Journal of Applied Sciences

ISSN 1812-5654

science alert

ANSInet
an open access publisher
http://ansinet.com

Journal of Applied Sciences 11 (3): 438-449, 2011
ISSN 1812-5654 / DOI: 10.3923/jas.2011.438.449

Efficiency and Fairness of New-additive Increase Multiplicative Decrease Congestion Avoidance and Control Algorithm

H.N. Jasem, Z.A. Zukarnain, M. Othman and S. Subramaniam
Department of Communication Technology and Networks,
Faculty of Computer Science and Information Technology, University Putra Malaysia,
43400 UPM, Serdang, Selangor, Malaysia

Abstract: When using of the Internet increased dramatically; the congestion avoidance problem became even more important. The congestion is usually caused by the multiplexing for packets when the packets are at the bottleneck links. Efficiency and fairness are the important metrics in the performance of congestion avoidance mechanisms. And also all of the researches for the congestion avoidance algorithms, interest about this parameters metrics to evaluate the performance of the algorithms. This research studied the performance of the New-Additive Increase Multiplicative Decrease (AIMD) algorithm as one of the core protocols for the TCP congestion avoidance and control mechanism. In addition, to evaluate the effect of using the AIMD algorithm after its development to measure the efficiency and fairness and find new enhancement results for our approach, which named as the New-AIMD algorithm. The NCTUns simulator is used to obtain the results after implementing the modifications to the mechanism.

Key words: Congestion avoidance and control, AIMD algorithm

INTRODUCTION

The two main objectives of congestion control are to keep the load of the network within the available network capacity and the second is to share this available capacity fairly between flows that use this network at the same time. The developments of TCP protocol in the late 1970s resulted in the Internet (Poster, 1981). The TCP principles of congestion control were improved a few years later and were influenced by the experiences of "collapses of congestion" in the Internet (Jacobson, 1988; Nagle, 1984).

Control of window-based: In the TCP congestion control we have an important concept, which is the congestion window (cwnd). The cwnd is the amount of data sent from the sender node before it has received the acknowledgment (ACK).

When the congestion window is constant that means one new packet is transmitted for each received ACK. The rate of data sending is indirectly controlled by adjusting the cwnd. From the various versions of TCP the TCP-NewReno is the standard method for this mechanism (Allman *et al*., 1999; Floyd, 2003; Floyd *et al*., 2004). We have extended this TCP version to a TCP with selective acknowledgments (TCP-SACK) (Mathis *et al*., 1996; Blanton *et al*., 2003).

The main goal of TCP-SACK is to avoid reducing the TCP window multiple times when more than one packet from the window is lost (Chiang *et al*., 2007). The out of order arrived segments TCP-SACK will attach it at the packet information acknowledgment. It is the 'selective acknowledgment' or SACK. The sender can decide from this information when to fill the spaces at the buffer of the receiver so that it avoids the multiple decreases from losses that happen within one round trip time RTT (Welzl, 2005).

To implement TCP-SACK, both at the sender and the receiver, requires the SACK options. When one of the TCP's does not implement this option the behaviour of the TCP-SACK will be similar to the behaviour of the TCP-Reno (Blanton *et al*., 2003).

Statuses of the TCP congestion control: In the TCP congestion control there are four distinct statuses, as shown in Fig. 1. The congestion window statuses are related to the congestion control cwnd and also the slow start threshold ssthresh. The ssthresh value determines the window increase rule if (cwnd<ssthresh). For each received ACK the TCP will increase cwnd by one packet

Corresponding Author: H.N. Jasem, Department of Communication Technology and Networks, Faculty of Computer Science and Information Technology, University Putra Malaysia, 43400 UPM, Serdang, Selangor, Malaysia

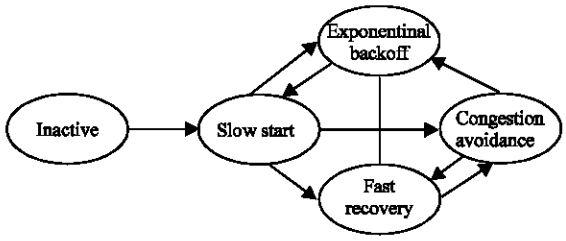

Fig. 1: TCP congestion control statuses

(status of slow start) and the TCP will increase the window size if (cwnd = ssthresh), by one packet per RTT (status of congestion avoidance). When the TCP leaves the status of inactive and enters the status of slow start, the initial values are a two packets cwnd and ssthresh is infinite.

Status of the slow start: The status of slow start is the first status entered creating a flow, or when a flow is reactivated after being inactive. In addition, the status of slow start can be entered as the timeout result. For each non-duplicate ACK, the cwnd will increase by one packet in this status. Two new packets will transmit in response to each received ACK. This means that the congestion window and rate of sending will increase by doubling, once for each RTT.

The TCP will stay in the slow start status until (cwnd>ssthresh). In this case the TCP will go to the congestion avoidance status or, when timeout occurs, the TCP will either go to the exponential back-off status, or after receiving duplicated ACKs three times, the TCP will go to fast recovery status.

When a new flow enters the network, it will motivate the slow start status; also, in the case of a path with a bottleneck link, the old flows will share that link and need some time for slow down before there is space for the new flow to send the packets using the available speed.

Status of the congestion avoidance: The cwnd in the status of congestion avoidance will increase by one packet for each RTT and the cwnd will stay on the maximum value if reached. This is like the sending rate with a linear increase. The TCP will go to the status of exponential back-off when timed out. In addition, after ACKs is duplicated three times, it will go to the status of fast recovery.

When the TCP does not know the current available capacity that it can use, this is the motivation for the mechanism of congestion avoidance; it will need to probe the network to know at what rate the data can get through.

Status of the exponential back-off: After timeout, the TCP will go to the status of exponential back-off. When the TCP enters this status it will result in various actions: retransmitted for the lost packet, or updated of the variables of the status by (ssthresh=cwnd/2), (cwnd=1 packet). The value of retransmission timeout is doubled.

