http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 9 (7): 1327-1335, 2010
ISSN 1812-5638

The Delay with New-Additive Increase Multiplicative Decrease Congestion Avoidance and Control Algorithm

Hayder Natiq Jasem, Zuriati Ahmad Zukarnain, Mohamed Othman and Shamala Subramaniam
Department of Communication Technology and Networks, Faculty of Computer Science and Information Technology, University Putra Malaysia, 43400 UPM, Serdang, Selangor, Malaysia

Abstract: As the Internet becomes increasingly heterogeneous, the issue of congestion control becomes ever more important. And the queue length and end-to-end (congestion) delays are some of the important things in term of congestion avoidance and control mechanisms. In this research we continued to study the performances of the New-Additive Increase Multiplicative Decrease (AIMD) algorithm as one of the core protocols for TCP congestion avoidance and control mechanism, we want now to evaluate the effect of using the New-AIMD algorithm to measure the queue length and end-to-end delays and we will use the NCTUns simulator to get the results after make the modification of the mechanism. And we will use the Droptail mechanism as Active Queue Management (AQM) in the bottleneck router. After implementation of our new approach with different number of flows, we will measure the delay for two types of delays (queuing delay and end-to-end delay), we expect the delay will be less with using our mechanism comparing with the mechanism in the previous study. Now and after got this results as low delay for bottleneck link case, we know the New-AIMD mechanism work as well under the network condition in the experiments.

Key words: Congestion control, TCP, AIMD, queue length, end-to-end delay

INTRODUCTION

End-to-end congestion avoidance and control as well as fair network resource management would have had great benefit had the TCP sender known of the behavior of the bottleneck queue and the delay in this queue.

Several methodologies have been developed to estimate bandwidth and bottleneck queue based on temporary measurements of throughput, inter-packet gap, or RTT. For example, TFRC (Floyd *et al*., 2007) calculates throughput via a throughput equation that incorporates the loss event rate, round-trip time and packet size. TCP-Vegas (Brakmo and Peterson, 1995) estimated the level of congestion using throughput-based measurements.

TCP-Vegas demonstrated that measurement-based window adjustments is a viable mechanism, however, the corresponding estimators can be improved. In TCP-Westwood (Caseti *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 (Chiu and Jain, 1989). It is coded into TCP and adjusts its sending rate mechanically, according to the signals TCP gets from the network.

AIMD-based congestion avoidance and controls (Lahanas and Tsaoussidis, 2003) developed the AIMD algorithm to AIMD-FC to get more 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 traveling from the same source to the same destination. The TCP-based New-AIMD congestion avoidance and control (Jasem *et al*., 2008) developed the AIMD algorithm into the New-AIMD to get more efficiency and fairness than the AIMD-FC+ algorithm and evaluated the efficiency compared to AIMD-FC+ in

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

(Lahanas and Tsaoussidis, 2003, 2002). Jasem *et al.* (2009a) investigated the fairness of New-AIMD and evaluated it compared to AIMD-FC+ (Lahanas and Tsaoussidis, 2003). And now in this study we want to investigate and evaluate the implementations of the New-AIMD algorithm in TCP on the network to avoid and control any congestion and to keep the queue size less than the queue size in the related work to reduce the delay for data transmission in the network system.

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, since it is proven to converge on both a desirable level of efficiency as well as a desirable level of fairness among competing flows (Chiu and Jain, 1989).

In the years that followed the establishment of AIMD as the standard algorithm to be 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 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 the different in-use TCP versions (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; Brakmo and Peterson, 1995), as well as special-purpose TCP versions (T/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, 1989) 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 give a few important points made in this study.

