assume float arithmetic. In Integer arithmetic we are assuming that system reaches fairness in m cycle. It indicates that

$$\frac{x_2}{2^{m-1}}+k_1+k_2...k_m-\frac{x_1}{2^{m-1}}+k_1+k_2...k_m \approx 1 \quad \text{(Eq. 5)}$$

$$\frac{x_2}{2^{m-1}}+\frac{x_1}{2^{m-1}} \approx 1 \quad \text{(Eq. 6)}$$

$$\frac{x_1}{2^{m-1}}+\frac{n}{2^{m-1}}-\frac{x_1}{2^{m-1}} \approx 1 \quad \text{(Eq. 7)}$$

$$n \approx 2^{m-1}, m \approx 1+\log(n). \quad \text{(Eq. 8)}$$

But in AIMD fairness is reflected as $1+\log(x_2)$ [10].

Obviously convergence to fairness of New AIMD is faster than that of AIMD.

3.2. Responsiveness

Numbers of RTTs required for equilibrium (Responsiveness) is measured as:

$$(1+k_1)+(1+k_2)...(1+k_m)=$$

$$m+(k_1+k_2+...k_m)=$$

$$m+(k_1+\left(\frac{w-2k_1}{2}\right)\left(1-(\frac{1}{2})^{m-1}\right). \quad \text{(Eq. 9)}$$

In AIMD algorithm *k* is defined as

$k_i=\frac{w}{4}$ for *i* >=2.

It means number of RTTs is fixed in each cycle. But in our approach

$k_i=\frac{k_{i-1}}{2}$ for *i* >= 3.

It means number of RTTs in each cycle are half of its previous cycle for *i* >=3. Obviously we have less number of RTTs.

4. PROPOSED TOPOLOGY SCHEME

To evaluate the fairness of New AIMD algorithm, we conducted experiments based on NCTUns4.0 simulation. The NCTUns4.0 simulation help us to evaluate the behavior of New AIMD under diverse network condition.

Fig. 5 shows the network topology used in the simulation. The topology is a simple dumbbell topology network. The bottleneck link is set to 5Mbps. The links that connect the senders and the receivers to the routers have bandwidth of 5Mbps. The end-to-end RTT is set to 30ms. The router queue size is 100 packets. The router queue's managed by DropTail.

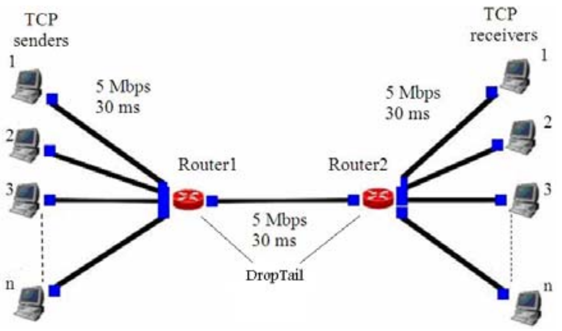

Fig. 5: Multiple flows experimental set-up for New AIMD evaluation.

To measure the fairness, we consider multiple TCP flows and propose the following tests:

(i) Fairness vs multiple flows starting at the same time: Measure the average fair throughput of each flow when each flow operates the same protocol, has the same propagation delay and has a shared bottleneck link and determine the optimal throughput for all flows.

(ii) Fairness vs multiple flows starting at different times: Fairness is calculated here in each interval where the number of flows is constant. In different periods optimal throughput is different as the number of flows is different.

4.1. National Chiao Tung University Simulator (NCTUns):

The NCTU network simulator is a high-fidelity and extensible network simulator and emulator capable of simulating various protocols used in both wired and wireless IP networks. The NCTUns can be used as an emulator, it directly uses the Linux TCP/IP protocol stack to generate high-fidelity simulation results, and it has many other interesting qualities. It can simulate various networking devices. For example, Ethernet hubs, switches, routers, hosts, IEEE 802.11 wireless stations and access points, WAN (for purposely delaying/dropping/reordering packets), optical circuit switch, optical burst switch, QoS DiffServ interior and boundary routers. It can simulate various protocols for example, IEEE 802.3 CSMA/CD MAC, IEEE 802.11 (b) CSMA/CA MAC, learning bridge protocol, spanning tree protocol, IP, mobile IP, Diffserv (QoS), RIP, OSPF, UDP, TCP, RTP/RTCP/SDP, HTTP, FTP and telnet. [25]

4.2. Results

In this section, we present results based on two different kind of measurements that are conducted using all flows have same start time (SST), and different flows have different start time (DST). All simulations are performed under NCTUns. We provide both measurements over total run time (TRT) and over only the second half time (SHT).