The TCP will go to the status of slow start if it receives the retransmitted packet ACK. Without further packet losses the previous ssthresh update means that the TCP will change from the slow start status to the status of congestion avoidance when the cwnd is increased to half its value before the timeout.

It will repeat the retransmission for the packet. If there is no ACK for it, the retransmission timer will expire again and the retransmission timeout is doubled. In addition, the ssthresh will set to 1 packet (Braden, 1989). The retransmission timeout upper bound is in the order of one or a few minutes.

Until receiving the ACK of the packet the exponential back-off will continue. In this case the TCP will go to the status of the slow start, or the (stack or application) for the TCP will give up and end the connection.

We must repeat the decrease for the load on the network to avoid the congestion collapse, until there is a reasonable probability of small packet loss.

Status of the fast recovery: When the ACKs are duplicated three times, the TCP will go to the status of fast recovery. The first actions in this status for TCP, is retransmitting the packet that was lost and also make the ssthresh equal to cwnd/2. This update for ssthresh is because the next window will increase from the value of cwnd/2 and furthermore, it will use the additional increase for congestion avoidance status and not use the slow start. The TCP will continue sending the new data at nearly or the same data transmission rate; for each duplicate ACK received it will send one new data packet.

The TCP will go to the status of the exponential back-off if did not receive the ACK for the packet that retransmitted within the retransmission timeout (Allman *et al*., 1999) or, when received, the TCP will make cwnd equal to ssthresh for the ACK of the packet that was retransmitted, which means it is equal to half the value of the cwnd when the procedure of recovery started and go to the status of the congestion avoidance. The procedure of fast recovery for the TCP Reno is that it can only recover one packet for every RTT. However, if within the same window it has more than one lost packet, this problem is solved by the TCP-NewReno (Floyd *et al*., 2004) and TCP-SACK (Blanton *et al*., 2003; Mathis *et al*., 1996).

THE AIMD WITHIN TCP

Additive Increase and Multiplicative Decrease (AIMD) is the algorithm that controls and avoids the congestion in the Internet (Chiu and Jain, 1989). According to the TCP signals that it receives from the network, it mechanically adjusts the rate of sending and is coded into TCP (Tsaoussidis and Zhang, 2002).

To achieve greater efficiency and fairness from the AIMD algorithm, Lahanas developed the AIMD algorithm to AIMD-FC (Lahanas and Tsaoussidis, 2003). TCP-Jersey, when the network congestion is detected, will operate based on the estimate of the available bandwidth to optimize the window size (Xu *et al*., 2004). The end-to-end capacity of the path is estimated by the Packet-Pair technique, which uses different arrival times for two packets of the same size transmitting to the same destination from the same source (Keshav, 1991). The TCP with AIMD mechanism for congestion avoidance and control developed to the New-AIMD algorithm (Jasem *et al*., 2008), to achieve more efficiency and fairness than the AIMD-FC+ algorithm (Lahanas and Tsaoussidis, 2003; Lahanas and Tsaoussidis, 2002). Jasem *et al*. (2009b) they investigated, measured and evaluated the fairness of the New-AIMD. In this work we analyse and evaluate the performance of the New-AIMD algorithm within TCP-SACK on the network to avoid and control any congestion and to achieve high efficiency and fairness. Furthermore, we compare the new results with the previous results in Lahanas work (Lahanas and Tsaoussidis, 2003).

AIMD with the congestion problem: Jacobson in (1988) concerned this algorithm, which employed the Additive Increase Multiplicative Decrease (AIMD) principle. However, at that time it was not the finally suggested congestion control algorithm that is now widely accepted (Jacobson, 1988). According to the AIMD, adjustment is necessary each time and a protocol increases its sending rate by a constant amount and decreases it by a fraction of its original value. In the Internet today, this mechanism is the base of virtually all TCP implementations. This is because it is proven to achieve both a desirable level of efficiency and a desirable level of fairness among competing flows (Chiu and Jain, 1989).

The Internet underwent numerous changes and achieved rapidly increasing popularity after the AIMD was established as the standard algorithm to be used in TCP. However, due to the availability of widespread services such as e-mail and the World Wide Web (WWW) the number users of the Internet increased and included users lacking any particular familiarity with computers. TCP not only survived but also became an integral ingredient of the Internet. It experienced only minor modifications, even though new competing technologies emerged and the demands from a transport layer protocol were highly increased. We can see the different uses of TCP versions reflected by different modifications (TCP-Tahoe, TCP-Reno, TCP-NewReno, Westwood) (Jacobson, 1988; Allman *et al*., 1999; Floyd and Henderson, 1999; Caseti *et al*., 2002), experimental TCP versions (TCP-SACK, TCP-Vegas) (Mathis *et al*., 1996; Brakmo and Peterson, 1995; Tsaoussidis and Zhang, 2002), as well as special-purpose TCP versions (T/TCP) (Braden, 1994).

Principle of AIMD: When the network infrastructure consisted of hard-wire connected components, the basic concept of AIMD was proven to yield satisfactory results. Chiu and Jain (1989) one year after the appearance of AIMD, provided a detailed analysis of different congestion control strategies as well as what makes the existence of such a strategy in a transport protocol crucial. Here in this section we will give some points that are important in this work.

The efficiency is the major issue of transport protocol. When a number of different flows running the same protocol cross on a network link and utilize as much of the available bandwidth without introducing congestion, it is the ideal situation similar to when the packets queue up on the router. In Fig. 2 we can see the achieved throughput as a function of the network load. Since the achieved throughput will diminish, this means we need to avoid overloading the link. The congestion control mechanism is necessary when the protocol operates in the area between the points labelled as Knee and Cliff. Depending on (Chiu and Jain, 1989) we can define the efficiency as the closeness of the total load to the Knee, which is a good starting point.

