Scientific Research and Essays Vol. 5(23), pp. 3719-3729, 4 December, 2010
Available online at http://www.academicjournals.org/SRE
ISSN 1992-2248 ©2010 Academic Journals

Full Length Research Paper

On the delay and link utilization with the new-additive increase multiplicative decrease congestion avoidance and control algorithm

Hayder Natiq Jasem[1,2]*, Zuriati Ahmad Zukarnain[1], Mohamed Othman[1] and Shamala Subramaniam[1]

[1]Department of Communication Technology and Networks, Faculty of Computer Science and Information Technology, University Putra Malaysia.
[2]Department of Computer Science, Faculty of Science for Woman, University of Baghdad, Iraq.

Accepted 15 November, 2010

Additive increase multiplicative decrease (AIMD) algorithm is the prevailing algorithm for congestion avoidance and control in the Internet. Reducing the end-to-end delays and enhancement of the link utilization are the important goals of this algorithm. In this work, we continue to study the performance of the New-AIMD (additive increase multiplicative decrease) mechanism as one of the core protocols for TCP, to avoid and control the congestion. We want to evaluate the effect of using the AIMD algorithm after developing it, which we called the New-AIMD algorithm, to find a new approach to measure the end-to-end delay and bottleneck link utilization and use the NCTUns simulator to obtain the results after making the modification for the mechanism. We will use the DropTail mechanism as the active queue management mechanism (AQM) in the bottleneck router. After the implementation of our new approach with a different number of flows, we expect the end-to-end delay to be less when we measure the delay dependent on the throughput for the entire system. In addition, we will measure the bottleneck link utilization using this mechanism and expect to get high utilization for bottleneck link and avoid the collisions in the link.

Key words: Congestion control, TCP, AIMD, delay, link utilization.

INTRODUCTION

End-to-end congestion avoidance and control, as well as fair network resource management would be of considerable benefit, if the TCP sender knows of the behaviour and any delay in the bottleneck queue. Several methodologies have been developed to estimate bandwidth and bottleneck queue, based on the temporary measurements of throughput, inter-packet gap or RTT. For example, TFRC (Handley et al., 2003) calculates throughput via a throughput equation that incorporates the loss event rate, round-trip time and packet size. TCP-Vegas (Brakmo and Peterson, 1995) estimates the level of congestion using throughput-based measurements.

TCP-Vegas demonstrated that measurement-based window adjustment is a viable mechanism; however, the corresponding estimators can be improved. In TCP-Westwood (Casetti et al., 2002), the sender continuously measures the effective bandwidth used by monitoring the rate of returned ACKs. TCP-Real (Tsaoussidis and Zhang, 2002) uses wave patterns: a wave consists of a number of fixed-sized data segments sent back-to-back, matching the inherent characteristic of TCP to send packets back-to-back. The protocol computes the data-receiving rate of a wave, which reflects the level of contention at the bottleneck link. Bimodal congestion avoidance and control mechanisms (Attie et al., 2003) compute the fair-share of the total bandwidth that should be allocated for each flow at any point during the system's execution.

Additive Increase/Multiplicative Decrease (AIMD) is the algorithm that controls congestion in the Internet

*Corresponding author. E-mail: hayder_n@yahoo.com.

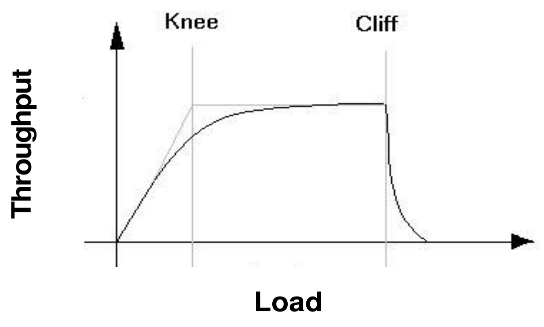


Figure 1. Throughput as a function of load (Chiu and Jain, 1998).

(Chiu and Jain, 1998). It is coded into TCP and adjusts its sending rate mechanically, according to the 'signals' the TCP receives from the network.

AIMD-based congestion avoidance and controls (Lahanas and Tsaoussidis, 2003) developed the AIMD algorithm to AIMD-FC to obtain greater efficiency and fairness than the AIMD algorithm. TCP-Jersey (Xu et al., 2004) operates based on an "available bandwidth" estimator to optimize the window size when network congestion is detected. The Packet-Pair technique (Keshav, 1991) estimates the end-to-end capacity of a path, using the difference in the arrival times of two packets of the same size travelling from the same source to the same destination. The TCP-based New-AIMD congestion avoidance and control (Hayder et al., 2008) developed the AIMD algorithm into the New-AIMD, to obtain greater efficiency and fairness than the AIMD-FC+ algorithm and evaluated the efficiency compared to AIMD-FC+ in (Lahanas and Tsaoussidis, 2003; Lahanas and Tsaoussidis, 2002).

