International Journal of Advanced Computational Engineering and Networking, ISSN(p): 2320-2106, ISSN(e): 2321-2063
Volume-6, Issue-2, Feb.-2018, http://iraj.in

EXPOSURE OF TCP-ICCW CONGESTION MECHANISM FOR RELIABLE COMMUNICATION IN COMPUTER NETWORKS

[1]NELOFAR ASLAM, [2]KEWEN XIA, [3]M. TAFSEER HAIDER

[1,2,3]Hebei University of Technology, Tianjin 300401, China, University of Engineering and Technology, Lahore, Pakistan
E-mail: [1]nelofar.aslam@ieee.org, [2]kwxia@hebut.edu.cn, [3]taphseer@gmail.com

Abstract - The TCP/IP congestion control mechanism becomes essential to ensure the reliable communication, where data sending rate is larger than the underlying link capacity. Therefore, more sophisticated techniques are required to implement for Quality of Service (QoS) of data which ultimately holds reliable communication in different types of networks (wired/wireless). In this article, we propose a technique to extend the functionality of TCP-ICCW (Initial Constant Congestion Window) regarding throughput, data sending rate, data loss and latency or delay during congestion. The TCP-ICCW presents two different thresholds (Sµ, SF) for increasing the sending rate of the data to prevent it from link congestion. The comparative analysis done for a TCP-ICCW mechanism with the other TCP variants outperforms in data integrity and accuracy over different types of networks.

Index terms - Congestion control, QoS, wireless networks, wired networks, reliable communication.

I. INTRODUCTION

TCP protocol, a connection-oriented service which guarantees the reliable establishment of connection among computer networks, is a crucial requirement for secure node-to-node transmission. It provides the congestion control that is one of the critical issues during the data communication [1], [2]. Congestion control ensures the QoS by enhancing the throughput, reducing the delay and loss during the bottleneck (congestion).

As yet, a lot of research work has been conducted for congestion maintenance with the help of intelligent routing mechanism (TCP/IP). Park presented a geographic routing protocol for congestion avoidance. This protocol forwards the data based on the location information of the nodes [3]. Kafi et al. described a new schedule of sending the data named as REFIACC (Reliable, Efficient, Fair and Interference-Aware Congestion Control) mechanism for enhancing the throughput [4]. In [5], the author proposes a new method for Active Queue Management (AQM) to detect the early loss probability. This method uses fuzzy logic with the Proportional Integral Derivative (PID) as FuzzyPID. Kafi et al. defined an IACC (Interference-Aware Congestion Control), where congestion is controlled with the help of fair utilization of the link capacity [6]. Huang et al. also presented a congestion control algorithm to define a fixed flow of the route [7].

With the existing research work, TCP protocol further needs to enhance at its congestion detection phase. In our previous work, we proposed a TCP-ICCW (Initial Constant Congestion Window) routing protocol to detect the congestion as soon as it begins to build up [8]. The TCP-ICCW procedure utilizes the first two acknowledgments for maintaining the sending rate which works as a congestion avoidance mechanism. In this paper, an extension of our previous work has been done for considering another parameter (delay). Furthermore, the proposed protocol is now implemented in two types of computer networks (wired/wireless).

II. TCP CONTROLS CONGESTION OVER SHARED LINKS

TCP controls the congestion and offers reliable transport service among the shared links of two different hosts [9]. The following Figure 1 provides an overview of TCP connection set up between hosts A and B.

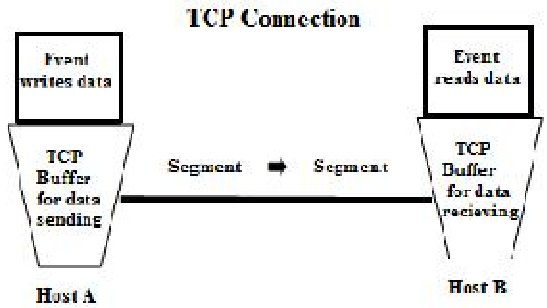

Figure 1: TCP Connection-oriented send and receive buffers.

When too many sources tend to send data at a high rate from the same shared link of a network, a situation of blockage occurs like a traffic jam on the roads. This situation is called congestion. The congestion degrades the performance of the network due to packet loss and increase in delay during the transmission. Data routing needs high energy consumption so that the retransmission of the packets can create burden [10]. Congestion may lose the high priority data rather than the lower one because of the network saturation. The continuous congestion casts the adverse effect on the application of QoS and eventually causes the low utilization of link capacity. To tackle with the congestion and its causes, TCP protocol has many variants (TCP New Reno, TCP Reno, SACK, VEGAS, FACK) and procedures which are suitable for different kinds of the network environment.