The transport protocol must be fair to the rest of the flows traversing the same part of the network, as well as

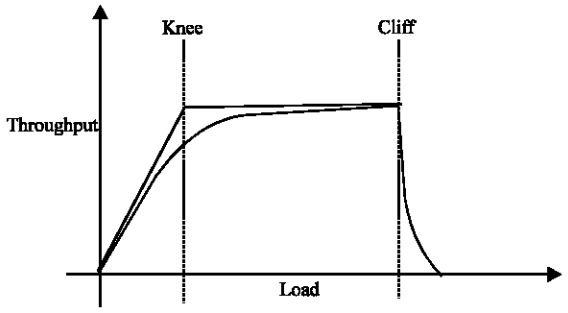

Fig. 2: Throughput as a function of load (Chiu and Jain, 1989)

using a high portion of the available bandwidth. When the transport protocol is efficient, it does not necessarily mean that it is also fair. This is because all the flows may be idle, except one flow that takes the largest portion of the available bandwidth. This behaviour is not desirable and in some cases, it is better to obtain higher fairness even if it results in reduced efficiency.

As a result, you must achieve fairness as well as efficiency so that each flow receives its fair share.

SYSTEM MODEL

Chiu and Jain (1989) proposed a distributed congestion avoidance mechanism named 'additive increase/multiplicative decrease' (AIMD), to solve the problem of congestion avoidance. It was formulated as a resources management problem. In that work, as a network model, as shown in Fig. 3 and 4, they used a "binary feedback" scheme with one bottleneck router (Ramakrishnan and Jain, 1990). It includes a set of m users as senders and receivers, with each of the senders sending data in the network at a rate of w_i. The data sent by each user is aggregated in a single bottleneck and the network checks whether the total amount of data sent by the 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). Binary feedback is sent from the system to each user (sender) informing whether the flows are more than the network threshold or not.

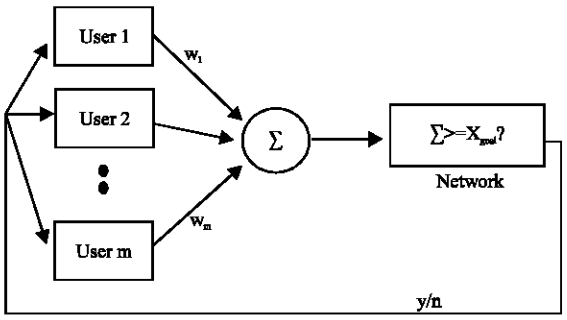

Fig. 3: Control system model of m users sharing a network (Lahanas and Tsaoussidis, 2003)

The response of the system is 0 when the bandwidth is exhausted and 1 when the bandwidth is available.

The feedback is sent by the network and arrives for all users at the same time. All the users take the same action when the signal arrives, because all of them receive the same signal. When the users respond to the previous signal, the next signal will send. We can call this system a synchronous feedback system, or in simplified terms a synchronous system. The elapsed time between the arrival of two consecutive signals is discrete and the same after the arrival of every signal. This time is referred to as RTT.

The following time units can define the system behaviour:

- A step or (RTT) is the elapsed time between the arrivals of two consecutive signals.
- A cycle is the elapsed time between two consecutive events of congestion.

Practically, the network capacity parameter is X_{goal} (i.e., it is the ability of the router buffer and the link to hold

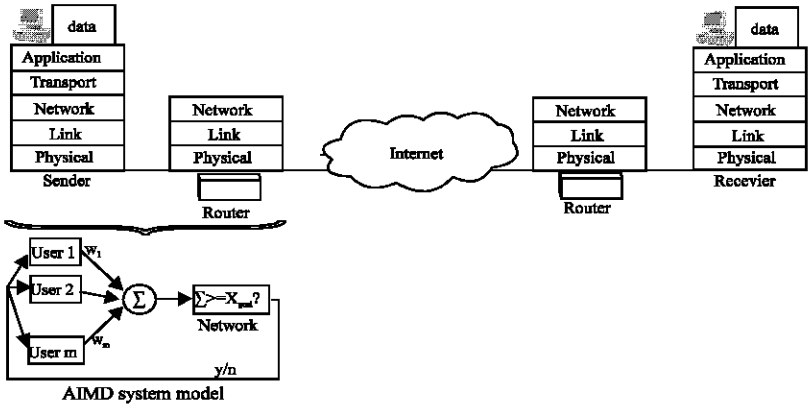

Fig. 4: The AIMD system model within layered architecture in the internet environment

the packets). The flows will start to lose packets, when the aggregate flow rate exceeds the capacity of the network. They can detect the congestion event or packet loss, when the reliability is supported by the transport protocol, similar to the TCP. The mechanism of binary feedback has an implicit presence, when the majority of the applications use reliable transport protocols (like TCP), the successful data transmission means that the bandwidth is available and the loss of packets means a congestion event (Jacobson, 1988).

The AIMD can be expressed algorithmically with the following lines:

```
AIMD (W)
1   α1 : constant = packet-size (W)
2   W : integer // congestion window
3   repeat forever
4   send W bytes in the network
5   receive ACKs
6   if W bytes are ACKed
7   W ← W + α
8   else
9   W ← W/2
10  end
END-AIMD
```

In Lahanas work (Lahanas and Tsaoussidis, 2003), which is related to our work, we can see the improvement of the AIMD algorithm, when he developed the efficiency for this algorithm to 88.9%. Lahanas called the algorithm (AIMD-FC+), by algorithmically with the following lines:

```
AIMD-FC+ (w,dw,q)
1   k← 0
2   while (feedback is 1)
3   process (w+k)
4   k ←k + a1
5   end
6   dw ← 1/2 dw + (w + k − dw)
7   w ← dw + f      (q)
END
```

When w is the window size, dw is variable to record the decrease window and f (q) is $q = k^{j-2}$ when, $0 \le f(q)/2$ as described by Lahanas and Tsaoussidis (2003).

THE NEW-AIMD APPROACH

Theorem: Let us assume that the capacity for the network (Window size or X_{goal}) is W. For simplicity, we will suppose that we have two flows, f1 and f2. Initially let flows f1 and f2 include x_1 and x_2 windows sequentially. To maintain the generality we will assume that $x_1 < x_2$ and $x_1 + x_2 < W$, moreover, we are supposing that the system converges to 'fair' in 'm' cycle.

