The Reset Frequency Controlled Parameter Re-estimation for the Improvement of Congestion Control in DCCP_TCPlike

B. Chellaprabha[1], Dr. S. Chenthur Pandian [2] and Dr. C. Vivekanandan[3]

1. Head, Department of Computer Science and Engineering, SNS College of Engineering, India
2. Principal, Dr. Mahalingam College of Engineering & Technology, Tamilnadu, India,
3. Professor and Dean, Electrical Sciences, SNS College of Engineering, Tamilnadu, India.
e-mail: chellapraba@gmail.com

ABSTRACT: The performance of the congestion control algorithms of most of the reliable transport protocol of internet, particularly wireless sensor networks (WSN) and mobile ad-hoc networks, are not satisfactory under a high density sensor network applications, since those algorithms were designed mainly for wired networks and not for WSN. Authors in their previous work [1], [2] and [3], have evaluated the performance some of the well-defined transport protocols on a congested sensor network scenario and established that the DCCP transport protocol is more suitable for sensor network applications. An improved *reset frequency controlled parameter re-estimation* method for enhanced Congestion Control in DCCP_TCPlike protocol, RC_DCCP_TCPLike, is suggested in this paper. The proposed protocol was implemented on a typical network scenario, simulated using *ns2,* and its performance was evaluated with respect to standard metrics *viz.* throughput, dropped packets, energy consumption, routing load, MAC load and end-to-end delay and compared with those of the networks with normal DCCP_TCPLike protocol. From the results it is observed that the performance of RC_DCCP_TCPLike protocol based networks is better than normal DCCP-TCPLike based networks.

[B. Chellaprabha, S.Chenthur Pandian and C. Vivekanandan. **The Reset Frequency Controlled Parameter Re-estimation for the Improvement of CongestionControl in DCCP_TCPlike.** *Life Sci J* 2013:10(1):3234-3240]. (ISSN: 1097-8135). http://www.lifesciencesite.com. 408

Keywords: Congestion Control, Transport Protocols, Sensor Network, DCCP, DCCP_TCPLike, RC_DCCP_TCPLike.

1. INTRODUCTION

1.1 The Wireless Sensor Network (WSN)

A wireless sensor network is a wireless network consisting of spatially distributed autonomous devices with sensors to cooperatively monitor physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or pollutants, at different random locations [4]. The information sensed by those devices are sent in form of data packets and for a data packet to be delivered to the destination reliably, a transport layer protocol must be embedded between application layer and network layer. Under such a scenario if an application involves voluminous data transfer persistent congestion may occur and hence, requires a more reliable transmission [5]. In such high rate sensor network applications a fairly reliable solution is mandatory to avoid congestion and to maintain complete and efficient data transfer between many sources and one or more sinks [6].

The paper is organized in the following manner. The necessity of this work in the present scenario is discussed in the next section. An overview of DCCP transport protocols and a brief description on congestion control are given in Chapter II. The proposed reset controlled parameter estimation method is discussed in detail in Chapter III. The parameters of a typical WSN scenario and values assigned to them are given in Chapter IV. The simulated responses of the WSN using *ns2*, in terms of the metrics considered in this work along with an exhaustive analysis, are given in Chapter V. Chapter VI discusses the inferences arrived out of the analysis.

1.2 Need of this work

A typical wireless sensor network is highly unstable as it is error-prone due to various reasons such as interference of radio signal, radio channel contention, and survival rate of nodes [7]. The comparatively new protocol DCCP is having interesting properties which makes it possible to use it in an error-prone sensor network scenario. Even though DCCP works under sensor networks with minimum overheads as shown in our earlier evaluation [3],The satisfactory performance of the DCCP protocol based networks with minimum overheads has been established by the authors in their previous work [3]. However, in a detailed analysis that followed it was felt that there are certain aspects of DCCP need to be improved to make it more suitable for WSN. As it is evident that the *reset* function and *timeout* parameters have considerable influence on reporting interval and overall performance of the sensor network, it is proposed to exploit those parameters to implement DCCP in WSNs more efficiently.