In (Hayder et al., 2009; Hayder et al., 2010) the delay and utilization in various experiments for implementation of New-AIMD were investigated and evaluated and were found to be comparable to AIMD-FC+ (Lahanas and Tsaoussidis, 2003). In this work, we investigate and evaluate the implementation of the New-AIMD algorithm in TCP on the network, to avoid and control any congestion. We focused on delay and link utilization with different scenarios, and maintain a lower queue size, than that in the related work to reduce the delay for data transmission in the network system and to increase the link utilization.

Congestion control

It was not until 1988 that a widely accepted congestion control algorithm was finally suggested (Jacobson, 1988). 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 the Internet today, as it is proven to converge both a desirable level of efficiency as well as a desirable level of fairness among competing flows (Chiu and Jain, 1998).

In the years that followed the establishment of AIMD as the standard algorithm used in TCP, the 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 wider 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 greatly increased, experiencing only minor modifications, TCP not only survived but also became an integral constituent of the Internet. These modifications reflect the different TCP versions in-use (TCP-Tahoe, TCP-Reno, TCP-NewReno) (Jacobson, 1988; Allman et al., 1999; Floyd and Henderson, 1999), experimental TCP versions (TCP-SACK, TCP-Vegas) (Mathis et al., 1996; Barkmo and Peterson, 1995), as well as special-purpose TCP versions (T/TCP-TCP) (Braden, 1994).

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 (Chiu and Jain, 1998) 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. Below, we provide a few of the 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 (that is, 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, as the achieved throughput will diminish. For a protocol to operate in the area between the points labelled as the Knee and Cliff, a congestion control mechanism is necessary.

The AIMD system model

Chiu and Jain (1998) 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 used a "binary feedback" scheme with one bottleneck

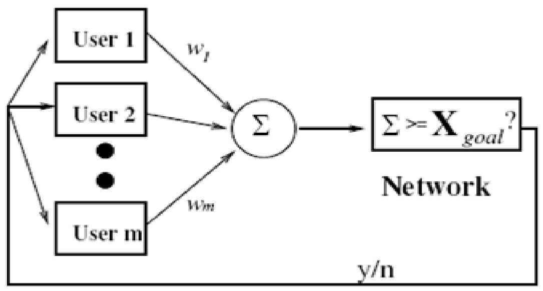


Figure 2. The control system model of m users sharing a network (Lahanas and Tsaoussidis, 2003).

router (Ramakrishnan and Jain, 1990), as shown in Figures 2 and 3. It consists of a set of m users, each of which sends data in the network at a rate 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 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 indicating 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 is received at the same time by all users. The signal is the same for all users and they take the same action when the signal arrives. The next signal is not sent until the users have responded to the previous signal. Such a system is called a synchronous feedback system or simply a synchronous system. The time that elapses between the arrival of two consecutive signals is discrete and the same after every signal arrival. This time is referred to as RTT. The system behaviour can be defined using the following time units:

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

In practice, the parameter x_{goal} is the network capacity (that is, the number of packets that the link and the router buffer can hold – or on-the-fly packets). When the aggregate flow 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 a congestion event (Jacobson, 1988).

Algorithmically, the AIMD can be expressed using the following lines:

```
AIMD (W)

1) $\alpha^i$ : 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 $\leftarrow$ W + $\alpha^i$
8) else
9) W $\leftarrow$ 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+), we can see it algorithmically with the following lines:

```
AIMD-FC+ (w,dw,q)
1) $k \leftarrow 0$
2) while (feedback is 1)
3) process (w+k)
4) $k \leftarrow k + a_i$
5) end
6) $dw \leftarrow \frac{1}{2} dw + (w + k - dw)$
7) $w \leftarrow dw + f(q)$
END
```

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

Delays

Delay and latency are similar terms that refer to the amount of time it takes a bit to be transmitted from the source to its destination (Welzl, 2005). Jitter is the delay that varies over time. One way to view latency is how long a system holds on to a packet. The system may be a single device like a router, or a complete communication system including routers and links (Welzl, 2005; Ravi, 2008). Closely related topics include bandwidth and throughput. These are illustrated in Figure 4. Bandwidth is often used to refer to the data rate of a system; however, it also refers to the width of the frequency band that a system operates in. Data rate and wire speed are

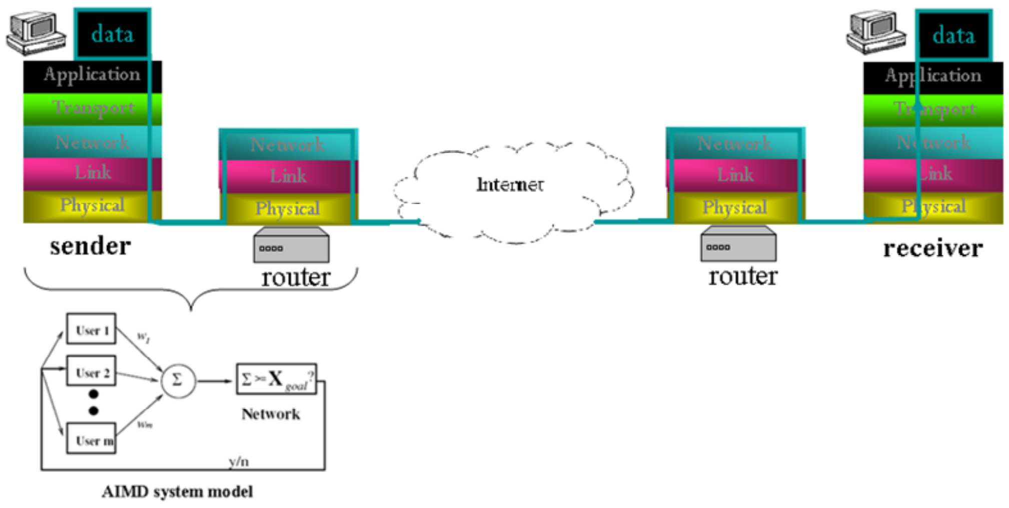


Figure 3. The AIMD system model within Layered Architecture in the Internet environment.

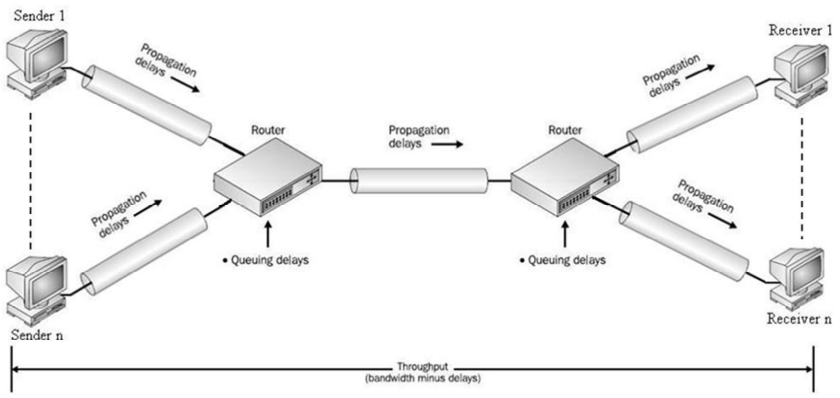


Figure 4. The relationships between bandwidth, delay, and throughput.

better terms when discussing transmitting digital information. The speed of the system is affected by congestion and delays. Throughput refers to the actual measured performance of a system when delay is considered.

Delays are caused by distance, errors and error recovery, congestion, the processing capabilities of systems involved in the transmission and other factors. The delay of distance (called propagation delays) is especially critical when transmitting data to more distant destinations. Most communications require a round-trip exchange of data, especially if the sender is waiting for an acknowledgement of receipt from the receiver. Increasing data rates allows more bits to be sent in the same amount of time; however, it does not help improve delay. Excessive delay may cause a receiving system to time out and request a retransmission. The delay factor has to be adjusted when excessive delay exists (Welzl, 2005). Delay is problematic for real-time traffic such as interactive voice calls and live video. Delay can also be a problem with time-sensitive transaction processing systems. Delay caused by congestion must be avoided; therefore, bandwidth management, priority queuing and QoS are important to ensure that enough packets get through on time. Variation in delay (jitter) is more disruptive to a voice call than the delay itself (Brakmo and Peterson, 1995; Johari and Tan, 2001).

Causes of delay

Delay is caused by hardware and software inefficiencies, as well as network congestion and transmission problems that cause errors. Delay may be caused by the following:

i. Network congestion, caused by excessive traffic.
ii. Processing delays, caused by inefficient hardware.
iii. Queuing delays, which may occur when buffers in the network devices are flooded.
iv. Propagation delays, which are related to how long

it takes a signal to travel across a physical medium (Welzl, 2005).

In this work, we will investigate and focus on the network congestion delay and queuing delay from the different causes of delays.

Congestion delay

As traffic increases on the network, congestion increases. Congestion occurs at routers and switches, causing delay that is variable (jitter). Ethernet shared mediums are prone to congestion. A user must wait if the cable is being used and collisions occur if two people try transmitting at the same time. Both users wait and then try again, causing further delay for the end-user application (end-to-end delay) (Jahwan et al., 2005; Wang et al., 2006). When a TCP/IP host begins to transmit, it has no way to monitor the network for downstream congestion problems. The host cannot immediately detect that a router is becoming overburdened. Only when the sender is forced into retransmitting dropped packets, does it sense that the network must be busy and then start to slow down its transmissions. Several techniques (Welzl, 2005) have been developed to resolve congestion problems on TCP/IP networks, such as slow start and congestion avoidance. Congestion controls help hosts adapt to traffic conditions. A transmission starts slowly and builds up until congestion is detected.

Network utilization

Network utilization is the ratio of current network traffic to the maximum traffic that the port can handle. It indicates the bandwidth used in the network. While high network utilization indicates the network is busy, low network utilization indicates the network is idle. When network utilization exceeds the threshold under normal conditions, it will cause low transmission speed, intermittence, request delay and so on. Networks of different types or in different topology have different theoretical peek values under general conditions. However, this does not mean that the higher network utilization is better. We must make sure there is no packet loss when network utilization reaches a certain value. For a switched Ethernet, 50% network utilization can be considered as high efficiency. If using a router or hub as a core switch device in the network, the network utilization should be lower than the link bandwidth capacity to avoid the increasing collisions. Through monitoring network utilization, we can understand whether the network is idle, normal or busy (Welzl, 2005; Ravi, 2008; Lin et al., 2005).

Furthermore, a file of size f with a total transfer time of Δ on a TCP connection results in a TCP transfer throughput denoted by r and obtained from Equation (5)

$$r = f / \Delta \tag{5}$$

We can also derive the bandwidth utilization, pu, assuming that the link bandwidth is B, by Equation (6)

$$pu = r / B \tag{6}$$

In our approach, when we implement the New-AIMD mechanism, we try to get high bottleneck link utilization for link capacity (network resources) and conduct many experiments, that depend on the number of flows, using the link at the same time, to show the difference between them.

Drop tail AQM algorithm

Drop Tail (DT) is the simplest and most commonly used algorithm in the current Internet gateways, which drops packets from the tail of the full queue buffer (Aun et al., 2002). Its main advantages are simplicity, suitability to heterogeneity and its decentralized nature.

However, this approach has some serious disadvantages, such as no protection against the misbehaving or non-responsive flows (that is, flows which do not reduce their sending rate after receiving the congestion signals from gateway routers) and no relative Quality of Service (QoS).

The QoS is ideal in the traditional “best effort” Internet, in which we have some guarantees of transmission rates, error rates and other characteristics in advance. QoS is of particular concern for the continuous transmission of high-bandwidth video and multimedia information. Transmitting this kind of content is difficult using the present Internet with DT.

Generally, DT is used as a baseline case for assessing the performance of all the newly proposed gateway algorithms (Eitan et al., 2005; Aun et al., 2003).

MATERIALS AND METHODS

National Chiao Tung University network simulator

The NCTU network simulator is a high-fidelity 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, which directly uses the Linux TCP/IP protocol stack to generate high-fidelity simulation results and 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

(Wang et al. 2007).

The new approach of algorithm (New-AIMD)

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 k_1+1 Round Trip Time (RTTs) or steps. Let $x_1 + x_2 + 2k_1 \geq$ 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 $\frac{x_1}{2}+\frac{x_2}{2}+2k_1+2k_2 \geq W$ 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 $\frac{x_1}{2^2}+\frac{x_2}{2^2}+2k_1+2k_2+2k_3 \geq W$ 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:

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

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 equilibrium, that is, all flows share a fair allocation of the network resources. The algorithmical 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 ( $x_1, x_2, W$ )
1) $z \leftarrow x_1 + x_2$
2) $k \leftarrow 1$
3) $t \leftarrow 1$
4) while (feedback is 1)
5) $k \leftarrow k+1$
6) $z \leftarrow x_1 + x_2 + 2t$
7) $t \leftarrow t+1$
8) if (z >= W)
9) $x_1 \leftarrow \frac{x_1}{2}$
11) $x_2 \leftarrow \frac{x_2}{2}$
12) $z \leftarrow x_1 + x_2 + 2t$
13) $k \leftarrow k+1$
14) end if
15) end
END-AIMD
```