Induction proof: When we prove the correctness we will make it for two flows and similarly it can be generalized for many flows. In the 1st cycle, the Pseudocode is given by the total flow:

$$x_1 + x_2\ 2k_1 \tag{1}$$

And in the AIMD it is $x_1 + x_2 + 2k_1$

It is clear that in the 1st cycle the system has +1 Round Trip Time (RTTs) or steps. Let $x_1 + x_2 + 2k_1 \ge W$ then congestion occurs and the system will give 0 as feedback. Now we will use the decrease step. In the 2nd cycle Pseudocode is given by total flow:

$$\frac{x_1}{2} + \frac{x_2}{2} + 2k_1 + 2k_2 \tag{2}$$

But in the AIMD it is:

$$\frac{x_1}{2} + \frac{x_2}{2} + k_1 + 2k_2$$

It is clear that the 2nd cycle includes $k_2 + 1$ RTT. Let:

$$\frac{x_1}{2} + \frac{x_2}{2} + 2k_1 + 2k_2 \ge W$$

then the system will give 0 as feedback. It is clear that we will use the decrease step again. In the 3rd cycle (RTT) Pseudocode is given by the total flow:

$$\frac{x_1}{2^2} + \frac{x_2}{2^2} + 2k_1 + 2k_2 + 2k_3 \tag{3}$$

But in the AIMD it is:

$$\frac{x_1}{2^2} + \frac{x_2}{2^2} + k_1 + k_2 + 2k_3$$

Now here the 3rd cycle includes $k_3 + 1$ RTTs. Let:

$$\frac{x_1}{2^2} + \frac{x_2}{2^2} + 2k_1 + 2k_2 + 2k_3 \ge W$$

then the system will give 0 as feedback. It is clear we will use the decrease step also. In the same way at m^{th} cycle we will have total flow:

$$\frac{x_1}{2^{m-1}} + \frac{x_2}{2^{m-1}} + 2k_1 + 2k_2 \ldots 2k_m \tag{4}$$

But in the AIMD it is:

$$\frac{X_1}{2^{m-1}} + \frac{X_2}{2^{m-1}} + k_1 + k_2 ... 2k_m$$

We will expect m^{th} cycle indicates to equilibrium, that is all flows share a fair allocation of the network resources.

The algorithmically approach, when the initial window size of two flows and the window size are x_1, x_2, w, respectively is given by:

New-AIMD ()
1 z ← x_1, x_2
2 k ← 1
3 t ← 1
4 while (feedback is 1)
5 K ← k + 1
6 Z ←
7 t ← 1
8 if (z > = W)
9 $x_1 \leftarrow \frac{x_1}{2}$
10 $x_2 \leftarrow \frac{x_2}{2}$
11 z ← x_1 + x_2 + 2t
12 k ← k + 1
13 end if
14 end
END-AIMD

We used the following notation:
Z indicates used network capacity
K indicates numbers of RTTs
t is number of steps
m Integer, to represent the number of cycles
w the window size
x_1, x_2 indicates the two flows that use the resources

The total number of packets in different cycles:
In the 1st cycle, the total number of packets is produced by $(k_1 + 1)$ $(x_1, x_2, 2k_1)$ but from the 1st cycle we have $x_1, + x_2 + 2k_1 = W$, So $x_1, + x_2\, k_1 = W\text{-}k_1$.

Thus, the total number of packets is produced by $(1 + k_1)$ $(W\text{-}k_1)$.

In the 2nd cycle, the total number of packets is produced by:

$$(1+k_2)(\frac{X_1}{2} + \frac{X_2}{2} + 2k_1 + k_2)$$

But from the 2nd cycle we have:

$$(\frac{X_1}{2} + \frac{X_2}{2} + 2k_1 + 2k_2) = W$$

So:

$$\frac{X_1}{2} + \frac{X_2}{2} + 2k_1 + k_2 = W - k_2$$

Thus, the total number of packets is produced by$(1 + k_2)$ $(W\text{-}K_2)$.

In the same way for the 3rd cycle, the total number of packets is produced by $(1 + k_3)$ $(W\text{-}k_3)$.

In the same way in the m^{th} cycle, the total number of packets is produced by $(1 + k_m)$ $(W\text{-}k_m)$.

And so on; the total number of packets in all cycles is produced by $(1+ k_1)$ $(W\text{-}k_1) + (1 + k_2)$ $(W\text{-}k_2) + (1 + k_3)$ $(W\text{-}k_3) + ... + (1 + k_m)$ $(W\text{-}k_m)$.

But from the 1st equation we have $k_2 = (W\text{-}2k_1)/k_2$
And from Eq. 2 and 3 we have $k_3 = \frac{1}{2}k_2$
And from Eq. 3 and 4 we have:

$$k_4 = \frac{1}{2}k_3 \text{ , } k_4 = (\frac{1}{2^2})k_2$$

and so $k_m = (\frac{1}{2^{m-2}})k_2$ for m=3.

The efficiency of New-AIMD is more than 99%; this means it is 11% higher than the efficiency of AIMD-FC+ in (Lahanas and Tsaoussidis, 2003).

The Additive Increase/Multiplicative Decrease (AIMD), (AIMD-FC) and (New-AIMD) algorithms are described in (Lahanas and Tsaoussidis, 2003; Jasem *et al*., 2008; Jasem *et al*., 2009a, b), respectively.

PERFORMANCE METRICS

Now we want to define the performance metrics that are related to this work and are used to evaluate the different congestion avoidance schemes (or network performance).

Throughput is the data transfer rate, or the average of data sent by a flow per RTT.

Goodput is the average of data that is already received by a flow per RTT. It is also used to show the ability of the protocol to use the available bandwidth.

The throughput and goodput can be measured by (bits/sec, bytes/sec or packets/sec).