2. The TRANSPORT PROTOCOLS AND CONGESTION CONTROL

2.1 Congestion Control

Congestion, transmitting packets beyond the admissible limit of a link, may not be constant over the different points of the WSN, due to its multi-hop nature and a different degree of congestion might be felt at different points over WSN [8]. It is obvious that the congestion is high around the base station or 'sink', due to the convergent nature of the traffic towards the base station. Hence, it is required not only to detect the congestion but also to implement an appropriate avoidance technique to minimize the losses and to increase the overall performance of WSN.

2.2 Data Congestion Control Protocol (DCCP)

With User Datagram Protocol (UDP) as the base, DCCP was developed for effective and efficient handling of congestion over WSN resulting in more reliable transmission of datagrams or packets [11]. The main objective of DCCP is to extend support for implementing different congestion control schemas out of which the most suitable one may be selected by the applications, so as to provide efficient congestion control. Hence, according to the type of data being transmitted a schema will be selected to assure a better flow of packets. A mechanism, known as Congestion Control Identification (CCID), is implemented in DCCP, enabling it to assign separate CCID for each direction of data flow. The nature of congestion is defined by CCID and the source as well as destination select appropriate mechanism to handle this congestion by feature negotiation [9], a method that selects the best suited algorithm for the present scenario.

2.2.1.DCCP_TCPLike Congestion Control (DCCP_TCP_Like)

The DCCP_TCP_Like Congestion Control mechanism implements an algorithm that controls the congestion through tracking a transmission window, and regulating the transmit rate similarly to that of TCP and called as CCID 2. CCID 2 takes the advantage of available bandwidth in a rapidly changing environment and is suitable for senders who can adapt to the abrupt changes in congestion window, typical to that of TCP's Additive Increase Multiplicative Decrease (AIMD) congestion control [10].

2.2.2 DCCP TCP Friendly Congestion Control (DCCP_TFRC)

Congestion control is implemented in DCCP_TFRC by tracking the packet loss rate and varying the transmission rate in a smoother manner using additive increase and subtractive decrease, and this technique of congestion control is known as CCID 3. DCCP_TFRC is a receiver-based congestion control mechanism that provides a TCP-friendly sending rate while minimizing the abrupt-rate-change characteristic of TCP or TCP-like congestion control.

The state diagram for the client and server DCCP Process

Figure 1 and 2 show the state diagram of a typical DCCP client server communication scenario. When the DCCPAgent wants to send a packet, it queries the congestion control mechanism for permission by calling *send_askPermToSend()*. After a packet has been sent successfully the *send_packetSent()* method is called. The methods *recv_packetRecv()* and *send_packetRecv()* informs the receiver and the sender respectively of an incoming packet. DCCP informs the congestion control mechanism about DCCP-Data, DCCP-Ack and DCCP-DataAck packets sent or received during the OPEN state. The *reset()* function is called when the connection has to be closed for some reason. In particular, if the congestion control state needs to be returned to the initial setting. If the congestion control mechanism deploys timers, the *cancelTimers()* method must be overridden as it cancels all timers when a connection is closing or being reset. All timeout events should be taken care of through the *timeout()* method.

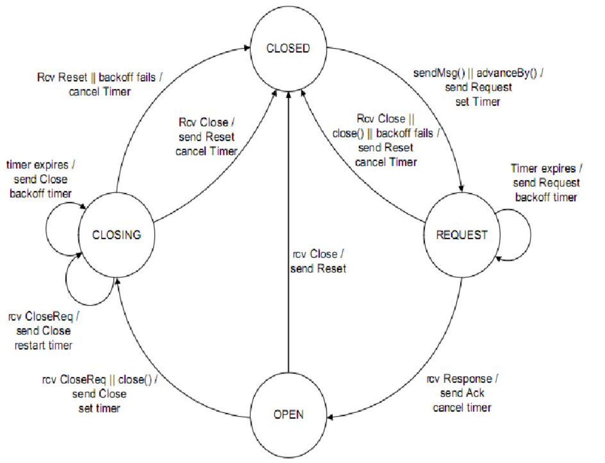
CLOSED
CLOSING
REQUEST
OPEN
Rcv Reset || backoff fails / cancel Timer
sendMsg() || advanceBy() / send Request set Timer
Rcv Close / send Reset cancel Timer
Rcv Close || close() || backoff fails / send Reset cancel Timer
timer expires / send Close backoff timer
Timer expires / send Request backoff timer
rcv Close / send Reset
rcv CloseReq / send Close restart timer
rcv CloseReq || close() / send Close set timer
rcv Response / send Ack cancel timer