III. EXTENDED TCP-ICCW (INITIAL CONSTANT CONGESTION WINDOW) TO CONTROL THE FLOW RATE

TCP congestion control procedure usually follows three steps to handle the congestion more accurately:

- Detection of Congestion: Congestion can be detected on receiving the duplicate acknowledgments (ACKs) or Retransmission Time Out (RTO).
- Notification for Congestion: It is the initial signal before the loss of data over bottleneck links.
- Congestion Treatment (Control): Congestion is controlled with the help of four algorithms (Slow start, Congestion avoidance, Fast retransmit and Fast Recovery) for defining the flow rate of sender.

TCP-ICCW (Initial Constant Congestion Window) mechanism begins with Slow start and Congestion avoidance phases to throttle the novel flow rate in the typical TCP. The procedure invokes with an SYN + Clock Counter for creating a connection between client and server. This clock counter is used to calculate the RTT (Round Trip Time) until the ACK of that packet. Figure 2 is showing the calculation of RTT.

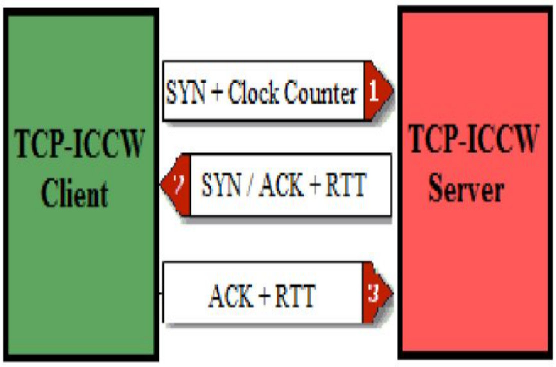

Figure 2: Three Way connection and calculation of RTT.

The TCP New Reno procedure uses the function baserrt intended for the calculation of first RTT. Thus, the basertt is equal to the first ACK. The Slow start phase declares the ssthresh value which maintains the threshold value of data flow. The ssthresh is calculated with the help of second and third ACKs. Congestion detection slows down due to time wasted until the third ACK is received. However, in TCP-ICCW ssthresh is calculated with the help first and second received acknowledgments. The baserrt function replaces the third ACK to detect the congestion earlier as shown in the Algorithm 1.

Algorithm 1: Calculation of ssthresh

```
elseif ( ACK == 2 )
ACK2 = Scheduler::instance().ClockCounter()
SS_F = ssthresh
Novel_SS_F = int(basertt * (data_size / (basertt – ACK2)))
/ data_size
If newreno_changes > 0 && Novel_SS_F < SS_F
Ssthresh = Novel_SS_F
End if
```

After detecting the congestion, TCP-ICCW proposes a new flow rate to treat the cause of congestion even at fully congested (Bottleneck) link.

- **Slow Start**

Initially, the Congestion Window (CWND) is set to the Send Window (SWND) which has null value. Then, the preliminary threshold is defined as SSμ after the measurement of μ. This is evaluated as follows:

$$CWND = SWND = 0; \quad (1)$$
$$ICCW \leftarrow CWND + \mu; \quad (2)$$
$$\mu = f(CWND) = \{2 (CWND + 3); \quad (3)$$
$$CWND \leftarrow ICCW \quad (4)$$
$$SST \leftarrow SS_{\mu} \quad (5)$$
$$If\ SST \geq SS_{\mu} \quad (6)$$
$$CWND \leftarrow CWND + 1 \quad (7)$$

When the transmission of packets touches the SSμ, the flow immediately turns to increase exponentially till the final threshold SSF.

$$SST \leftarrow SS_F \quad (8)$$

- **Congestion Avoidance**

At some point the capability of CWND reaches its maximum value SSF, the second phase Congestion avoidance invokes the linear increase of data flow. This linear approach increases the size of congestion window with one packet per RTT.

$$CWND = CWND + 1 \setminus CWND \quad (9)$$

- **Packet Loss**

After detecting the congestion, the sender needs to figure out its cause. If congestion is recognized because of the duplicate ACK then it is not a sign of packet loss and flow invokes the Fast Retransmit. The duplication of ACKs can be generated due to delay and out of order packets. However, if three or more duplicate ACKs remain continue to receive for the same packet. Then it is a strong indication that at least one segment has been lost. At this stage, the value of CWND should be cut in half. The following values of CWND and threshold need to assign after experiencing the delay and packet loss.