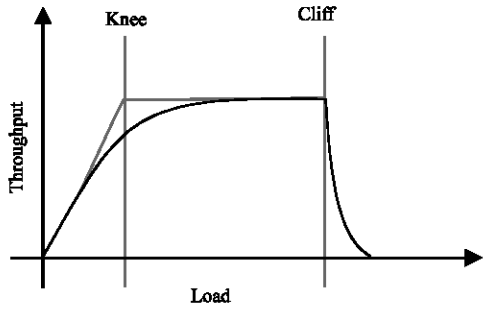

Fig. 1: Throughput as a function of load

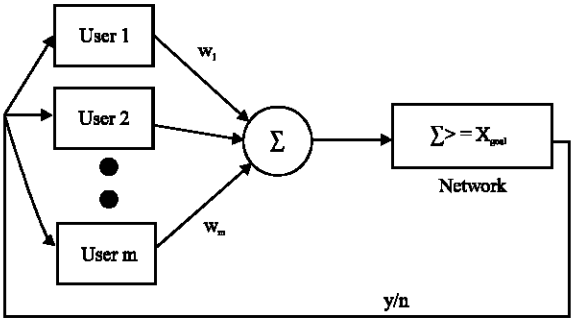

Fig. 2: A control system model of m users sharing a network

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 Fig. 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.

The AIMD system model: Chiu and Jain (1989) 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 router (Ramakrishnan and Jain, 1990), as shown in Fig. 2. It consists of a set of m users, each of which sends 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 for 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 elapsed 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 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)

In practice, the parameter X_{goal} is the network capacity (i.e., 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 with the following lines:

```
AIMD ()
αi  : Constant = Packet-size()
W  : Integer//congestion window
repeat forever
      send W bytes in the network
      receive ACKs
      if W bytes are ACKed
            W ← W + αi
      else
            W ← W/2
      end
END-AIMD
```

The New-AIMD: Let us assume network capacity (Window size or X_{goal}) is W. For Simplicity let us assume we have two flows system f1 and f2. Initially let flows f1 and f2 contain x_1 and x_2 window, respectively. With out loss of generality we assume that $x_1 < x_2$ and $x_1 + x_2 \leq W$ furthermore, we are assuming that system converges to fair in m cycle, K is the number of RTT. In 1st cycle Pseudocode is given by total flow is:

$$x_1 + x_2 + 2k_1 \quad (1)$$

In AIMD also is $x_1 + x_2 + 2k_1$

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

$$\frac{x_1}{2} + \frac{x_2}{2} + 2k_1 + 2k_2 \quad (2)$$

In AIMD is:

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

Obviously 2nd cycle contains k_2+1 RTT. Let

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

then system gives 0 feedback. Obviously we will use decrease step. In 3rd cycle Pseudocode is given by total flow is:

$$\frac{x_1}{2^2} + \frac{x_2}{2^2} + 2k_1 + 2k_2 + 2k_3 \quad (3)$$

In AIMD is

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

Here, 3rd cycle contains k_3+1 RTTs. Let:

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

then system gives 0 feedback. Obviously we will use decrease step. Similarly at mth cycle we have total flow is:

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

In AIMD is

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

Suppose mth cycle points to equilibrium that is all flows share fair allocation of resources.

The algorithmic approach when initial window size of 2 flows and Window size are x_1, x_2, respectively, is given by:

```
AIMD (x1, x2, W)
z = x1+x2//z denotes used Capacity of Network.
k = 1, t = 1/k denotes numbers of RTTs
while (1)
{
k = k + 1
z = x1+x2+2t
t = t + 1
if (z >= W)
{
x1 = x1/2
x2 = x2/2
z = x1+x2+2t
k = k + 1
}
}
END-AIMD
```

Total number of packets in various cycles:
In 1st cycle, total number of packets is given by: $(k_1+1)(x_1+x_2+2k_1)$, But from 1st cycle we have $x_1+x_2+2k_1 = W$. Therefore, $x_1+x_2+k_1 = W-k_1$.
Thus total number of packet is given by $(1+k_1)(W-k_1)$.
In 2nd cycle, total number of packets is given by:

$$(1+k_2)(\frac{x_1}{2} + \frac{x_2}{2} + 2k_1 + k_2)$$

But from 2nd cycle we have:

$$(\frac{x_1}{2} + \frac{x_2}{2} + 2k_1 + 2k_2) = W$$

Therefore

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

Thus total number of packets is given by: $(1+k_2)(W-k_2)$.
Similarly in 3rd cycle, total number of packets is given by: $(1+k_3)(W-k_3)$.
Similarly in mth cycle, total number of packets is given by: $(1+k_m)(W-k_m)$.
Thus total number of packets in all cycles is given by: $(1+k_1)(W-k_1) + (1+k_2)(W-k_2) + (1+k_3)(W-k_3) + ... +(1+k_3)(W-k_3)$