Figure 1: State chart for the client

There are two differences between the state charts presented in this paper in Figure 1 and Figure 2, and the those presented in the *DCCP draft* [11]. The major difference is that the TIMEWAIT state present in *DCCP_draft* is removed in the state diagrams presented this work. In addition to this, in the chart presented in this paper, when a connection is closed, or reset, the server side reverts back to LISTEN state instead of CLOSED state. This simplifies the simulation by removing the need to call *listen()* after a connection is closed, to put the node in listening mode. All agents are started in the CLOSED state and when an application issues a call to *listen(),* the agent becomes a server by changing its state to LISTEN. During the OPEN state, the server and the client exchange DCCP-Data, DCCP-Ack and DCCP-DataAck packets as governed by the congestion control mechanism.

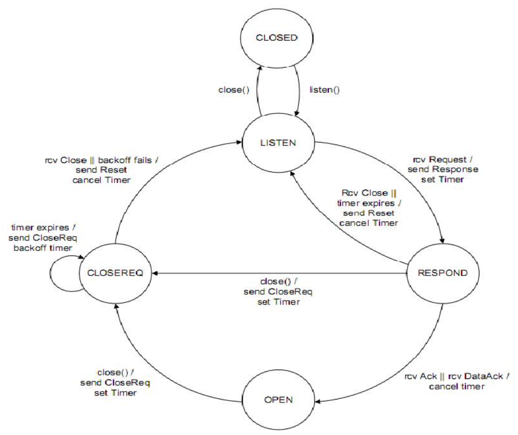
CLOSED
close()
listen()
LISTEN
rcv Close || backoff fails / send Reset cancel Timer
rcv Request / send Response set Timer
Rcv Close || timer expires / send Reset cancel Timer
timer expires / send CloseReq backoff timer
CLOSEREQ
RESPOND
close() / send CloseReq set Timer
close() / send CloseReq set Timer
rcv Ack || rcv DataAck / cancel timer
OPEN

Figure 2 : State chart for the server

3. The Proposed Reset Frequency Controlled Parameter Re-estimation Method

From the two figures given in the previous section, it is obvious that the reset function is frequently called in several states of the client server communication process during the typical DCCP communication scenario. A DCCP-Reset may be initiated by both the server and the client to abort a connection in case an error occurs. The DCCP agent will reset the connection if any feature negotiation fails during the handshake or if the feature option is of an erroneous size. Upon the reception of a reset request in any state, the server moves to LISTEN and the client moves to CLOSED. The reset function sets lot of parameters to their respective initial values. It may be noticed that under DCCP protocol the *reset()* function is invoked repeatedly in a few seconds of interval due to unexpected or unrecognizable events [11]. Similarly, in the simulated sensor network with different data reporting intervals also, it was observed that the *reset()* function was called very often to handle congestion. During each reset, all the parameters were set to their respective predefined initial values to reduce the overhead and recover from congestion. In addition to that, under normal working of DCCP, on realizing congestion the nodes stop transmitting data up to *timeout* period. The most important parameters of DCCP controlling the overall process are *maximum_retransmit_time_out* and *timeout_value.* If this *timeout* period is constant, then it has much influence on the sensor data reporting interval. For example, if the sensor data reporting interval is 5 seconds and the DCCP timeout period is 10, then instead of sending data at every 5 seconds, the sensor node can send data only at every 10 seconds of more. Hence, just keeping this timeout period as constant under a sensor network certainly affects the sensor reporting interval and overall performance. From the above it may be concluded that timeout period and retransmission period have definite impact on the overall performance of communication under the sensor network and also influence the reporting interval of a sensor node.