Packet loss when the queue in the routers and other network devices in shared networks fill they will start to drop packets. The packet that is sent from the source but did not arrive at its destination in the period of time as a dropped packet (or lost packet).

Packet Loss% = (No. of unacknowledged packets/Total No. of packets transmitted) X 100

Efficiency is the average flows, (goodput per RTT) over the system capacity. Efficiency is a value between 0 and 1 or 0 and 100%.

Fairness recognizes the effectiveness of the sharing of resources in the bottleneck link. To measure the fairness in a system, we have many ways. In this work we used the known function (Jain fairness index) that is used in a related work (Lahanas and Tsaoussidis, 2003)

$$F(x)=\frac{(\sum x_i)^2}{n(\sum x_i^2)}$$

Where, n is the total number of flows and x_i is the goodput of the flow i^{th}. The range of the system fairness index is from 0 to 1. If it equals 0 it means the system is completely unfair and if it equals 1 it means the system is completely fair (Lahanas and Tsaoussidis, 2003; Jasem *et al.*, 2009b; Welzl, 2005).

NATIONAL CHIAO TUNG UNIVERSITY NETWORK SIMULATOR

The works of Wang *et al.* (2007) explained the uses and performance of the NCTU network simulator. In addition, they said that it has 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.

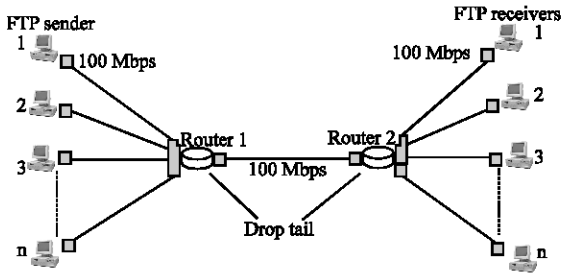

Fig. 5: Multiple flow experimental set-up for AIMD evaluation

EXPERIMENT SET-UP FOR EVALUATION AIMD ALGORITHMS WITHIN TCP

When implementing our evaluation plan on the NCTUns network simulator, multiple flows shared a bottleneck link. The network topology used as a test-bed is the typical single-bottleneck two dumbbells, as shown in Fig. 5.

For the simulation scenario, as a general case, we use the following setup details:

The time for experiment is fixed at 100 sec. The link's capacity at the senders, receivers and bottleneck link is (full-duplex) 100 Mbps. The mechanism of active queue management (AQM) in the queues is the Droptail mechanism (DT). We used an equal number of sender and receiver nodes. The traffic for each source is generated by FTP application when the TCP flow runs in each node. We suppose that all the flows are sent at the same time. All DT queues have 100-packet lengths. Furthermore, we used the TCP-SACK as one variant of TCP versions with AIMD, AIMD-FC+ and New-AIMD, to evaluate the performance of the new algorithm compared with the algorithms in the related work (Lahanas and Tsaoussidis, 2003).

RESULTS OF SIMULATION

Figure 6-10 show the new result for the mechanism in our approach. In the evaluation experiments for this work we take one flow and study it as an example for all flows that share the network resources and to show the behaviour for the flow when the flow uses the network resources alone, or when it shares it with other flows.

In addition, from Fig. 6-10 we can recognize the small difference between the throughput (the data rate sent from source) and goodput (the data rate received in the destination). This means that from the development of the algorithm we achieve the benefit of little loss rate. Furthermore, we can see that the data transmit rate decreased in the experiments with a greater number of