We used the following notation:

Z indicates used network capacity	
K	indicates numbers of RTTs
t	is the 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 $x_1+x_2+k_1=W-k_1$. Thus, the total number of packets is produced by $(1+k_1)(W-k_1)$.

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 $\frac{x_1}{2}+\frac{x_2}{2}+2k_1+k_2=W-k_2$.

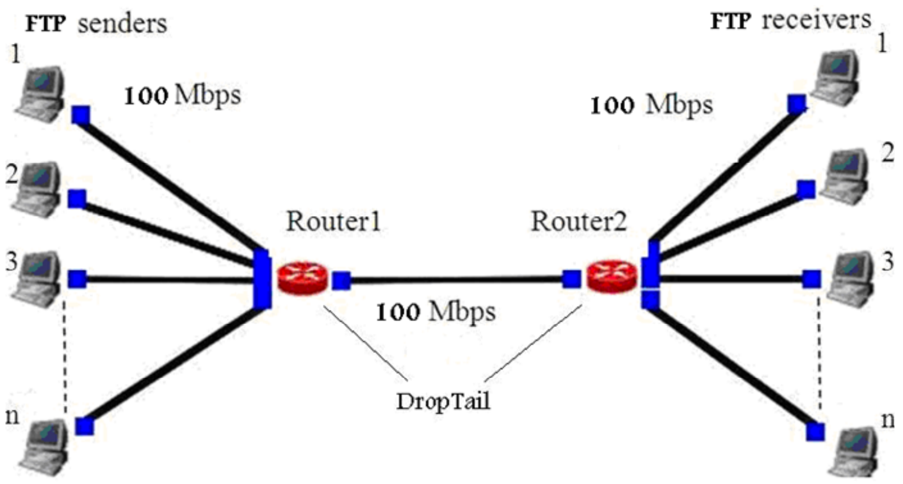


Figure 5. Multiple flow experimental set-up for AIMD evaluation.

Thus, the total number of packets is produced by $(1+k_2)(W-k_2)$.

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

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

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

But from the 1st equation we have $k_2=(W-2k_1)/4$.

