Volume 2, Issue 2, February 2012 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Analysis for Go-Back-N ARQ Protocols on Multi Channels by using AIMD Congestion Control Algorithm

[1]P.Radha Krishna Reddy M .Tech Student
[2]Ashim Roy M.Tech Student
[3]G.Sireesha Assistant Professor
[4]K.Pushpa Rani Associate Professor

Abstract: **In this paper we proposed to improve channels transmission rate but different time-invariant error rates. By assuming the Gilbert–Elliott model (GEM) for each channel and TCP for high speed. The additive increase/multiplicative-decrease (AIMD) algorithm is a feedback control algorithm best known for its use in TCP Congestion Avoidance. AIMD combines linear growth of the congestion window with an exponential reduction when congestion takes place. Multiple flows using AIMD congestion control will eventually converge to use equal amounts of a contended link. The related schemes of multiplicative-increase/multiplicative-decrease (MIMD) and additive-increase/additive-decrease (AIAD) do not converge. We extend our analysis to time-varying channels. We compute the probability mass functions of the sequencing buffer occupancy and the sequencing delay for time-invariant channels. Our approach is based on the logarithm of the window size evolution has the same behaviour as the workload process in a standard G/G/1 queue. The Laplace-Stieltjes transform of the equivalent queue is then shown to directly provide the throughput of the congestion control algorithm (CCA) and the higher moments of the window size.**

Keywords— ***AIMD*, GEM, TCP, CCA, PGF, PMF.**

1. INTRODUCTION:

Go-Back-N ARQ is a specific instance of the automatic repeat request (ARQ) protocol, in which the sending process continues to send a number of frames specified by a window size even without receiving an acknowledgement (ACK) packet from the receiver. It is a special case of the general sliding window protocol with the transmit window size of N and receive window size of 1.

The receiver process keeps track of the sequence number of the next frame it expects to receive, and sends that number with every ACK it sends. The receiver will ignore any frame that does not have the exact sequence number it expects – whether that frame is a "past" duplicate of a frame it has already ACK'ed [1] or whether that frame is a "future" frame past the last packet it is waiting for.

Once the sender has sent all of the frames in its window, it will detect that all of the frames since the first lost frame are outstanding, and will go back to sequence number of the last ACK it received from the receiver process and fill its window starting with that frame and continue the process over again.

There are a few things to keep in mind when choosing a value for N.

1: The sender must not transmit too fast. N should be bounded by the receiver's ability to process packets.

2: N must be smaller than the number of sequence numbers (if they are numbered from zero to N) to verify transmission in cases of any packet (any data or ACK packet) being dropped.

3: Given the bounds presented in (1) and (2), choose N to be the largest number possible.

The receiver window is one frame wide; on a frame error the receiver discards the frame and all subsequent fames and sends no ACKs. Eventually the senders will timeout and resend the damaged frame and all subsequent frames. This can waste a lot of bandwidth if the error rate is high.

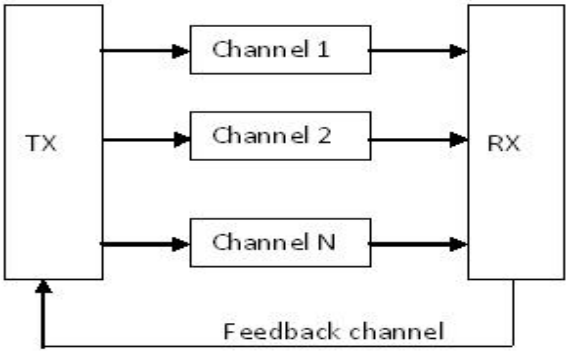


Fig 1: Basic Structure

Following figure shows a Go-back-N ARQ protocol; for simplicity the sequence numbers are shown as continuously ascending. Sequence number 2 is received with an error and discarded and all subsequent frames are discarded (sequence numbers 3 to 5). Eventually sequence number 2 timeouts and is resent together with all subsequent frames.

(a) The sender must store a message and all subsequence messages until an ACK arrives for the first,
(b) The receiver only needs to store a single frame which can then be passed to the host.

Although that are MaxSeq + 1 sequence numbers (0 to MaxSeq) only MaxSeq frames can be outstanding at any time.