The Proposed Modifications in DCCP Reset Function:

The following pseudo code explains the simple modification in the reset function of DCCP_TCPLike. Three of the parameters involved are set according the "Reset Frequency", *i.e.*, with respect to the frequency of the *reset()* function calls. Hence, this enhancement of DCCP is referred to as "Reset Frequency Controlled or Reset Controlled Initial Parameter Re-estimation".

```
ResetInterval←now()
PrevResetTime← 0
AvgResetInterval← 0

OnReset() {
 //Normal DCCP_TCPLIKE
  If Method= DCCP_TCPLIKE{
          ssthresh_← INIT_SSTHRESH_
          initial_rtx_to_ ← INITIAL_RTX_TO
          max_rtx_to_ ← MAX_RTX_TO
          cwnd_ ← INIT_CWND }

//Proposed RC_DCCP_TCPLIKE
If Method=RC_DCCP_TCPLIKE {
PrevResetInterval ← ResetInterval
ResetInterval←now() - PrevResetTime;
AvgResetInterval←(AvgResetInterval+ResetInterval)/2
.0;
PrevResetTime←now();
   if ( ResetInterval>PrevResetInterval)  {
ssthresh_=INIT_SSTHRESH * AvgResetInterval
cwnd_= INIT_CWND * AvgResetInterval ;
max_rtx_to_=MAX_RTX_TO-(AvgResetInterval/
MAX_RTX_TO )
          initial_rtx_to_=1;
          }
   if ( ResetInterval<PrevResetInterval)  {
          ssthresh_← INIT_SSTHRESH_
          initial_rtx_to_ ← INITIAL_RTX_TO
          max_rtx_to_ ← MAX_RTX_TO
          cwnd_ ← INIT_CWND  }
DoOtherNormalResetActions()
   }  }
```

The three parameters *viz. ResetInterval, PrevResetTime and AvgResetInterval* alone are considered for modification in this work. The parameter *ResetInterval* is set to current-time and the remaining two are initialized to zero in both normal DCCP and proposed DCCP, once the transmission process is initiated as shown in the very beginning of the algorithm. As shown in the algorithm, in case of normal DCCP, upon *reset,* the parameters slow start threshold value (*ssthresh),* initial retransmission timeout *(initial_rtx_to),* maximum retransmission timeout *(max_rtx_to)* and the congestion window size (*cwnd)* are set to their corresponding initial values.

In the proposed method, upon the reception of reset*()* it estimates *ResetInterval*, the time elapsed (t_r) between the current reset and the immediate previous reset, tracked by *PrevResetTime*. It is obvious that smaller the t_r is higher the congestion and higher the t_r is lower the congestion. Then it updates the average reset time by including the recent one and sets the parameter *PrevResetTime* with current time to estimate the time elapsed between the current call to *reset()* function and the next call that may arise in future.

Table 1 : Parameters of the Sensor node and Network

Parameter		Value
Transmission Range		
Sink Node	:	150 m
Sensor Node 1 to 7	:	150 m
Other Sensor Nodes	:	60 m
Channel	:	Wireless Channel
Propagation	:	Two Ray Ground
Physical Medium	:	Wireless Physical
Antenna	:	Omni Antenna
Routing Protocol	:	AODV
Mac Type	:	802.11
Queue	:	DropTail/PriQueue
Queue Size	:	50
Sensor Reporting	:	1, 2, 5, 7 nd 10 sec
Interval	:	CBR
Traffic Application	:	256 Kilo bytes
Sensor Data Size	:	56
Number of Nodes	:	800m x 400m
Topographical Area	:	*DCCP_TCPlike and*
Transport Protocols		*RC_DCCP_TCPlike*
Simulation Time		100 sec
Node Receiving		3.652e-10
Threshold		2.4e09 Hz
Node Signal Frequency		

Based on the above estimations it is possible to evaluate the current congestion level by comparing the two consecutive congestion intervals tracked by *PrevResetInterval* and *ResetInterval*. If the value of current reset interval *ResetInterval*is greater than its immediate predecessor *PrevResetInterval,* it is obvious that there is less congestion and hence, still better transmission may be achieved by increasing the threshold value *ssthresh* and the congestion window size *cwnd*. The improvement in parameters *ssthresh* and *cwnd* are based on the *AverageResetInterval* as given in the algorithm. Further the parameter *max_rtx_to,* which specifies the time that a sender has to wait for the acknowledgement from the receiver, may also be reduced in view of lesser congestion accordingly as specified in the algorithm.