And from 2nd and 3rd equations we have $k_3=\frac{1}{2}k_2$.

And from 3rd and 4th equations we have $k_4=\frac{1}{2}k_3$, $k_4=(\frac{1}{2^2})k_2$

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) and (New-AIMD) algorithms are described in detail in (Lahanas and Tsaoussidis, 2003; Hayder et al., 2008; Hayder et al., 2009; Hayder et al., 2010), respectively.

Experimental set-up and network topology

When we implement our evaluation plan on the NCTUns network simulator, multiple flows share a bottleneck link. The network topology used as a test-bed is the typical single-bottleneck two dumbbells, as shown in Figure 5. For the simulation scenario, as a general case, we use the following setup details:

The time for the experiment is fixed at 100 s. The link's capacity at the senders, receivers and bottleneck link is (full-duplex) 100Mbps and the maximum load for data transmission will be 12000 KB per time, in this bandwidth capacity for link. The mechanism of active queue management (AQM) in the gateways queues is the DropTail mechanism (DT). We used an equal number of senders and receivers 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

In the following Figures (6, 7 and 8), we supposed that the maximum data size that we want to transmit from one of the senders to the receiver is equal to 100 MB. After complete data transmission to the receiver, we can calculate the total time taken to do the data transmission; we mentioned above, about the relation between the throughput and delay. In addition, we have different cases for calculating the time and the results will depend on the number of flows in the bottleneck link at the same time.

In Figures 7 and 8, we show the comparison between our mechanism New-AIMD with AIMD and AIMD-FC+, as in the previous related work (Lahanas and Tsaoussidis, 2003). In Figure 9 we can see the comparison between using our approach New-AIMD algorithm and the AIMD, AIMD-FC+ algorithms in previous related work.

DISCUSSION

In this evaluation we found new results when using the New-AIMD algorithm within TCP-SACK. First, and for evaluating the end-to-end delay with the implementation of this new approach (New-AIMD) in TCP-SACK, and after the implementation of this experiment, we found that

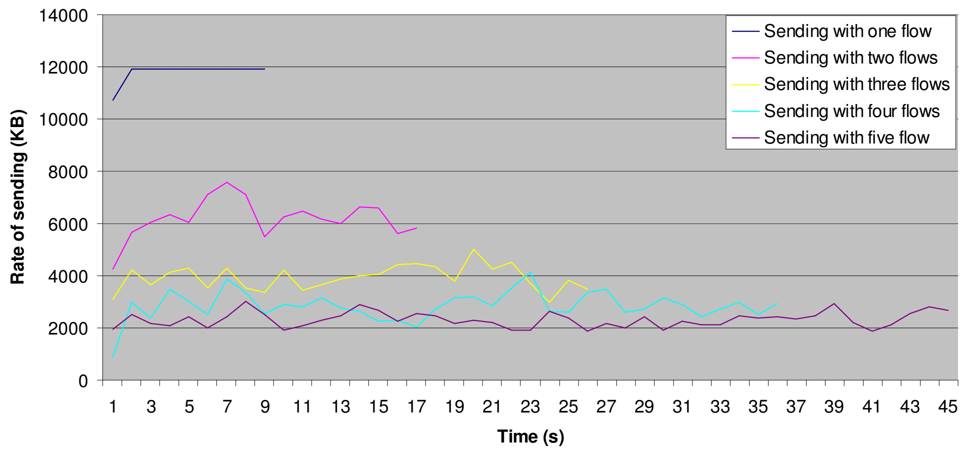


Figure 6. Throughput (KB) vs. time (s) for transmit 100 MB with varying number of flows of TCP-SACK with New-AIMD.

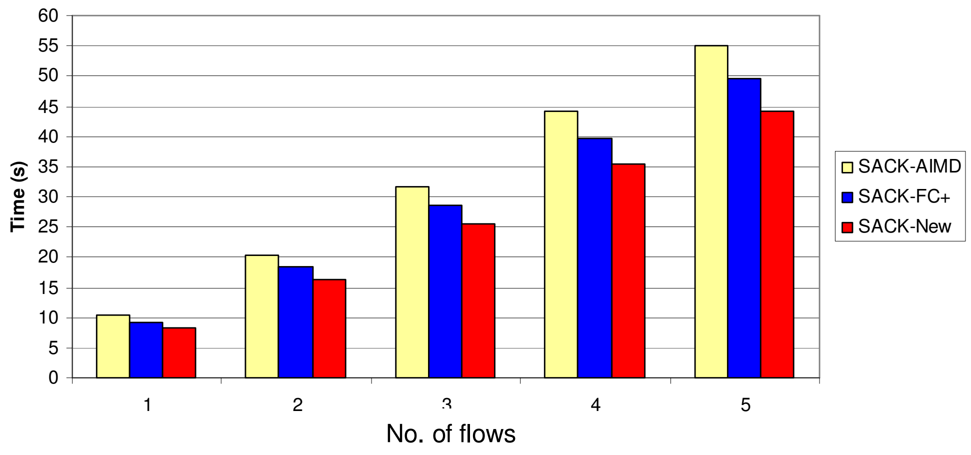


Figure 7. The time (sec) needed to transmit the 100 MB depending on the number of flows (1, 2, 3, 4 and 5) of TCP-SACK with AIMD, AIMD-FC+ and New-AIMD algorithms.