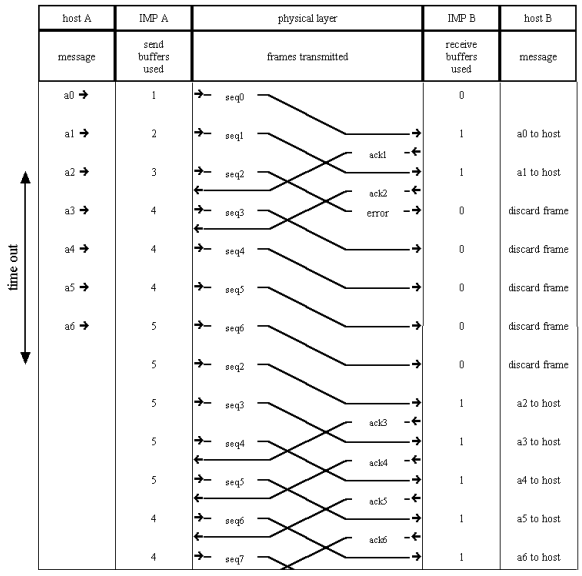


Fig. 2 Frames Transmission

Consider the case where a three bit sequence number is being used with sequence numbers ranging from 0 to 7 and we can have 8 frames outstanding:

The sender sends frames 0 to 7

The receiver receives each frame (0 to 7) in turn and passes message to the host, advances its window and sends an ACK

The ACKs for frames 0 to 7 are all lost (lightning strike on comms line)

The receiver is now expecting frame sequence number 0

After timeout the sender resends frames 0 to 7

The receiver thinks these are the next 8 frames and passes them to the host, etc.

To avoid this gap is introduced in the sequence numbers to ensure no overlap, ie:

The sender sends frames 0 to 6.

The receiver receives each frame (0 to 6) in turn and passes message to the host, advances its window and sends an ACK .

The ACKs for frames 0 to 6 are all lost (lightning strike on comms line).

The receiver now is expecting frame sequence number 7.

After timeout the sender resends frames 0 to 6 .

The receiver discards all these frames (outside its window) but sends ACKs.

The sender gets ACKs for 0 to 6 and sends frame 7, etc.

2. THE MODEL

The approach taken is to increase the transmission rate (window size), probing for usable bandwidth, until loss occurs. The policy of additive increase may, for instance, increase the congestion window by a fixed amount every round trip time. When congestion is detected, the transmitter decreases the transmission rate by a multiplicative factor; for example, cut the congestion window in half after loss. The result is a saw-tooth behavior that represents the probe for bandwidth.

AIMD requires a binary signal of congestion. Most frequently, packet loss serves as the signal; the multiplicative decrease is triggered when a timeout or acknowledgement message indicates a packet was lost. It is also possible for in-network mechanisms to mark congestion (without discarding packets) as in Explicit Congestion Notification (ECN).

Let w(t) be the sending rate (e.g. the congestion window) during time slot t, a (a > 0) be the additive increase factor, and b (0 < b < 1) be the multiplicative decrease factor.

$$w(t+1) = \begin{cases} w + a & \text{if congestion is not detected} \\ w \times b & \text{if congestion is detected} \end{cases}$$

In TCP, after slow start, the additive increase factor a is typically one MSS (maximum segment size) per round-trip time, and the multiplicative decrease factor b is typically 1/2.

3. ANALYSIS:

Consider the following discrete time stochastic recursive equation Wn+1 = max (AnWn,1)…. (1) The process, {Wn}, can be viewed as a sequence of observations of a continuous time process sampled at certain, not necessarily equal, time intervals. The sequence An 2 (0, 1) is assumed to be stationary and ergodic. Taking the logarithm of equation (3), we obtain

log[Wn+1] = max(log[An] + log[Wn], 0).

Using the substitutions Yn = log[Wn], and Un = log[An] in the above equation, we obtain

Yn+1 = max (Yn + Un , 0)…. (2)

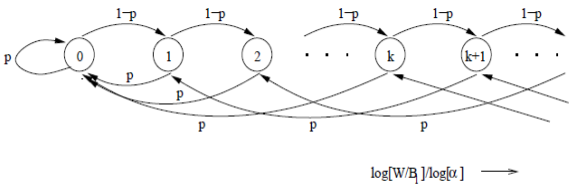


Figure 1. Transition probabilities of Y_n.