4.2.1 Same Start Time (SST)

We will run several simulations using 1, 2, 3, 4, and 5 flows as sources and destinations with the same features. The flow capacity is 5Mbps and the delay is 30ms. Fig. 6, 7, 8 present the fairness for measured over all simulation period (300s) (SST).

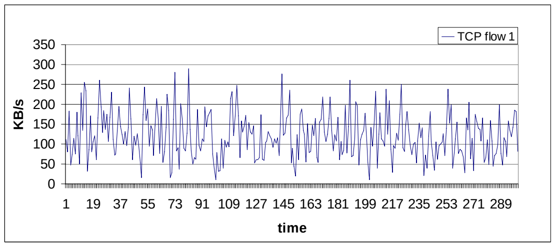

Fig. 6: Throughput fairness for first flow from 5 TCP flows with New AIMD.

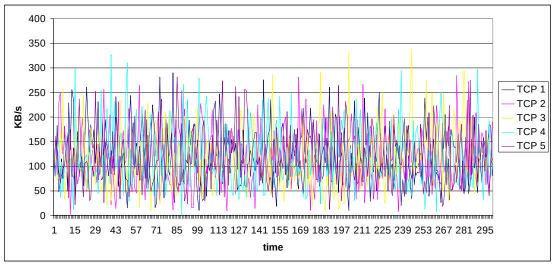

Fig. 7: Throughput fairness for 5 TCP flows with New AIMD.

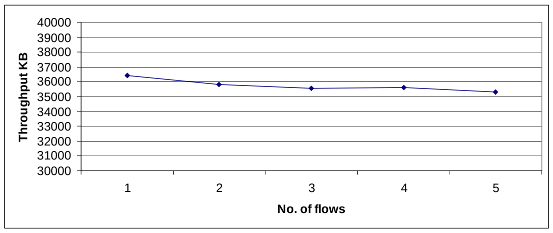

Fig. 8: Throughput vs No. of flows (SST).

4.2.2 Different Start Time (DST)

We run the same simulation were each flow starts at different time. The first flow starts at 0.1s and after each 10s a new flow from the remaining ones starts. Efficiency has to be considered during all the simulation time and we can measure the fairness after 50s from the simulator start time. Fig. 9, 10, 11 present the throughput fairness for 5 flows (DST).

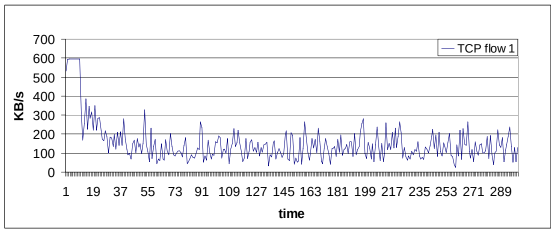

Fig. 9: Throughput fairness for the first flow from 5 TCP flows with New AIMD.

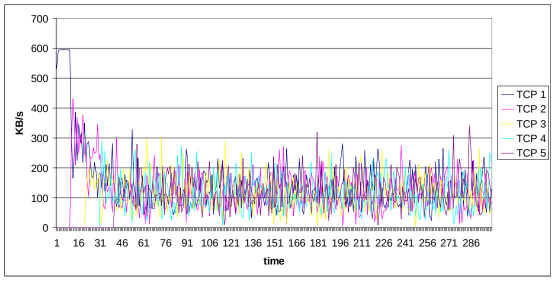

Fig. 10: Throughput fairness for 5 TCP flows with New AIMD.

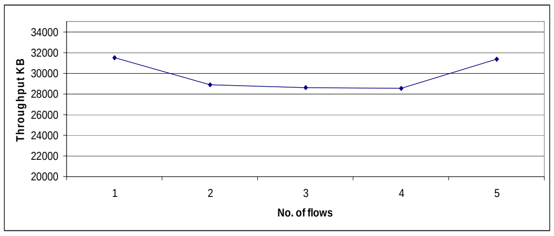

Fig. 11: Throughput vs No. of flows (DST).

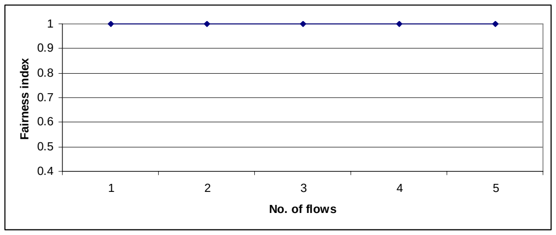

Fig. 12: fairness index

5. CONCLUSION AND FUTUER WORK

In this paper we have focused on the presentation and evaluation of fairness in the new model of AIMD that presented in the previous paper of New AIMD [24], and we make experimental on the new model in the network simulator NCTUns, and we determined that fairness resulting from this model through the equation ($1+\log(n)$) is the best of fairness resulting from the AIMD model in [10], as well as we find the best result of the convergence to fairness through less RTT required for responsiveness to equilibrium for fairness in the new model. And we read the result after make the experimental in two set of flows (SST) and (DST).
In future work can implement this work with more than 5 flows of the TCP and comparing the new results with the results in this paper.