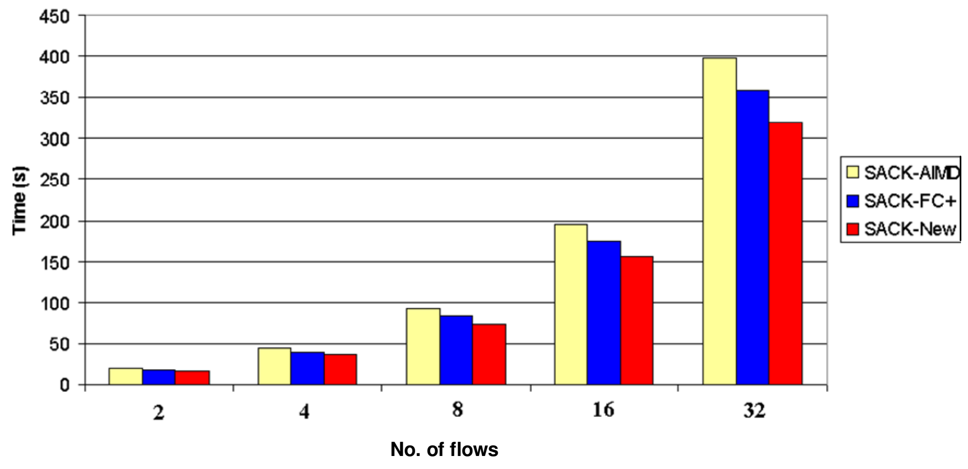
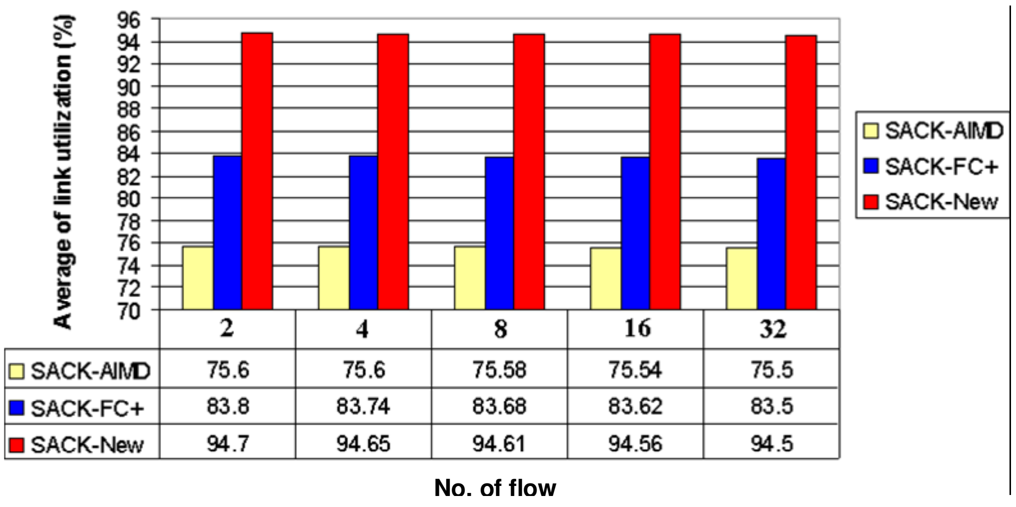
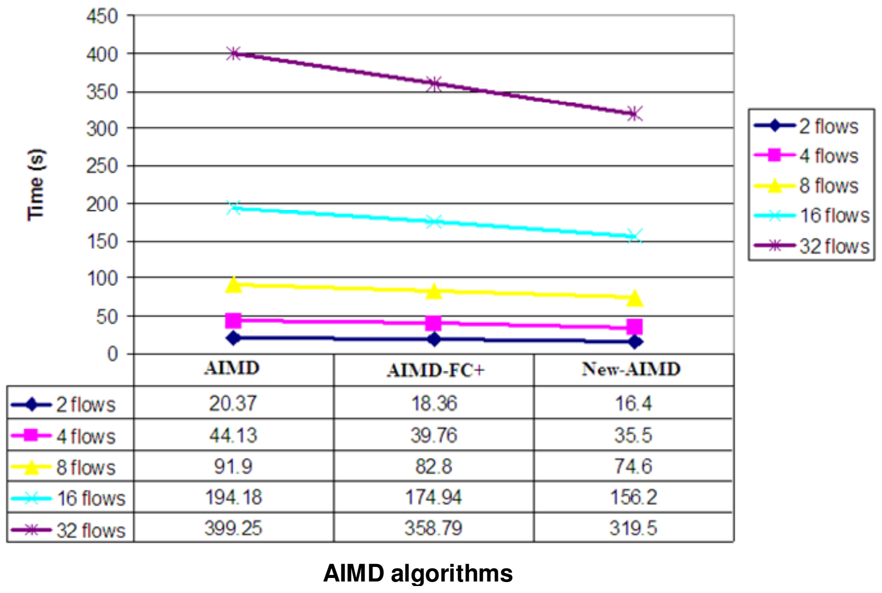


Figure 8. The time (sec) needed to transmit the 100MB depending on the number of flows (2, 4, 8, 16 and 32) of TCP-SACK with AIMD, AIMD-FC+ and New-AIMD algorithms.