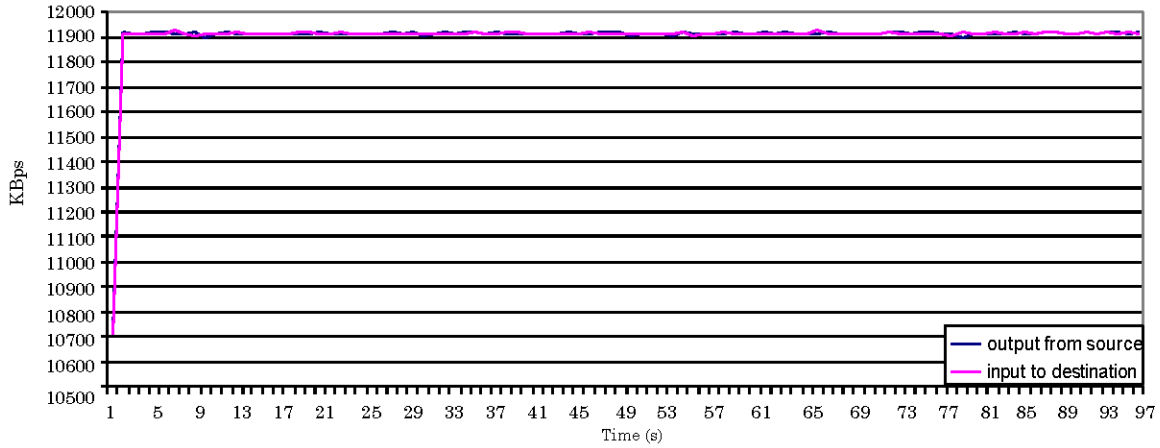

Fig. 6: The result for one flow in experiment with 1 flow

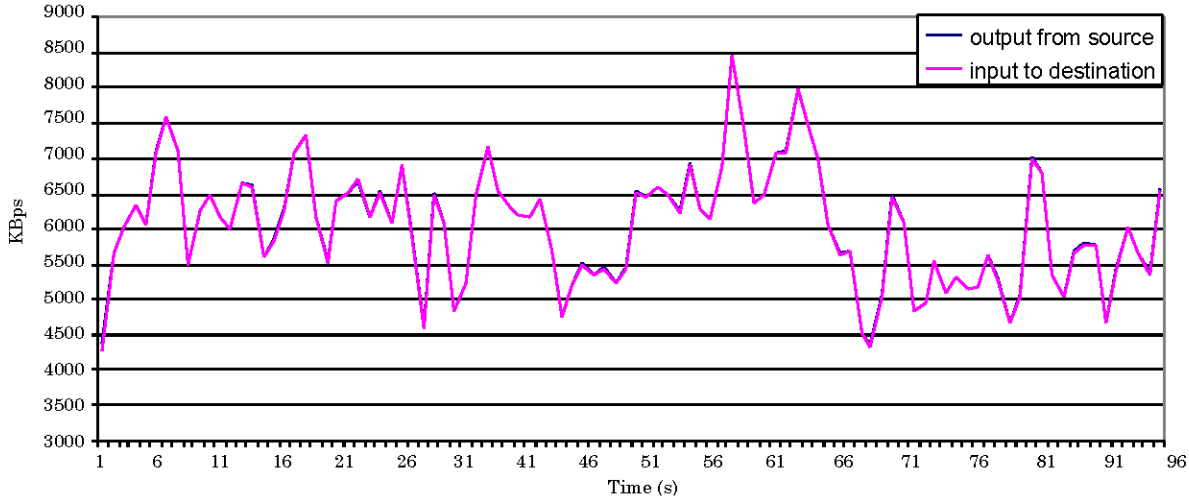

Fig. 7: The result for one flow in experiment with 2 flows

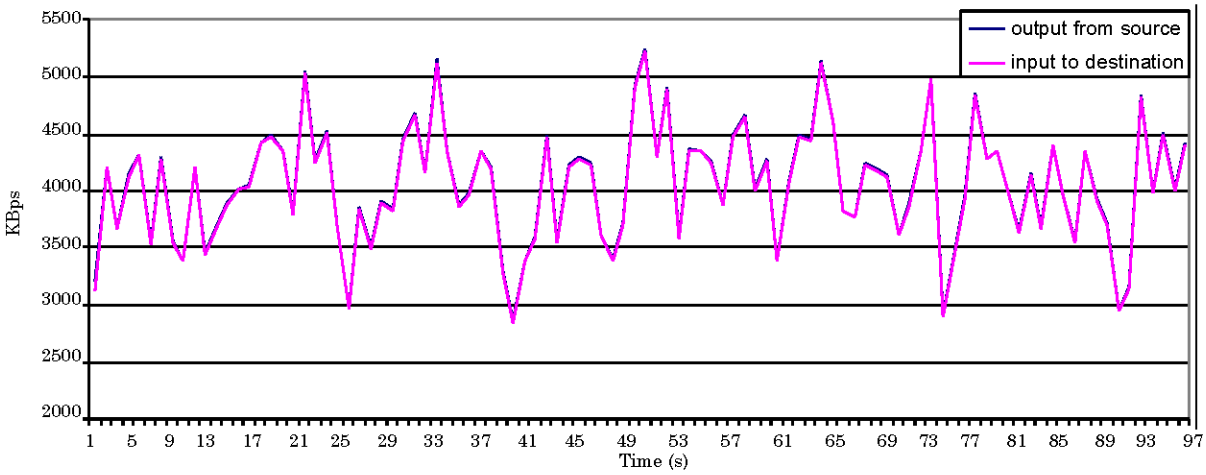

Fig. 8: The result for one flow in experiment with 3 flows

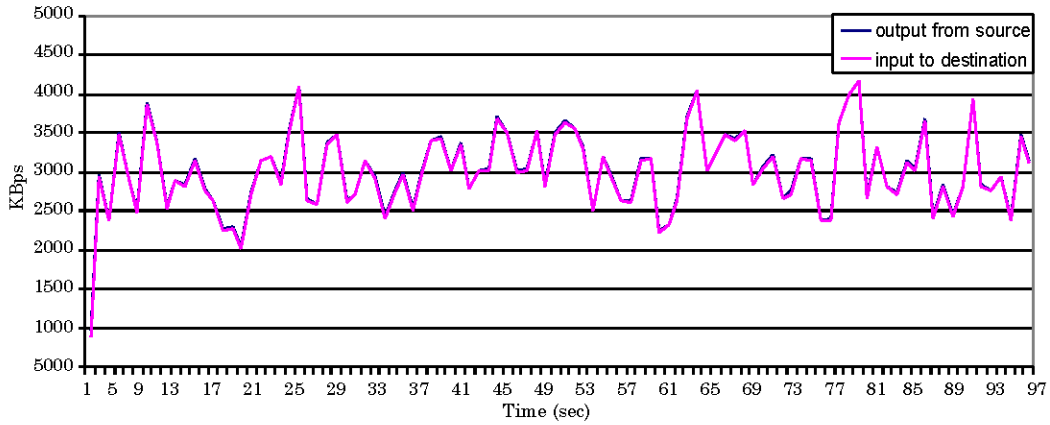

Fig. 9: The result for one flow in experiment with 4 flows

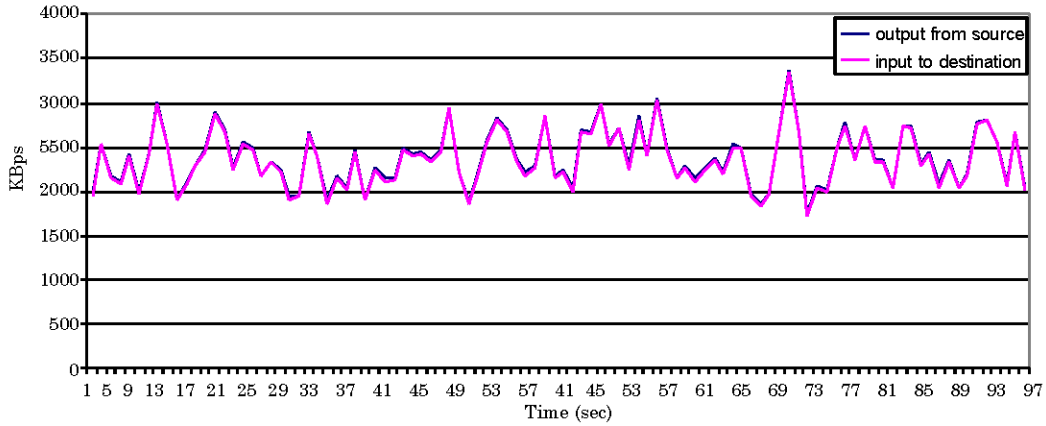

Fig. 10: The result for one flow in experiment with 5 flows

Table 1: Throughput, goodput and loss rate for different numbers of flows in a single bottleneck link with New-AIMD algorithm