We now make the following observation: The recursive equation (4) has the same form as the equation describing the workload process in a G/G/1 queue observed at, say, just after an arrival Un denotes the 5 difference between the service time of the nth customer and the inter arrival time between the nth and the (n + 1) th customer. Since the introduced transformation, log(·), is invertible, there is a one to one correspondence between the processes {Yn, n ¸ 0} and {Wn, n ¸ 0}. This observation allows us to study the stability of the window process {Wn, n ¸ 0} via that of {Yn, n ¸ 0}. Furthermore, the analogy with queuing theory of the process {Yn, n ¸ 0} allows us to obtain the steady state moments of Wn. Theorem 3.1 Assume that E [logA0] < 0. Then there exists a unique stationary ergodic process

{W * n}, defined on the same probability space as {Wn}, that satisfies the recursion (1).Moreover, for any initial value W0 = w, there is a random time Tw, which is finite with probability 1, such that Wn = W * n for all n >= Tw. If E [logA0] > 0 then Wn tends to infinity w.p.1 for any initial value W0 = w. The log transformation allows us to obtain the moments of Wn in the stationary regime (i.e., moments of W*n) from the Laplace-Steiltjes Transform (LST) of Y*n in the stationary regime (i.e., LST of Y * n). The LST of Y * n is given by G(s) = E [e - $^{sY *n}$]…(4) which is defined for s 2 S, where S is the region of convergence of G(s). For a given integer k ¸ 0, the kth moment of W* n is obtained as follows

E[(W*n)k] = E[exp(kY *n)] = G(−k)….(5) where −k is assumed to belong to S. If −k 62 S then the corresponding moment is 1. Thus, all finite moments of W* n can be obtained from the LST of Y * n . A similar analysis can be done for the stochastic recursive equation

Wn+1 = min(AnWn,B)….(6) by making the transformation Yn = log[B]−log[Wn]. The moments of W*n can then be obtained from the LST of Y * n using the relation E[(W*n)k] = E[Bk exp(−kY *n)] = BkG(k)…..(7)All the moments of W*n are finite since G(s) is finite for s ¸ 0.The recursive equation for model (i), as given by (1), is similar to equation(3). Therefore, the analysis of this model can be done along the lines of the analysis of (3). Similarly, the analysis of models (ii) and (iii) can be done along the lines of the analysis of (6). We note that the analysis of model (iii) is similar to that of model (ii). The equivalent queuing system of model (iii) can be obtained by deleting the idle periods of the equivalent queuing system of model (ii). The throughput of the AIMD algorithm, or the first moment of the window size, under different models, can be obtained from equation (5) and (7). These two assumptions allow us to use a discrete state space, S = {0, 1, 2, ...} for Yn. Thus, Yn can be modeled as a discrete state space Markov chain. The state Yn = i corresponds to Wn = Bl®i. The transition probabilities for this model are shown in Figure 2. Let Pn(j), j 2 S, be the probability of Yn being in state j at the end of the nth RTT. The probability of being in state j at the end of the (n + 1)th RTT is given by

Pn+1(j) = (1 − p)Pn(j − 1) + pPn(j + k), j >=1 ,p Pk i=0 Pn(i), j = 0…..(8)

Denote the z-transform of Yn by Yn(z). Yn(z) is defined as Yn(z) = ∑ Pn(j)zj.(9)
j=0

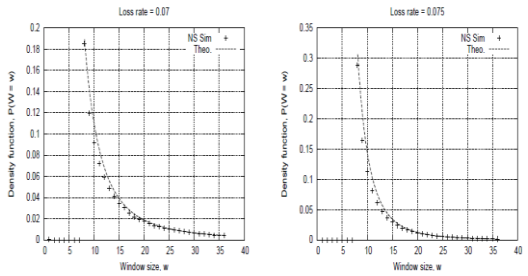
4. EXPERIMENTAL EVALUATION:

Upper Bound on Window Size and Window Dependent Random Losses: The probability of a loss in an RTT was independent of the window size in that RTT. In this section, we consider a model in which the losses in an RTT depend on the window size in that RTT. Specifically, we assume that each packet is dropped (or, equivalently, is in error) with a constant probability q. As a consequence of this assumption, the probability of packet drops in an RTT is no longer independent of the window size in that RTT. First, we present the model with window dependent losses. Then we propose an approximation to this model which will enable us to compute the throughput in the window dependent model using the expression for throughput in the window independent model (model (ii)). In each RTT, the window is reduced only once even in the presence of multiple packet drops. Loss recovery mechanisms of the recent TCP flavours such as New Reno and SACK. let Wn be the window size in the nth RTT. Let pn be the probability that the window is reduced in the nth RTT. Then, pn is given by pn = 1 − (1 − q)Wn…..(9) The window size evolution for this model can be written as Wn+1 = min(AnWn,Bu), where Bu is the upper bound on the window size, and An is now given by