	2	4	8	16	32
☐ SACK-AIMD	75.6	75.6	75.58	75.54	75.5
■ SACK-FC+	83.8	83.74	83.68	83.62	83.5
■ SACK-New	94.7	94.65	94.61	94.56	94.5

Figure 9. Link utilization for bottleneck link using AIMD, AIMD-FC+ and New-AIMD within TCP-SACK.



	AIMD	AIMD-FC+	New-AIMD
◆ 2 flows	20.37	18.36	16.4
■ 4 flows	44.13	39.76	35.5
▲ 8 flows	91.9	82.8	74.6
× 16 flows	194.18	174.94	156.2
✳ 32 flows	399.25	358.79	319.5

Figure 10. The averages of time need to transmit (100MB) with AIMD, AIMD-FC+ and New-AIMD algorithms within TCP-SACK.

the time that the network needs to transmit the complete data that we want to send (100 MB) from the sender to the receiver will be less than the time that the other AIMD algorithms need, to transmit the same data by around 12% less. This is because in this approach we solved the problem of congestion delay, which led to a reduction in the waiting time in the queues, thereby reducing the time required to transfer the data.

Second, we achieved more than 94% utilization for the bottleneck link; this means we achieved 11% higher utilization than in the previous related work (Lahanas and Tsaoussidis, 2003). In the utilization in this evaluation, we can observe the difference between the link utilization when we add new numbers of flows to the experiment. This is because of the full-duplex links between the nodes of the network. Additionally, the utilization will be less when we add more flows, as the ACK of Transmission Control Protocol (TCP) from the receivers also need space in the link to send it to, the senders at the same time as the senders send other data to the receivers.

Figures 10 and 11, show the difference between the averages of the performance for the algorithms and the value of evaluation for enhancement of the algorithm. Figure 10 is dependent on the evaluation of delay for different number of flows. Figure 11 is dependent on the

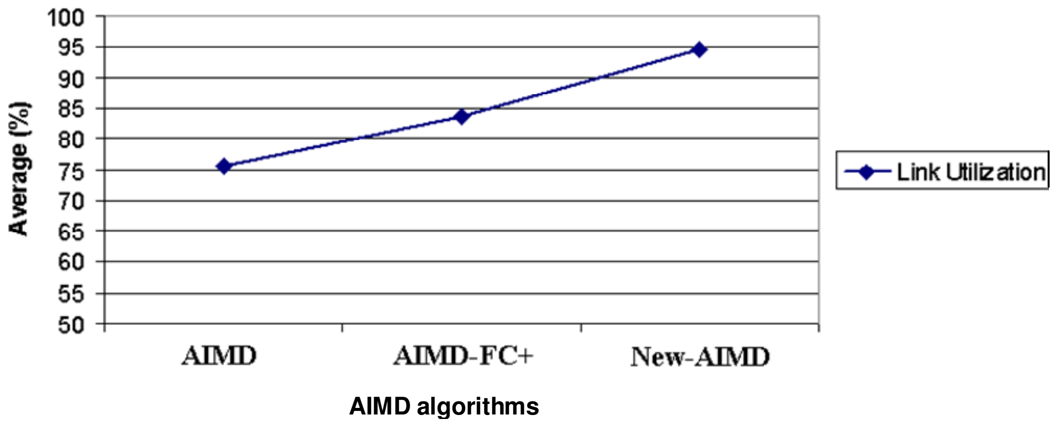


Figure 11. The average Link Utilization for AIMD, AIMD-FC+ and New-AIMD algorithms within TCP-SACK.

evaluation of link utilization. The average is from 20 experiments with minimal statistical deviation. The number of flows in these experiments is (2, 4, 8, 16 and 32).

Conclusion

In the experiments for this work, we investigated two types of performance metrics, the first one is end-to-end congestion delay; in which we found that, the results after implementing the New-AIMD algorithm were better than the results in the previous work, because the delay was less when we measured the delay depending on the throughput for the entire system and we obtained end-to-end delay of approximately 12% less than the delay with AIMD-FC+ (Lahanas and Tsaoussidis, 2003).

In addition, in these experiments we have evaluated the utilization of bottleneck link for the New-AIMD mechanism. We found the results after implementing the New-AIMD algorithm and achieved high utilization (more than 94%) for the link and avoided the congestion in this experiment for multi flows, using the same link at the same time. This means we achieve 11% higher utilization than the utilization with the AIMD-FC+ (Lahanas and Tsaoussidis, 2003).

It can be said that this mechanism works well under the conditions for the network experiments above. In addition, the benefit from implementing the New-AIMD algorithm in this study is that, it reduces the average queue length, thereby decreasing the end-to-end delay and increasing the link utilization, as well as avoiding the network congestion. This can be deemed to be the major contribution in this work for this algorithm.