REFERENCES:

[1] J. Postel, “Transmission Control Protocol,” RFC 793, September 1981.
[2] Nader F. Mir, computer and communication networks, Prentice Hall, 2007.
[3] V. Jacobson, “Congestion avoidance and control” in Proc. Of ACM SIGCOMM ’88, August 1988.
[4] D. Chiu and R. Jain. Analysis of the Increase/Decrease Algorithms for Congestion Avoidance in Computer Networks. Journal of Computer Networks and ISDN, 17(1), June 1989.
[5] M. Allman, V. Paxson, W. Stevens, “TCP Congestion Control”, RFC 2581, April 1999.
[6] S. Floyd, T. Henderson, “The New Reno Modification to TCP’s Fast Recovery Algorithm”, RFC 2582, April 1999.
[7] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, “TCP Selective Acknowledgement Options,” RFC 2018, April 1996.
[8] L. Brakmo and L. Peterson, “Tcp Vegas: End to End Congestion Avoidance on a Global Internet,” IEEE Journal on Selected Areas of Communications, October 1995.
[9] R. T. Braden, “T/TCP-TCP Extensions for Transactions, Functional Specification,” RFC 1644, July 1994.
[10]A. Lahanas and V. Tsaoussidis. Exploiting the Efficiency and Fairness Potential of AIMD-based Congestion Avoidance and Control. Journal of Computer Networks, 2003.
[11]D. Chiu and R. Jain. Analysis of the Increase/Decrease Algorithms for Congestion Avoidance in Computer Networks. Journal of Computer Networks and ISDN, 17(1), June 1989.
[12]K. Ramakrishnan and R. Jain. A Binary Feedback Scheme for Congestion Avoidance in Computer Networks with a Connectionless Network Layer. ACM Transactions on Computer Systems, 8(2):158–181,May 1990.
[13]S. Shenker. A Theoretical Analysis of Feedback Flow Control. In ACM SIGCOM Symposium, September 1990.
[14] A. Tang, J. Wang, S. Hedge and S. H. Low. Equilibrium and fairness of networks shared by TCP Reno and Vegas/FAST. Telecommunication Systems, 30(4):417-439, December 2005.
[15] L. Wang, L. Cai, X. Liu and X. Shen. AIMD Congestion Control: Stability, TCP-friendliness, Delay Performance, Tech. Rep., Mar. 2006.
[16] Sachin Kumar, M. K. Gupta, V. S. P. Srivastav, Kadambri Agarwal. “On the Efficiency and Fairness of Congestion Control Algorithms”, Innovative Algorithms and Techniques in Automation, Industrial Electronics and Telecommunications, 405-407, Springer 2007.
[17] Y. Yang and S. Lam, “General AIMD Congestion Control,” in Proceedings of the IEEE International Conference on Network Protocols, November 2000.
[18] A. Lahanas and V. Tsaoussidis. Additive Increase Multiplicative Decrease - Fast Convergence (AIMD-FC). In Proc. Networks 2002, Atlanta, Georgia.

[19] Comer, Douglas E. Internetworking with TCP/IP, 5E, Prentice Hall: Upper Saddle River, NJ. (2006).

[20] D. Bertsekas and R. Gallager. Data Networks, chapter 6, pages 493–530. Prentice Hall, 1987.

[21] J. M. Jaffe. Bottleneck Flow Control. IEEE Transactions on Communications, 29:954–962, 1981.

[22] F. Kelly. Charging and Rate Control for Elastic Traffic. European Transactions on Telecommunications, 8:33–37, 1997.

[23] J. H. Saltzer, D Reed, and D. Clark. End-To-End Arguments in System Design. ACM Transactions on Computer Systems, 2(4):277–288, November 1984.

[24] Hayder Natiq, Zuriati Ahmed, Mohamed Othman, Shamala Subramaniam. “The TCP-Based New AIMD Congestion Control Algorithm”, International Journal of Computer Science and Network Security, VOL.8 No.10, 2008, 331-338.

[25] S. Y. Wang, C. L. Chou, C. C. Lin. The design and implementation of the NCTUns network simulation engine. Science Direct, Simulation Modeling Practice and Theory, 2007, 57-81.