On the other hand if the value of *ResetInterval*is smaller than *PrevResetInterval* then it is the indication of increased congestion, which forces all the parameters to their respective initial values as in the case of normal DCCP.

4. SIMULATION OF SENSOR NETWORK

The proposed sensor network simulations have been successfully developed using ns2 simulator. Except the parameters maximum retransmit timeout (DCCP_MAX_RTX_TO) and timeout value in RESPOND state (DCCP_RESP_TO) the default values of all other parameters were intact. DCCP_MAX_RTX_TO and DCCP_RESP_TO were set to 5 seconds, as the default value, 75 seconds, was so high that the nodes will not transmit data for initial 75 seconds on realizing congestion, leading to make a break in periodic reporting of sensor data. The values are so chosen that DCCP is set to send data at intervals less than 10 seconds. To implement the proposed Reset Controlled Parameter Re-estimation for the Improvement of Congestion Control in DCCP, the necessary modification were made in ns-default.tcl, dccp_tcplike.cc and dccp_tcplike.h.

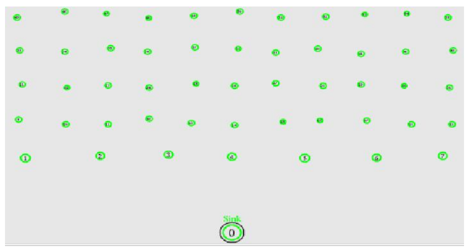

Figure 3 : Wireless Sensor Network

Figure 3 depicts the simulated sensor network field. The 0th node is the sink node. Node 1 to 7 are sensor nodes capable of communicating to Sink node with relatively higher Tx power All other nodes are normal sensor nodes and can transmit only for a short distance. The Transmission power of the sensor nodes and sinks are set based on the required transmission range, signal frequency and other related parameters.

5. RESULT AND ANALYSIS

With different Data Reporting Intervals the sensor networks with transport protocols DCCP_TCPlike and the proposed RC_ DCCP_TCPlike were simulated and all the events were logged in trace files. Based on the desired metrics the performance of the network is analyzed using the logged data as discussed below.

5.1 Performance in Terms of Throughput

Figure 4 shows the performance of protocols in terms of throughput, with respect to different sensor data reporting intervals. The proposed

RC_DCCP_TCPLike protocol is providing better throughput than normal DCCP_TCPLike for a wide range of data reporting interval by reducing the number of packets dropped during transmission. It is due to the effective usage of *cwnd* by increasing its size during less congestion in the network. The better average throughput in case of RC_DCCP_TCPlike is evident as shown in Figure 5.

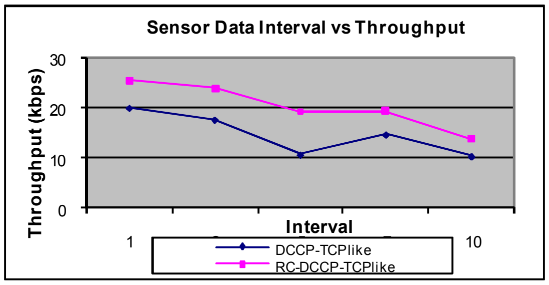

Figure 4 : Throughput with respect to Different Data Interval

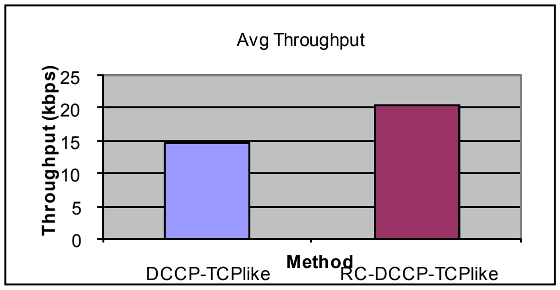

Figure 5 : Average Throughput

5.2 Performance in Terms of Energy Consumption

*I*n the energy front also the performance of RC_DCCP_TCPlike protocol based network is better than that of normal DCCP_TCPlke based counterpart as shown in Figure 6. Due to improved throughput the need for retransmission of packets decreases and hence, the overall energy required to send the given amount of data-packet also falls as shown in Figure 6 and Figure 7.