But from Eq. 1 we have: $k_2 = (W-2k_1)/4$.
From Eq. 2 and 3 we have: $k_3 = 1/2k_2$.
From Eq. 3 and 4 we have:

$$k_4 = \frac{1}{2}k_3,\ k_4 = (\frac{1}{2^2})k_2 \quad \text{Thus} \quad k_m = (\frac{1}{2^{m-2}})k_2 \text{ for } m = 3$$

The Additive Increase/Multiplicative Decrease (AIMD) and (New-AIMD) algorithms are described in more details by Lahanas and Tsaoussidis (2002, 2003), Natiq *et al.* (2008) and Jasem *et al.* (2008, 2009a, b).

DELAYS

Delay and latency are similar terms that refer to the amount of time it takes a bit to be transmitted from source to destination (Welzl, 2005).

Jitter is delay that varies over time. One way to view latency is how long a system holds on to a packet. That system may be a single device like a router, or a complete communication system including routers and links (Welzl, 2005; Prasad, 2008).

Closely related topics include bandwidth and throughput. These are illustrated in Fig. 3. Bandwidth is often used to refer to the data rate of a system, but it appropriately refers to the width of the frequency band that a system operates in. Data rate and wire speed are better terms when talking about 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.

Delay of distance (called propagation delays) is especially critical when transmitting data to farther destinations. Most communications require a round-trip

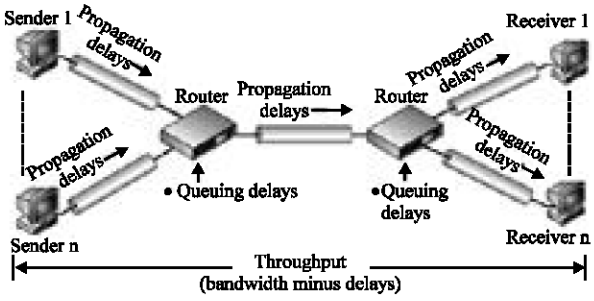

Fig. 3: The relationships between bandwidth, delay and throughput

exchange of data, especially if the sender is waiting for an acknowledgment of receipt from the receiver. Increasing data rates allows you to send more bits in the same amount of time, but it doesn't 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 like 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, so 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:

- Network congestion, caused by excessive traffic
- Processing delays, caused by inefficient hardware
- Queuing delays occur when buffers in network devices are flooded
- Propagation delay is related to how long it takes a signal to travel across a physical medium (Welzl, 2005)

In this study we will investigate and focus on the network congestion delay and queuing delay from 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 medium is 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) (Koo *et al*., 2003).

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 get a 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.

Queuing delay: After a router receives and examines a packet, it sends the packet to a buffer where it is queued up for transmission, usually on a first-in, first-out (FIFO) basis. Routers receive packets from many different sources, so the devices can easily be overwhelmed.

Buffers start to fill up when the network gets busy. Traffic may move into a queue faster than it can be moved out. If packets are delayed long enough, the source systems may begin retransmitting packets under the assumption that packets have been lost. This adds to network congestion and delay (Cai *et al*., 2005; Wang *et al*., 2006).

As mentioned, queues are usually processed on a first-come, first-served basis. Priority queuing techniques give some packets precedence over others. Packets may be marked or tagged in advance so that they are directed into a queue that matches their priority. Alternatively, a device may examine packet contents to determine priority (Welzl, 2005; Wang *et al*., 2006; Altman *et al*., 2005).

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. 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 (i.e., 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 idea 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 on the present Internet with DT.

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

NATIONAL CHIAO TUNG UNIVERSITY NETWORK 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 (Wang *et al*., 2007).