	TCP				
No. of flows in the experiment	Total throughput per-flow sent (KB)	Total goodput per-flow received (KB)	Drop (KB)	Goodput rate received (%)	Loss rate (%)
1	1190271.96	1190240.44	31.52	99.997	0.003
2	603924.83	603008.556	916.274	99.848	0.152
3	405776.368	404589.184	1187.184	99.707	0.293
4	302168.314	300639.482	1528.832	99.494	0.506
5	238063.174	236135.66	1927.514	99.19	0.81

flows. This is because when the other new flows start to send data they need to share the resources with the previous flows that were using the network before to achieve high fairness. This is another indication of good performance for the developed algorithm.

Table 1 shows the experimental results for the number of flows (1-5), as a sample for this case of measurement and explains the results of flow in the experiments with one or many flows.

These important results, such as throughput, goodput and drop or loss rate, as shown in Table 1 are the major parameters for evaluating the performance of the algorithm (efficiency and fairness).

In order to ensure a thorough evaluation of the performance of the developed algorithm, we compared the results obtained by the implementation of the developed algorithm with the results reached by the implementation of algorithms used in the previous related

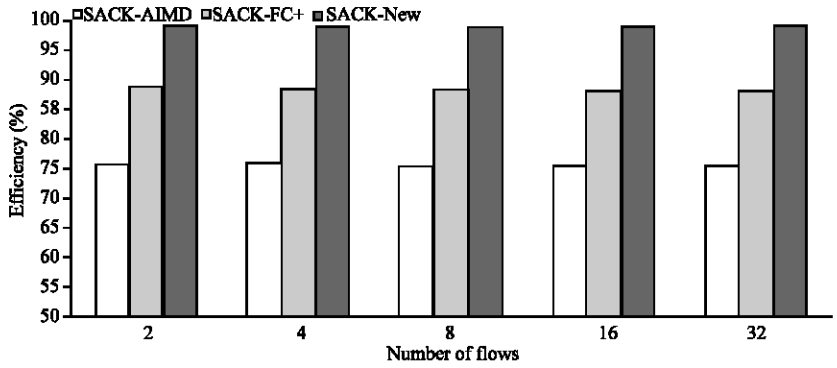

Fig. 11: The efficiency (%) for AIMD, AIMD-FC+ and New-AIMD algorithms with multiple flows within TCP-SACK

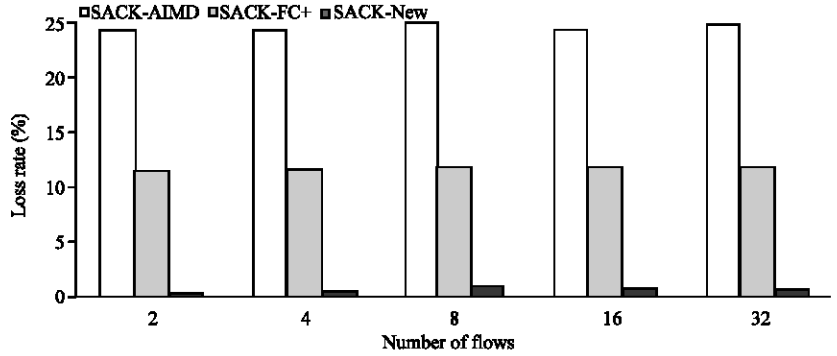

Fig. 12: The loss rate (%) for AIMD, AIMD-FC+ and New-AIMD algorithms with multiple flows within TCP-SACK

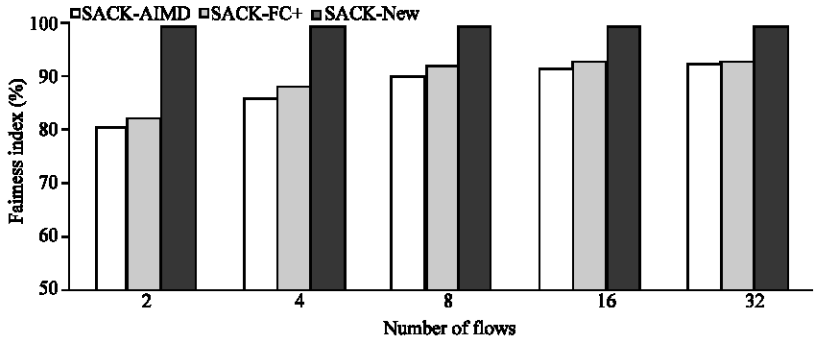

Fig. 13: The fairness (%) for AIMD, AIMD-FC+ and New-AIMD algorithms with multiple flows within TCP-SACK

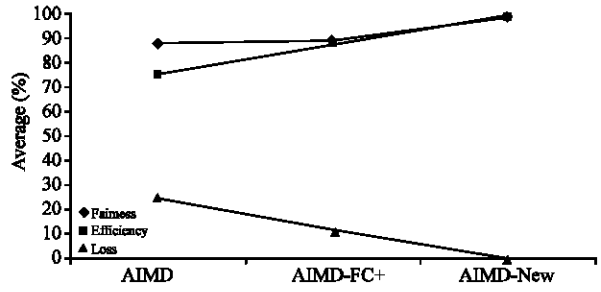

Fig. 14: The averages of fairness, efficiency and loss for AIMD algorithms within TCP-SACK

work (New-AIMD with AIMD and AIMD-FC+). As shown in the Fig. 11-13, the comparison between three algorithms (AIMD, AIMD-FC+ and New-AIMD) was judged on the efficiency, loss and fairness.