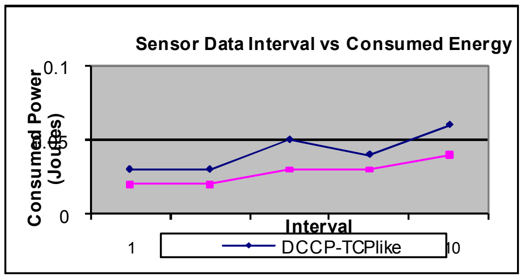

Figure 6 : Consumed Energy with respect to Different Data Interval

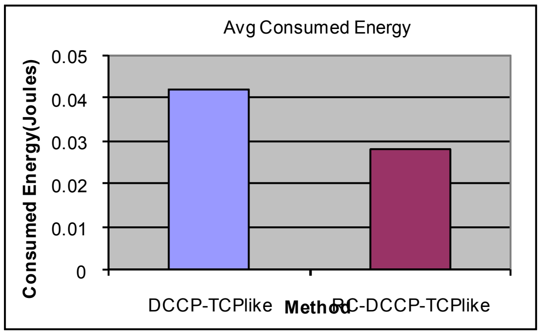

Figure 7 : Average Consumed Energy

5.3 Performance in Terms of Normalized Routing Load

The performance of the proposed RC_DCCP_TCPlike protocol along with the performance of DCCP_TCPlike protocol, in terms of normalized routing load with respect to different data reporting interval is given in Figure 8. It may be observed from the plot that the routing load in case of RC_DCCP_TCPlike is always lesser than that of DCCP_TCPlike, as the number of routing packets required to deliver the data packets successfully comes down due the fall in packet losses as already discussed. Figure 9 relates the average routing load and the data reporting interval and the reason better performance of RC_DCCP_TCPLike is evident from the foregoing discussions.

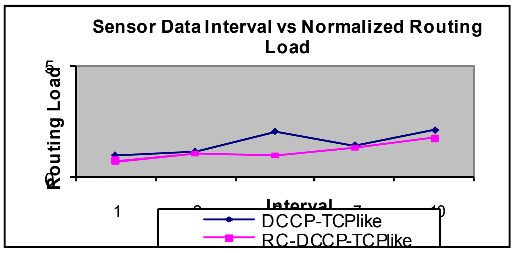

Figure 8 : Normalized Routing Load with respect to Different Data Interval

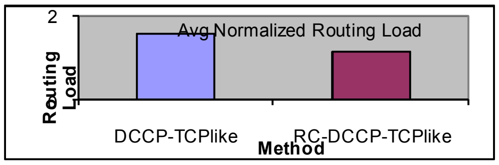

Figure 9: Average Normalized Routing Load

5.4 Performance in Terms of Normalized MAC Load

The performance of the proposed protocol over the range of data reporting intervals, evaluated in terms MAC Load, another metric considered in this work, is given in Figure 10, along with the performance of its DCCP_TCPlike counterpart. It may be observed that there is no much difference in the performances of the protocols during low data intervals. In the high data reporting intervals too the difference is not so significant. However, as far as the average MAC load is concerned, it may be observed from Figure 11, the performance of the RC_DCCP_TCPlike is marginally better than that of DCCP_TCPlike protocol.

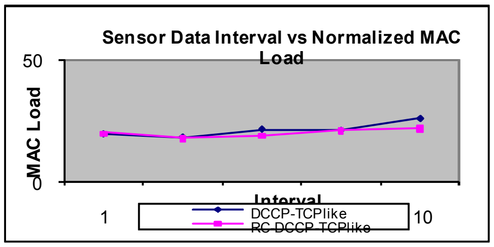

Figure 10 : Normalized MAC Load with respect to Different Data Interval

Figure 11: Average Normalized MAC Load (Omit)

5.5 Performance in Terms of Dropped Packets

The number of packets dropped over the desired Data reporting interval for the two protocols is given in Figure 12. It may be observed that the reset-control based protocol is more consistent with lesser dropped packets over period of time though the count of dropped packets increases with time in both cases. The better performance of the RC_DCCP_TCPlike is evident when considering the average packet loss as shown in Figure 13.