METHODOLOGY AND TOOLS

We have implemented our evaluation plan on the NCTUns network simulator. The network topology used as a test-bed is the typical single-bottleneck dumbbell, as shown in Fig. 4.

For the simulation scenario as a general case we will have the following setup details:

The link's capacity at the senders, receivers and bottleneck link is 5 Mbps. We used an equal number of senders and receivers nodes. All DT queues have 100-packet lengths. The link distance between nodes is 3000 m. And we will use the TCP-SACK with New-AIMD to evaluate the algorithm performance.

DISCUSSION OF THE SIMULATION RESULTS

Results of the first experiment: In the following figures (Fig. 5, 6), we supposed the maximum data size that we want to transmit it from the sender to the receiver is equal to 20000 KB. After complete data transmit to the receiver we can calculate the total time that takes to do the data transmission, as we mentioned above about the relation between the throughput and delay. Also, we have different cases for calculating the time and the results will depend on the number of flows in the bottleneck (1, 2, 3, 4, or 5 flows) at the same time. Also in the Fig. 6 we shows the comparison between our mechanism New-AIMD and AIMD-FC+ in the previous related work (Lahanas and Tsaoussidis, 2003).

The results for the time needed to transmit the data were less than the expected time needed if we implement AIMD, as in the related work, by around 12% less, which means the end-to-end delay was less as well.

Results of the second experiment: In Fig. 7a, b-10a, b, we observe the behavior of the queue using our mechanism

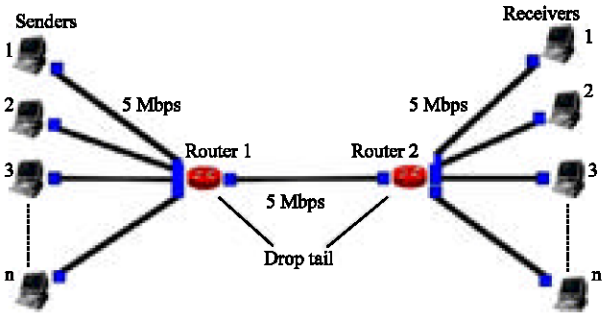

Fig. 4: Multiple flow experimental set-up for New-AIMD evaluation

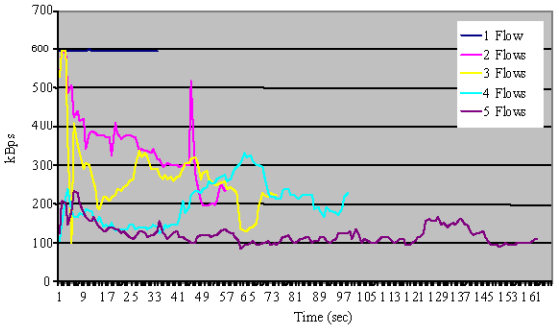

Fig. 5: Throughput (kB) vs. time (sec) for transmitting 20000 kB with varying number of flows of TCP-SACK with New-AIMD

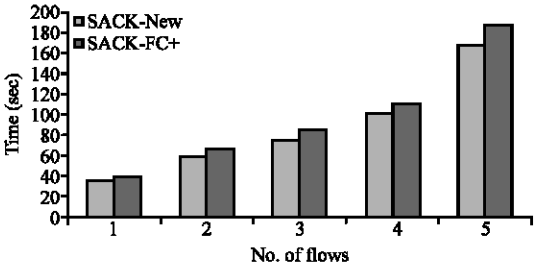

Fig. 6: The time (sec) needed to transmit the 20000 kB depending on the number of flows of TCP-SACK with AIMD-FC+ and New-AIMD

(New-AIMD). In this experiment, we measure every RTT. We show that this mechanism works very well, under the given network conditions. And the results will depend on the number of flows, the time for the packet waiting in the queue as the average delay time (m sec) and on the queue length in simulation time unit (s). In. Fig. 7-10, we separated the results depend on the number of flows in the experiments (2, 3, 4, or 5 flows) sequentially and we will put the result in two parts, part a: to show the results for