An =α w.p. 1 − pn, ¯β w.p. pn.….(10)

5. SIMULATION RESULT:

Scalable TCP was proposed as a modification to the existing standard TCP for high speed networks. In the congestion avoidance phase, Scalable TCP uses the following algorithm to update the sender's window at the end of every RTT: Wn+1 = 1.01 ×Wn if no losses are detected during the nth RTT,Wn+1= 0.875 ×Wn if one or more losses are detected during the nth RTT. As mentioned in the Introduction, Scalable TCP is an instance of AIMD protocols, and therefore, we validate our models by performing simulations with Scalable TCP. The simulation are performed using Ns-2.The simulation setup has a source and a destination node. The source node has infinite amount of data to send and uses Scalable TCP with New Reno flavor. The link bandwidth is 150 Mbps and the two way propagation delay is 120 ms. The window at the source is limited to 500 packets to emulate the receiver advertised window. The BDP for this system is approximately 2250 packets (packet size is 1040 bytes). In the Scalable TCP we have implemented in ns-2, the following assumptions are made: • the minimum window size, Bl, is 8. The growth rate of Scalable TCP is very small for small window sizes. It has been use the Scalable algorithm after a certain threshold. • There is no separate slow start phase since slow start can be viewed as a multiplicative increase algorithm with α = 2. • For each positive ACK received, the window is increased by α − 1 packets. When a loss is detected, the window is reduced by a factor of β. α is taken as 1.01 and β is taken as 0.86. This value of β gives k = −log[β]/ log[α] =15. We set α and β in this way so as to be close to the values recommended in (α = 1.01, β = 0.875).



E [Wn] = 8n a/ a − n, respectively. In the simulations, the density function of W is obtained by sampling the window at an interval of RTT = 0.12s. We would like to note that the RTT is very close to the propagation delay in the present setting, and does not vary much. This results in a small discrepancy between the simulations and the theoretical function. The throughput in (TCP packets)*/RTT* as a function of the loss rate, *p*. The error bars are the 99% confidence intervals. Figure 8 shows the throughput in (TCP packets)*/RTT* as a function of the loss rate, *p*, for the model in which the maximum window at the sender is limited by the receiver's advertised window. The receiver buffer is assumed to be limited to 500 packets. The error bars are the 99% confidence intervals. A good match is observed between the simulations and the analysis

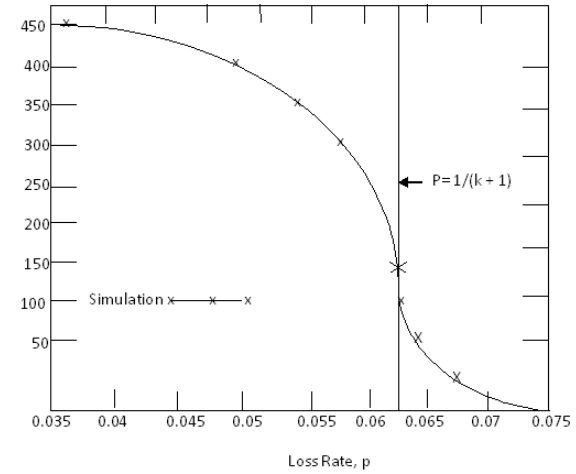


Fig 4: Throughput for maximum evaluation

two regions where model (i) and model (ii) are valid, respectively. As per approaches 1/(k + 1) from either direction, the approximate models (i) and (ii) diverge from the simulation results. However, model (i) gives a good estimate when (k + 1) p >> 1, i.e., p >> 0.625 (k = 15 in the simulations). Similarly, model (ii) gives a good approximation of the system when p << 0.625. The exact model fits well throughout the range of p. The throughput for model (i) is plotted for p ¸ 0.068 because a (in equation (18)) is > 1 for p>=0.0673.