For more detail we can see the value of all the compared parameters in the tables attached to the figures.

From the Fig. 11-13, we can see the difference between the results of the algorithms. It is clear that the New-AIMD has the better results for high efficiency, low loss and also high fairness.

The Fig. 14 shows the difference between the averages of the performance for the algorithms and the value of evaluation for enhancement of the algorithm. It is dependent on the evaluation of efficiency, loss and fairness. The average is from 20 experiments with minimal statistical deviation.

CONCLUSION

In the experiments for this study, we investigated two types of performance metrics. The first one being efficiency, which depends on the throughput and goodput for the flows in the system and for which we found that the results after implementing the New-AIMD algorithm were better than the results in the previous work (Lahanas and Tsaoussidis, 2003). This is because when we measured the efficiency we obtained more than 99%, which is around 11% higher efficiency than achieved in previous related work. The second one (performance metrics) is fairness. We found that the fairness result is also high being more than 99%.

Then we made a comparison between the new results for this work and the results for the previous related works, we found that the new result is much better, as can be seen in the previous figures.

Therefore, we can say that this new mechanism works well under the conditions for the network experiments above. In addition, we can also say that the benefit from the implementation of the New-AIMD algorithm in this work is to increase the average efficiency and decrease the loss rate and, furthermore, to increase the average of fairness for flows that use the network resources as well as avoid the network congestion.

REFERENCES

Allman, M., V. Paxson and W. Stevens, 1999. TCP congestion control. RFC 2581. http://portal.acm.org/citation.cfm?id=RFC2581.

Blanton, E., M. Allman, K. Fall and L. Wang, 2003. A conservative selective acknowledgment (SACK)-based loss recovery algorithm for TCP. RFC 3517, April. http://portal.acm.org/citation. cfm?id=RFC3517.

Braden, R., 1989. Requirements for internet hosts-communication layers. RFC 1122, October, pp: 112. http://www.faqs.org/rfcs/rfc1122.html.

Braden, R.T., 1994. T/TCP-TCP extensions for transactions, functional specification. RFC 1644, July. http://portal.acm.org/citation.cfm?id=RFC1644.

Brakmo, L. and L. Peterson, 1995. TCP vegas: End to end congestion avoidance on a global internet. IEEE J. Select. Areas Commun., 13: 1465-1480.

Caseti, C., M. Gerla, S. Mascolo, M.Y. Sansadidi and R. Wang, 2002. TCP westwood: End-to-end congestion control for wired/wireless networks. Wireless Networks, 8: 467-479.

Chiang, M., S.H. Low, A.R. Calderbank and J.C. Doyle, 2007. Layering as optimization decomposition: A mathematical theory of network architectures. Proc. IEEE, 95: 255-312.

Chiu, D.M. and R. Jain, 1989. Analysis of the increase/decrease algorithms for congestion avoidance in computer networks. J. Comput. Networks ISDN., 17: 1-14.

Floyd, S. and T. Henderson, 1999. The new reno modification to TCP's fast recovery algorithm. http://tools.ietf.org/pdf/rfc2582.

Floyd, S., 2003. HighSpeed TCP for large congestion windows. RFC3649, pp: 1-20. http://acs.lbl. gov/~evandro/hstcp/notes/hstcp-dsd.pdf.

Floyd, S., T. Henderson and A. Gurtov, 2004. The NewReno modification to TCPs fast recovery algorithm. RFC 3782, April. http://www. ietf.org/rfc/rfc3782.txt.

Jacobson, V., 1988. Congestion avoidance and control. ACM SIGCOMM Comput. Commun. Rev., 18: 314-329.

Jasem, H.N., Z. Ahmed, M. Othman and S. Subramaniam, 2008. The TCP-based new AIMD congestion control algorithm. Int. J. Comput. Sci. Network Security, 8: 331-338.

Jasem, H.N., Z.A. Zukarnain, M. Othman and S. Subramaniam, 2009a. The new AIMD congestion control algorithm. Proceedings of World Academy of Science, Engineering and Technology, Vol. 38, Penang, Malaysia.

Jasem, H.N., Z.A. Zukarnain, M. Othman and S. Subramaniam, 2009b. Fairness of the TCP-based new AIMD congestion control algorithm. J. Theory Applied Inform. Technol., 5: 568-576.

Keshav, S., 1991. Congestion control in computer networks. Ph.D. Thesis, University of California

Lahanas, A. and V. Tsaoussidis, 2002. Additive increase multiplicative decrease-fast convergence (AIMD-FC). Proceedings of the Networks 2002, Atlanta, Georgia.

Lahanas, A. and V. Tsaoussidis, 2003. Exploiting the efficiency and fairness potential of AIMD-based congestion avoidance and control. J. Comput. Networks, 43: 227-245.

Mathis, M., J. Mahdavi, S. Floyd and A. Romanow, 1996. TCP selective acknowledgement options. http://www.icir.org/floyd/sacks.html.

Nagle, J., 1984. Congestion control in IP/TCP internetworks. ACM SIGCOMM Comput. Commun. Rev., 14: 11-17.

Poster, J.B., 1981. Transmission control protocol. RFC 793. http://www.faqs.org/rfcs/rfc793.html.

Ramakrishnan, K. and R. Jain, 1990. A binary feedback scheme for congestion avoidance in computer networks with connections network layer. ACM Trans. Comput. Syst., 8: 158-181.

Tsaoussidis, V. and C. Zhang, 2002. TCP-Real: Receiver-oriented congestion control. Comput. Networks, 40: 477-497.

Wang, S.Y., C.L. Chou and C.C. Lin, 2007. The design and implementation of the NCTUns network simulation engine. Simulation Modeling Practice Theory, 15: 57-81.

Welzl, M., 2005. Network Congestion Control Managing Internet Traffic. 1st Edn., John Wiley and Sons Ltd., New York.

Xu, K., Y. Tian and N. Ansari, 2004. TCP-Jersey for wireless IP communications. IEEE JSAC., 22: 747-756.