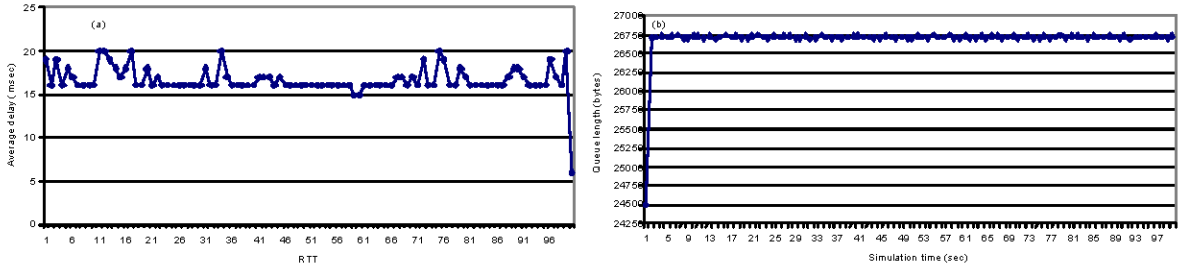

Fig. 7: (a) Average delays vs. RTT for two flows and (b) queue length vs. Simulation time for two flows

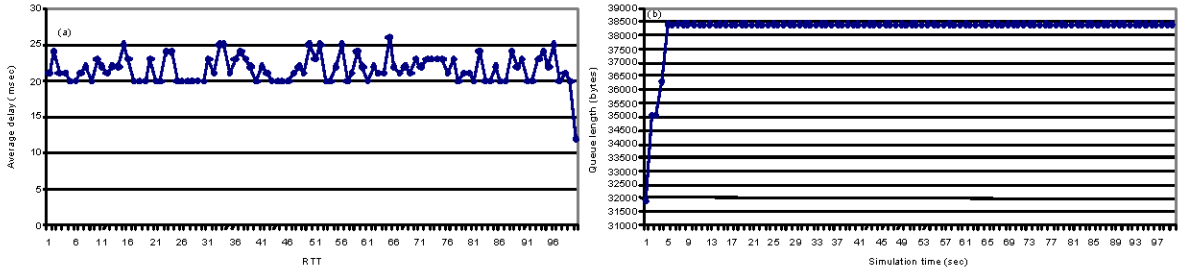

Fig. 8: (a) Average delays vs. RTT for three flows and (b) queue length vs. Simulation time for three flows

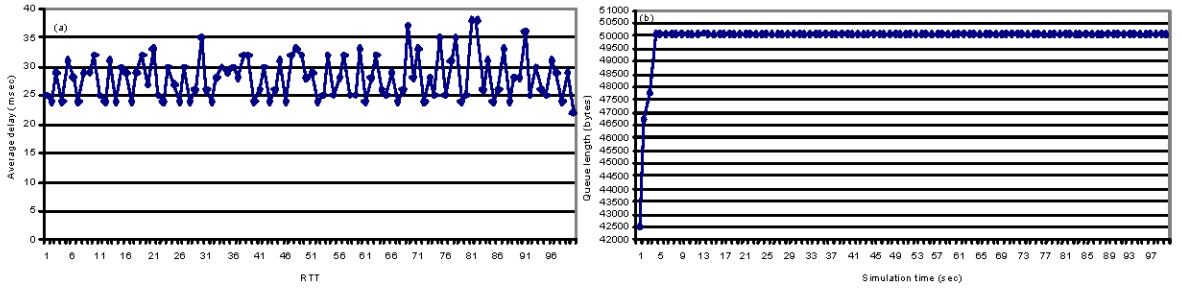

Fig. 9: (a) Average delays vs. RTT for four flows and (b) queue length vs. Simulation time for four flows