6. CONCLUSION:

On Go-Back-N ARQ protocol logarithm of the window size process of a connection using the AIMD congestion control algorithm is equivalent to the workload process in a G/G/1queue. The throughput of the connection and the higher moments of the window size process can be computed using the Laplace-Stieltjes transform of the equivalent workload process. For window independent losses, an exact expression can be obtained for the steady state probability distribution of the window size, and the throughput of the connection. In Future For window dependent losses an approximate expression, analogues to the square root formula for standard TCP, can be used to compute the throughput as well as SISD or MISD can be applied for calculating error rate for single and multiplicative channels when selective sequential queues are approached.

ACKNOWLEDGMENT

We would to thank the anonymous referee for helpful comments.

REFERENCES

[1] M. E. Anagnostou and E. N. Protonotarios, "Performance analysis of the selective repeat ARQ protocol," *IEEE Trans. Commun.*, vol. COM-34, pp. 127–135, 1986.

[2] Z.Rosberg and N. Shacham, "Resequencing delay and buffer occupancy under the selective-repeat ARQ," *IEEE Trans. Inf. Theory*, vol. 35, pp. 166–173, 1989.

[3] M. Moeneclaey, H. Nruneel, I. Bruyland, and D. Y. Chung, "Throughput optimization for a generalized stop-and-wait ARQ scheme," *IEEE Trans. Commun.*, vol. COM-34, pp. 205–207, 1986.

[4] J. G. Kim and M. K. Krunz, "Delay analysis of selective repeat ARQ for a Markovian source over a wireless channel," IEEE Trans. Veh. Technol., vol. 49, pp. 1968–1981, 2000.

[5] L. B. Le, E. Hossain, and A. S. Alfa, "Delay statistics and throughput performance for multi-rate wireless networks under multiuser diversity," IEEE Trans. Wireless Commun., vol. 5, pp. 3234–3243, 2006.

[6] M. Rossi, L. Badia, and M. Zorzi, "SR ARQ delay statistics on N-state Markov channels with non-instantaneous feedback," IEEE Trans.Wireless Commun., vol. 5, pp. 1526–1536, 2006.

[7] D. Towsley, "A statistical analysis of ARQ protocols operating in a non independent error environment," IEEE Trans. Commun., vol. COM-29, pp. 971–981, 1981.

[8] Z. Ding and M. Rice, "Throughput analysis of ARQ protocols for parallel multichannel communications," in Proc. IEEE GLOBECOM, 2005, pp. 1279–1283.

[9] N. Shacham and B. C. Shin, "A selective-repeat-ARQ protocol for parallel channels and its resequencing analysis," IEEE Trans. Commun., vol. 40, pp. 773–782, 1992.

[10] S. Floyd. HighSpeed TCP for Large Congestion Windows. RFC 3649, Experimental, December 2003. Available at www.icir.org/floyd/hstcp.html.

[11] L. Xu, K. Harfoush, and I. Rhee. Binary Increase Congstion Control (BIC) for Fast Long-Distance Network. In Proceedings of the IEEE INFOCOM, March 2004.

[12] F. Baccelli and D. Hong. AIMD, Fairness and Fractal Scaling of TCP Traffic. In Proceedings of the IEEE INFOCOM, April 2002.
[13] G. Sireesha Analysis for ARQ Protocols on Multi Channels by using MIMD Congestion Control Algorithm.

BIBLIOGRAPHY

Mr. P. Radha Krishna Reddy received his B.Sc(CS) from Sri Venkateswara University-Tirupati. M.Sc in Computer science from Sri Venkateswara University-Tirupati, and pursuing M.Tech in Computer Science and Engineering from Vagdevi Institute of Technology and Sciences, JNTU-Anantapur.

Mr. Ashim Roy received his B.Tech in Information Technology from Siliguri Institute Of Technology in the year 2006,affiliated to West Bengal University Of Technology . Pursuing M.Tech in Information Technology from I Tech College Falakata(West Bengal) affiliated to University Of Karnataka.

Ms. G.Sireesha received her B.Tech in Computer science and Engineering from Royal Institute of Technology and Science, JNTU, Hyderabad, M.Tech in Computer science (Parallel computing) from Aurora's Engineering College, JNTU, Hyderabad.She is working as a Assistant Professor in Computer Science and Engineering in Guru Nanak Institute of Engineering & Technology, JNTU-Hyderabad.

Mrs Pushpa Rani received her B.Tech in Computer science and Engineering from JNTU, and M.Tech in Computer science and Engineering from Acharya Nagarjuna University-Guntur. Having eight years of teaching experience working in Vagdevi Institute of Technology and Sciences, JNTU- Anantapur.