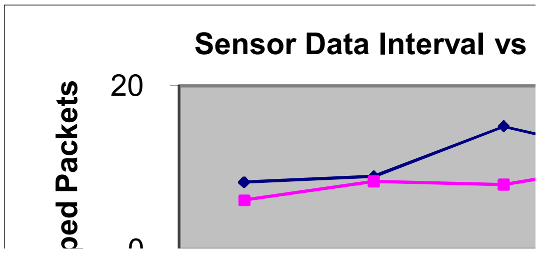

Figure 12 : Packets Dropped Vs Different Data Interval

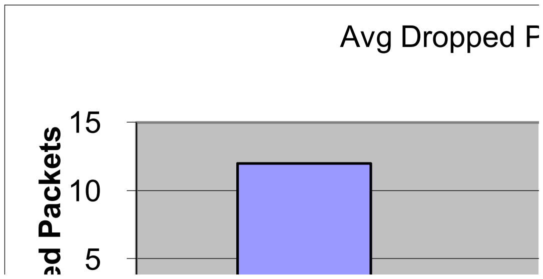

Figure 13 : Average Packets Dropped

5.6 Performance in Terms of End to End Delay (E2E)

Over the time period considered, identifying the better protocol among the two protocols concerned, with respect to E2E delay, is difficult from the corresponding performance chart given Figure 14, as the performance curves cross each other and the difference between them is negligible. However, on considering the average End-to-End delay it is found that RC_DCCP_TCPlike based networks provide better performance as shown in Figure 15.

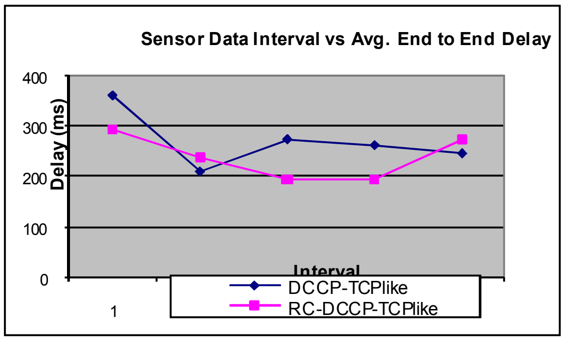

Figure 14 : The End to End Delay with respect to Different Data Interval

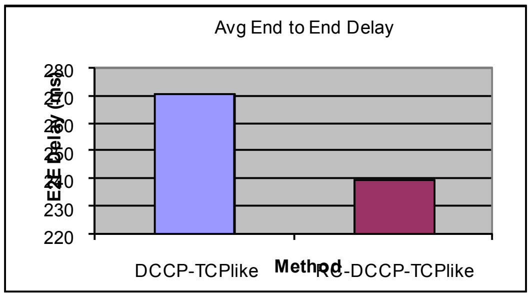

Figure 15 : Average End to End Delay

6. CONCLUSION

To improve the performance of DCCP_TCPlike protocol based networks with better congestion control techniques, modification on the reset-function is considered and the proposed model, RC_DCCP_TCPlike, is simulated using *ns2* simulator with parameters of *ns2* set to replicate a practical scenario. Over a wide range of data reporting interval, the scenario was simulated repeatedly and the performance of the networks based on the normal DCCP_TCPlike and proposed RC_DCCP_TCPlike, were analyzed based on standard metrics.

In this work three of the parameters of DCCP reset() function are dynamically updated to improve the transmission of data packets during low congestion and as a result an appreciable improvement in throughput is observed. Due to the improvement in throughput, packet-drop during transmission is brought down which in turn decreases the associated energy requirement, routing load and MAC load and improves the overall performance of WSN, barring E2E delay. Hence, it may be concluded that the networks based on RC_DCCP_TCPLike protocol during less congestion provide better congestion control by effectively updating the slow start threshold, congestion window size and maximum retransmission timeout, compared to its predecessor DCCP_TCP_Like. The same reset control mechanism may suitably be modified and applied to DCCP_TFRC networks to improve the overall performance. Further suitable modifications may also be suggested to improve the performance of DCCP based WSNs by exploiting its reset function by making more parameters to be dynamic instead of initializing them to standard predefined values upon reset.