REFERENCES

Allman M, Paxson V, Stevens W (1999). TCP Congestion Control. RFC 2581.
Attie P C, Lahanas A, Tsaoussidis V (2003). Beyond AIMD Explicit fair-share calculation. In Proceedings of ISCC 2003.
Aun H, Harsha S, Krzysztof P, Michael JF (2003). Congestion Control Algorithms in High Speed Telcommunication Networks. Department of Electrical and Electronic Engineering.
Braden RT (1994). T/TCP-TCP Extensions for Transactions, Functional Specification. RFC 1644.
Brakmo L, Peterson L (1995). TCP Vegas: End to end congestion avoidance on a global Internet. IEEE J. on Selected Areas in communi. 13(8):1465-1480.
Casetti C, Gerla M, Mascolo S, Sansadidi MY, and Wang R (2002). TCP Westwood: End-to-End Congestion Control for Wired/Wireless Networks. Wireless Networks J., 8: 467-479.
Chiu D, Jain R (1998). Analysis of the Increase/Decrease Algorithms for Congestion Avoidance in Computer Networks. J. Comput. Networks and ISDN. 17(1): 1-14.
Eitan A, Chadi B, Victor MR (2005). Analysis of AIMD Protocol Over Path With Variable Delay. J. Comput. Networks, 48(6): 960-971.
Floyd S, Henderson T (1999). The New Reno Modification to TCP's Fast Recovery Algorithm. RFC 2582.
Handley M, Floyd S, Pahdye J, Widmer J (2003). TCP Friendly Rate Control (TFRC): Protocol Specification. RFC 3448.
Hayder NJ, Zuriati A, Mohamed O, Shamala S (2008). The TCP-Based New AIMD Congestion Control Algorithm. Int.ernational J. Comput. Sci. Network Security, 8(10): 331-338.
Hayder NJ, Zuriati AZ, Mohamed O, Shamala S (2009). Experimental Evaluation of Bottleneak Link Utilization with the New-Additive Increase Multiplicative Decrease Congestion Avoidance and Control Algorithm. J. Comput. Sci., 5(12): 1058-1062.
Hayder NJ, Zuriati AZ, Mohamed O, Shamala S (2010). The Delay with New-Additive Increase Multiplicative Decrease Congestion Avoidance and Control Algorithm. Inf. Technol. J., 4: 1327-1335.
Jacobson V (1988). Congestion avoidance and control. in Proceeding of ACM SIGCOMM 88.
Jahwan K, Seongjin A, Jin WC (2005). A Comparative Study of Queue, Delay, and Loss Characteristics of AQM Schemes in QoS-Enabled Networks. Comput. Inf. J., 22: 1001-1019.
Johari R, Tan DKH (2001). End-to-end congestion control for the Internet: delays and stability. IEEE/ACM Transactions on Networking, 9(6): 818-832.
Keshav S (1991). Congestion Control in Computer Networks. Ph.D. Thesis, UC Berkeley, published as UCB TR 91/649.
Lahanas A, Tsaoussidis V (2002). Additive Increase Multiplicative Decrease-Fast Convergence (AIMD-FC). In Proceedings of the Networks 2002. August, Atlanta, Georgia.
Lahanas A, Tsaoussidis V (2003). Exploiting the Efficiency and Fairness Potential of AIMD-based Congestion Avoidance and Control. J. Comput. Networks, 43: 227-245.
Lin C, Xuemin S, Jon WM, Jianping P (2005). Performance Analysis of

AIMD-Controlled Multimedia Flows in Wireless/IP Networks. Department of Electrical & Computer Engineering, University of Waterloo.

Mathis M, Mahdavi J, Floyd S, Romanow A (1996). TCP Selective Acknowledgement Options. RFC 2018.

Ramakrishnan K, Jain R (1990). A Binary Feedback Scheme for Congestion Avoidance in Computer Networks with a Connectionless Network Layer. ACM Transactions on Comput. Syst., 8(2): 158-181.

Ravi SP (2008). An Evolutionary Approach to Improve End-to-End Performance in Tcp/Ip Networks. Ph.D. Thesis, College of Computing, Georgia Institute of Technology.

Tsaoussidis V, Zhang C (2002). TCP-Real: Receiver-Oriented Congestion Control. Comput. Networks J. (Elsevier), 40(4): 1-15.

Wang L, Cai L, Liu X, Shen X (2006). AIMD Congestion Control: Stability, TCP-friendliness, Delay Performance. Tech. Rep. 1-11.

Wang SY, Chou CL, Lin CC (2007). The design and implementation of the NCTUns network simulation engine. Science Direct, Simul. Modeling Pract. Theory, 15: 57-81.

Welzl M (2005). Network Congestion Control Managing Internet Traffic. 1st edition, John Wiley and Sons Ltd, ISBN-10: 0-470-02528-X.

Xu K, Tian Y, Ansari N (2004). TCP-Jersey for wireless IP communications. IEEE JSAC, 22(4): 747-756.