International Journal of Advanced Computational Engineering and Networking, ISSN(p): 2320-2106, ISSN(e): 2321-2063
Volume-6, Issue-2, Feb.-2018, http://iraj.in

Duplicate ACK (Delay):

CWND = CWND + a (10)

Packet Loss:

CWND = CWND * b (11)

The assigned values are (a = 1, b = 1 / 2).

SST ← CWND (12)

- **Retransmission Time Out**

Instantly, Fast Recovery algorithm starts after the Fast Retransmit. In this way, the flow of congestion window starts again one packet per RTT until the RTO. As retransmission time expires, a significant packet loss occurs. Now, the congestion window is reset to its initial values which automatically put the sender into the Slow start phase. During the Slow start, the initial and final threshold values of congestion window (ICCW) are SSμ, SSF correspondingly. Subsequently, congestion avoidance restarts to continue the data flow smoothly.

SST ← max (CWND / 2, ICCW) (13)

CWND ← ICCW (14)

The following Figure 3 is depicting the effect of threshold values (SS_{μ}, SS_F) in an ICCW:

Figure 3: Initial Constant Congestion Window

PERFORMANCE PARAMETERS AND SIMULATION SCENARIO

A. Parameters

- Throughput

Throughput is basically the total capability of data transfer in a defined time. For QoS improvement, throughput has a vital role to analyze. Throughput should be high and is calculated by the following formula:

Throughput = Size of Sink's Window / Defined Time

Size of Sink's Window = Buffer Window of Receiver

Defined Time = RTT

- **Packet Loss**

Packet loss occurs when a packet sent from the source fails to arrive at sink due to variety of reasons such as congested links, the overburden of data at the sink, out of date information and unnecessary delay. Packet loss is measured in ratio and this ratio should be low during the data transmission.

Packet Lost = Packet Sent – Packet Received

Packet Lost Ratio = Packet Lost / Packet Received * 100

- **Latency or Delay**

Latency or delay is a time experienced by a packet to get from source to designated sink. Different type of data faces different time span based on their requested link to follow. Network delay should be low as much as possible. It is the summation of all types of delays experienced by a packet.

Propagation Delay (PD) = Distance from source to sink / Available Link Speed

Transmission Delay (TD) = Size of Packet / Rate of Transmission

Queuing Delay (QD) = Time to wait in the queue

End-to-end Delay = Σ (PD + TD + QD)

B. Simulation Configuration

In the simulation setup, the Network Simulator 2 (NS-2) is used for the implementation and evaluation of TCP-ICCW procedure. We assume a topology of eight nodes, each of them having 2 MB link capacity. The initial four nodes are sending their data towards five and six nodes. Afterwards, the seventh node is collecting data from the fifth and sixth node via different links. In the end, the seventh node is forwarding the whole data to the eighth (sink) node with a single shared link. Every node is restricted to follow this shared route which has only 2 MB link capacity for transmission. Only one request for the link can entertain at once while other nodes must wait in the queue. When the waiting time in the queue increases, congestion takes place in the network. Table 1 presents the simulation notations used in the network model:

Table 1
Simulation Notations and Description

Notations	Detail Description of Notations
SST	Slow Start Threshold
SS_{μ}	Threshold switch from constant to exponential flow
SS_F	Final threshold terminate the Slow Start
a and b	Values to handle the delay and packet loss
2 mb	Capacity (speed) of shared (congested)link
Data Size	1500 KB
Data Types	Tcp, ftp, udp and cbr
Network Type	Wired, Wireless
5 ms	Simulation time in both networks
Full Duplex	Communication Link Type
Drop Tail	Interface Queue Type

V. RESULTS AND DISCUSSION

The extended TCP-ICCW mechanism is compared with other existing variants of the TCP like TCP New Reno, SACK, VEGAS and FACK. These TCP variants have pros and cons as stated by the parameters; throughput, loss and delay. The New Reno has an ability to detect the multiple packet loss, therefore, it allows multiple retransmissions at the Fast Recovery phase. The SACK uses "Selective Acknowledgments" to retransmit the missing segment only instead of multiple retransmissions. The VEGAS defines that the proactive calculation to detect the congestion is better than the reactive one. VEGAS detect the congestion prior to packet loss. FACK is built to employ with the SACK. It uses "Forward Acknowledgment" which provides additional information about the status of the network

A. Scenario 1: Results of Wired Network

The proposed mechanism is implemented in the wired network environment shown as follows in Figure 4:

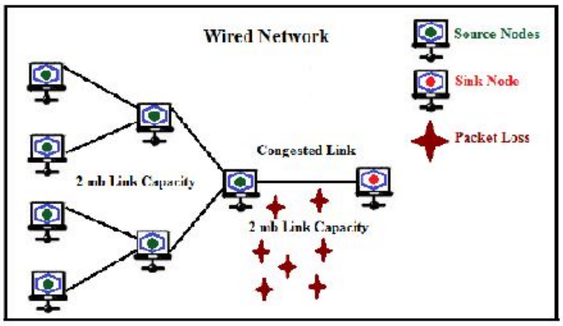

Figure 4: TCP-ICCW Topology in Wired Network

Figure 5 shows highest average throughput among all other TCP variants:

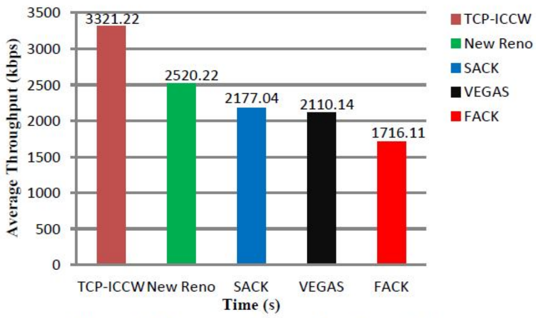

Figure 5: Average Throughput over Time

The packet delivery ratio depends on the successful delivery of the packets to the destination. So, it is an essential metric to represent the effect of congestion on the network performance. Table 2 illustrates the results of packets sent and lost along with their delivery ratio:

Table 2
Packet Delivery Ratio

	Packet Sent	Packet Lost	Packet Delivery Ratio
TCP-ICCW	3151	24	99.2 %
New Reno	2853	27	99.0%
SACK	2231	33	98.5%
VEGAS	2109	36	98.2%
FACK	1907	42	97.7%

Figure 6 demonstrates the average end-to-end delay. The TCP-ICCW gives the minimum delay as compared to others for fast transmission over the wired networks.

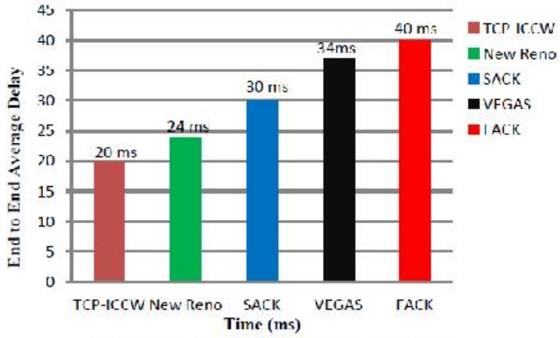

Figure 6: End to End Average Delay

B. Scenario 2: Results of Wireless Network

Figure 7 displays the pictorial view of the extended protocol in the wireless environment:

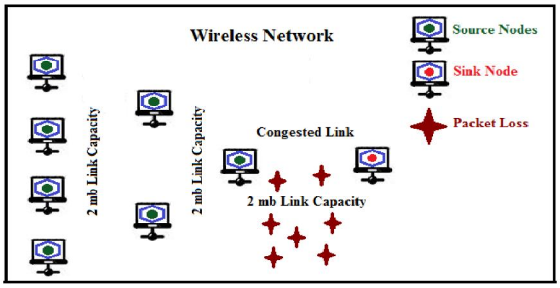

Figure 7: TCP-ICCW Topology in Wireless Network

The Figure 8 and Figure 9 analyze the average throughput and average end-to-end delay with the initial increase in sending rate of the CWND. Due to the effective utilizing of link capacity, throughput starts to increase under the assigned thresholds (Sµ, SF). After employing the proposed scenario, QoS parameters (throughput and delay) outperforms in the congestion control mechanism.

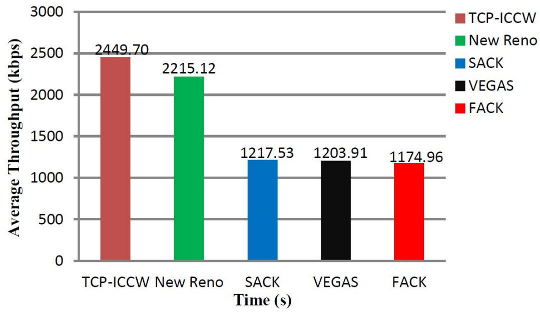

Figure 8: Average Throughput over Time

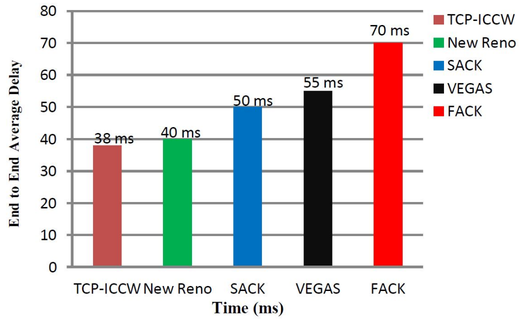

Figure 9: End to End Average Delay

The speed of wireless network remains low as compared to the wired one. Because wireless networks works on the basis of signals which can be interrupted by different factors (weather and more users)[11]. So, the probability of packet loss is higher in the wireless environment. The Table 3 lists the enhanced results of the proposed technique:

Table 3
Packet Delivery Ratio

	Packet Sent	Packet Lost	Packet Delivery Ratio
TCP-ICCW	1461	30	97.9%
New Reno	1053	33	96.8%
SACK	727	30	95.8%
VEGAS	719	28	96.1%
FACK	702	31	95.5%

CONCLUSION

Due to heavy traffic over a single shared link, nodes contend for that shared link. Also, limited link capacity creates a situation of congestion. A novel methodology of data routing is proposed based on the standard TCP protocol to handle this congestion. A new sending rate of data is devised considering the QoS metrics including throughput, loss and delay. We proposed two threshold values (Sμ, SF) for managing the flow rate on the behalf of efficient utilization of bandwidth during congestion. In addition, the delay and packet loss work on the values of linear increment and fifty percent decrement of the CWND (a, b) respectively. Finally, as compared to other TCP variants, extensive simulation results of TCP-ICCW shows a decrease in the delay, loss and increase in throughput during the wired/wireless computer networks.

ACKNOWLEDGMENT

This work was supported by the Research Project of Science and Technology for Hebei Province Higher Education Institutions under Grant BJ2014013 and in part by the Natural Science Foundation of Hebei Province under Grant 2016202341.

REFERENCES

[1] L. Zhang and S. Valaee, "Congestion Control for Vehicular Networks With Safety-Awareness", IEEE/ACM Transactions on Networking, vol. 24, no. 6, pp. 3290-3299, 2016.
[2] F. Hsu, Y. Hwang, C. Tsai, W. Cai, C. Lee and K. Chang, "TRAP: A Three-Way Handshake Server for TCP Connection Establishment", Applied Sciences, vol. 6, no. 11, p. 358, 2016.
[3] J. Park, "Congestion-Aware Geographic Routing Protocol for Wireless Ad Hoc and Sensor Networks", Wireless Personal Communications, vol. 78, no. 4, pp. 1905-1916, 2014.
[4] M. Kafi, J. Ben-Othman, A. Ouadjaout, M. Bagaa and N. Badache, "REFIACC: Reliable, efficient, fair and interference-aware congestion control protocol for wireless sensor networks", Computer Communications, vol. 101, pp. 1-11, 2017.
[5] A. Rezaee and F. Pasandideh, "A Fuzzy Congestion Control Protocol Based on Active Queue Management in Wireless Sensor Networks with Medical Applications", Wireless Personal Communications, 2017.
[6] M. Kafi, D. Djenouri, J. Othman, A. Ouadjaout, M. Bagaa, N. Lasla and N. Badache, "Interference-aware Congestion Control Protocol for Wireless Sensor Networks", Procedia Computer Science, vol. 37, pp. 181-188, 2014.
[7] P. Huang, X. Lin, and C.Wang, “A low-complexity congestion control and scheduling algorithm for multihop wireless networks with order-optimal per-flow delay,” IEEE/ACM Trans. Netw., vol. 21, no. 2, pp. 495–508, 2013.
[8] N. Aslam, K. Xia, A. Ali and S. Ullah, "Adaptive TCP-ICCW Congestion Control Mechanism for QoS in Renewable Wireless Sensor Networks", IEEE Sensors Letters, vol. 1, no. 6, pp. 1-4, 2017.
[9] H. Sun, "Congestion Control Based on Reliable Transmission in Wireless Sensor Networks", Journal of Networks, vol. 9, no. 3, 2014.
[10] N. Aslam, K. Xia, M. Haider and M. Hadi, "Energy-Aware Adaptive Weighted Grid Clustering Algorithm for Renewable Wireless Sensor Networks", Future Internet, vol. 9, no. 4, p. 54, 2017.
[11] P. Li and Y. Fang, "On the Throughput Capacity of Heterogeneous Wireless Networks", IEEE Transactions on Mobile Computing, vol. 11, no. 12, pp. 2073-2086, 2012

★★★