7. ANNEXURE

Results of Default DCCP-TCPlike , Proposed RC-DCCP-TCPlike different sensor data reporting intervals.

Table 2: Performance of DCCP-TCPlike

Interval	Avg Delay	Avg End to End Delay	Normalized Routing Load	Normalized MAC Load	Dropped Packets	Throughput	Consumed Power
1	98.13	362.94	0.98	19.39	8.23	20.13	0.03
2	77.96	209.05	1.15	18.04	8.96	17.74	0.03
5	126.15	274.37	2.08	21.27	15.07	10.69	0.05
7	112.52	262.30	1.44	20.73	11.37	14.73	0.04
10	117.48	245.84	2.13	25.56	16.36	10.34	0.06
Avg	106.448	270.9	1.556	20.998	11.998	14.726	0.042

Table 3 : Performance of Proposed RC-DCCP-TCPlike

Interval	Avg Delay	Avg End to End Delay	Normalized Routing Load	Normalized MAC Load	Dropped Packets	Throughput	Consumed Power
1	76.92	295.46	0.68	19.63	6.01	25.51	0.02
2	81.7	238.38	1.01	17.3	8.28	23.92	0.02
5	77.06	194.11	0.92	18.55	7.89	19.29	0.03
7	78.49	195.92	1.28	20.51	10.61	19.38	0.03
10	109.43	276.66	1.73	21.72	13.56	13.86	0.04
Avg	84.72	240.106	1.124	19.542	9.27	20.392	0.028

References

1. B. Chellaprabha, Dr. S. ChenthurPandian, Dr. C. Vivekanandan, "Performance of TCP, UDP and SCTP on Sensor Network with Different Data Reporting Intervals", IOSR Journal of Engineering , Apr. 2012, Vol. 2(4), pp. 621-628.
2. B. Chellaprabha, Dr. S. ChenthurPandian, Dr. C. Vivekanandan, "Performance of Datagram Congestion Control Protocol DCCP-TCPLike And DCCP-TFRC on Sensor Network", IRACST – International Journal of Computer Networks and Wireless Communications (IJCNWC), ISSN: 2250-3501, Vol.2, No.2, April 2012.
3. B. Chellaprabha, Dr. S. ChenthurPandian , Dr. C. Vivekanandan, "An Extensive Evaluation of Transport Protocols Under Highly Congested Sensor Network", European Journal of Scientific Research ISSN 1450-216X, Vol.78, No.1, 2012, pp.93-106.
4. Jang-Ping Sheu, Li-Jen Chang and Wei-Kai Hu, “Hybrid Congestion Control Protocol in Wireless Sensor Networks”, Journal of Information Science And Engineering 25, pp. 1103-1119, 2009.
5. Y. G. Iyer, S. Gandham, S. Venkatesan, “STCP: A Generic Transport Layer Protocol for Wireless Sensor Networks”, 14th International Conference on In Computer Communications and Networks, pp. 449-454, 2005.
6. Alam and Hong, “Congestion-Aware and Rate-Controlled Reliable Transport in WSNS”, The Institute of Electronics, Information and Communication Engineers, 2009.
7. Yao-Nan Lien, "Hop-by-Hop TCP for Sensor Networks", International Journal of Computer Networks & Communications – IJCNC, Vol.1, No.1, April 2009.
8. Md. AbdurRahman, Abdulmotaleb El Saddik and Wail Gueaieb, “Wireless Sensor Network Transport Layer: State of the Art”, Sensors, Springer-Verlag Berlin Heidelberg, pp. 221-245, 2008.
9. D StanimirStatev, SeferinMirtchev, "Experimental Study of DCCP Transport Protocol in Varied Network States", International Scientific Conference Computer Science, 2008.
10. Kohler E., S.FloydKohler E., S. Floyd, and M. Handley, "Designing DCCP: Congestion Control without Reliability," in Proc. ACM SIGCOMM, Pisa, Italy, 2006.
11. Floyd S., E. Kohler, and J. Padhye, "Profile for Datagram Congestion Control Protocol (DCCP)", RFC 4340 (Proposed Standard), March 2006.

3/3/2013