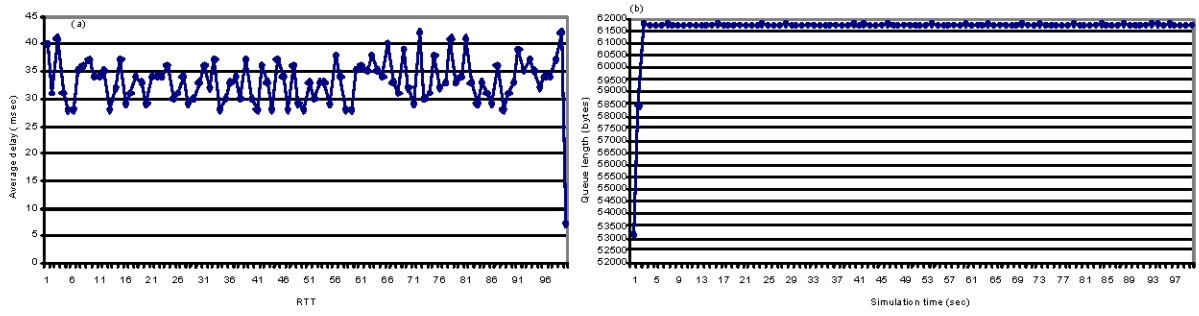

Fig. 10: (a) Average delays vs. RTT for five flows and (b) queue length vs. simulation time for five flows

average delay (msec) with the RTT, part b: to show the results for the queue length (bytes) with the simulation time (sec).

CONCLUSION

In the experiments of this study, we investigated about two types of delay, the first one is end-to-end congestion delay and we found the results after implementing the New-AIMD algorithm were better than the results in the previous work, because the delay was less when we measure the delay depend on the throughput for all the system and we got end-to-end delay at around 12% less. And we measured the second type of delay, a queuing delay and also the queue length to discover the bottleneck queue behavior.

Then we can say this mechanism work as well under the conditions for network experiments above.

And also we can say that the benefit from implementing the New-AIMD algorithm in this study is to reduce the average queue length in order to decrease the end-to-end delay, beside of avoid the network congestion as the major work for this algorithm as we studied it in previous studies about the efficiency and fairness for this mechanism.

REFERENCES

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

Altman, E., C. Barakat and V.M. Ramos, 2005. Analysis of AIMD protocol over path with variable delay. Comput. Networks, 48: 960-971.

Attie, P.C., A. Lahanas and V. Tsaoussidis, 2003. Beyond AIMD explicit fair-share calculation. Proceedings of 8th IEEE Symposium on Copmuters and Communication, June 30-July 03, Turke, pp:727-727.

Braden, R., 1994. T/TCP-TCP Extensions for Transactions Functional Specification. RFC., United States.

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.

Cai, L., X. Shen, J.W. Mork and J. Pan, 2005. Performance analysis of AIMD-controlled multimedia flows in wireless/IP networks. Department of Electrical and Computer Engineering, University of Waterloo. http://en.scientificcommons.org/43195596.

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.

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., M. Handley, J. Padhye and J. Widmer, 2007. TCP Friendly Rate Control (TFRC): Protocol specification. RFC 3448. http://www.icir.org/floyd/talks/tfrcbis-dec07.pdf.

Haider, A., H. Sirisena, K. Pawlikowski and M.J. Ferguson, 2003. Congestion control algorithms in high speed telcommunication networks. Department of Electrical and Electronic Engineering. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.102.248

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. Fairness of the TCP-based new AIMD congestion control algorithm. J. Theory Applied Inform. Technol., 5: 568-576.

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

Johari, R. and D.K.H. Tan, 2001. End-to-end congestion control for the internet: Delays and stability. IEEE/ACM Trans. Networking, 9: 818-832.

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

Koo, J., S. Ahn and J. Chung, 2003. A comparative study of queue, delay and loss characteristics of AQM schemes in QoS-enabled networks. Comput. Informatics, 22: 1001-1019.

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 Acknowledgment Options. University of Edinburgh, Scotland, UK.

Prasad, R.S., 2008. An evolutionary approach to improve end-to-end performance in Tcp/Ip networks. Ph.D. Thesis, College of Computing, Georgia Institute of Technology

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, L., L. Cai, X. Liu and X. Shen, 2006. AIMD congestion control: Stability, TCP-friendliness, delay performance. Technical Report.

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.