IMPROVED BINARY INCREASE CONGESTION CONTROL ALGORITHMS FOR DATA TRANSFER IN SATELLITE NETWORK

M. Nirmala¹, Ramachandra V. Pujeri² and M. L. Valarmathi³
¹Department of Computer Applications, Hindusthan College of Engineering & Technology, Coimbatore, India
²MIT College of Engineering, Kothrud, Pune, India
³Department of Computer Science & Engineering, Government College of Technology, Coimbatore, India.
E-Mail: nilarangan@gmail.com

ABSTRACT
In our previous work [18], it was proved that the performance of bic-tcp was good under satellite network, and a new adaptive window increment based bic-tcp algorithm (bic_AWI) was proposed to show an improved data transfer in satellite network. We, implemented the idea on ns2’s implementation of bic tcp. Proposed modifications has been made on bic-tcp, and improvement in performance has been realized. Even though, the congestion avoidance mechanism of standard bic will be very much influenced by this proposed adaptive window change strategy, in update method of the bic, some of the parameters like BICTCP_B are kept as constant during finding a optimum tcp windows size.

The constant value BICTCP_B has been cahnged dynamically with respect to the acceleration and deceleration of rtt, has shown a considerable improvement in performance. Based on that idea, we proposed a dynamic parameter estimation based binary increase congestion control algorithm (bic_DPE) [19] for data transfer in satellite network. In this method, with respect to the acceleration of deceleration of rtt the important parameter BICTCP_B will be dynamically adjusted to provide best performance. Further, we added the mechanism to minimize packet loss during terminal handover. Proposed modifications has been made on bic-tcp, and the performance of the improved protocol “bic_AWI” and “bic_DPE” has been compared with normal bic-tcp under Iridium satellite constellation. The performance of the algorithms has been measured using suitable metrics and the proposed bic_DPE performed better than normal bic-tcp as well as our previous “bic_AWI” method.

Keywords: BIC_TCP algorithm, iridium satellite constellations, BIC_AWI protocol, BIC_DPE protocol, acceleration, deceleration, RTT.

1. INTRODUCTION
The usage of Satellite Communications for internet traffic can be taken as an attractive proposal[1]. In the near future, the satellite component is believed to be integrated with terrestrial component [2], [7] and the interaction of satellite and terrestrial connections can give rise to performance problems which are actually partially unexplored and are yet to be solved [3].

2. PROBLEM SPECIFICATION
In our previous study [18], we improved perforce bic-tcp and proposed a adaptive window increment based bic-tcp algorithm (bic_AWI) for improved data transfer in satellite network. The window size with respect to bic_AWI will be incremented with respect to the acceleration and deceleration of RTT was kept as a constant factor with the parameters like BICTCP_B. If this parameter is set dynmically then considerable improvement in performance is realized.

In the paper (IJAER) a dynamic parameter estimation based binary increase congestion control algorithm (BIC_DPE) for data transfer in satellite network has been proposed to provide the best performance. The minimization of packet loss during terminal handover is also added.

We, implemented the idea on ns2’s implementation of bic tcp. The 3 algorithms BIC_TCP, BIC_AWI and BIC_DPE has been compared under Iridium satellite constellations and the results has been graphed.

The satellite constellations [4] [7]
Satellite constellation is a collection of working together artificial satellites. In this paper the Iridium satellite constellation has been explored.

3. TCP CONGESTION CONTROL ALGORITHMS UNDER CONSIDERATION
In satellite networks, the available bandwidth is not used by the sender due to long propagation delay [9]; TCP was basically designed to be used with low link error rates networks which means all segment losses were due to network congestion. When there is a congestion the sender decreases the transmission rate at each time a segment loss is detected. It is the satellite networks, link error often occurs instead of network congestion and causes throughput degradation.

Binary increase congestion control for TCP (bic)[5]
As per paper [5], a new protocol has been designed by Lisong, Hasfoush and Rhee and it has
satisfied various criteria such as scalability, Fairness in RTT and TCP friendliness. The present paper has also satisfied the above mentioned criteria. The size of the congestion window is the most important parameter for TCP flow. Window parameter determines the performance of TCP flow. A protocol already presented on paper [5] called as Binary increase congestion control (BIC) has been the major algorithm used in this paper.

The two major aspects of BIC are

1. Binary search increase
2. Additive increase

Binary search increase allows bandwidth to drastically improve depending on the differences of current window and target window. When the differences are large bandwidth is more aggressive compared to small differences. Binary search increase when combined with additive increase, initially increases the window size linearly and then logarithmically.

The Midpoint algorithm calculation referred from [5] is used as a reference point in the complete below mentioned process.

The original form of BIC TCP algorithm [5]

The TCP module in NS2 has paved way to various congestion control algorithms and has helped various researchers to construct TCP in a network simulator such as NS2. The deviation in the implementation of TCP under NS2 from Linux has created greater problems such as simulation speed and coding [17].

The pseudo code of BIC TCP with respect to algorithm in fast recovery mode and not in recovery mode has been referenced from [5]. Refer to the algorithm for its working operations.

The preset parameters and the variables used for BIC TCP Algorithm are referred from [5] are used in this paper.

The Linux implementation of TCP BIC [6]

In the NS2 Linux implementation of TCP BIC various variables and parameters used are listed below. BICTCP_B is a constant under binary increase, The binary increase is reduced by a constant factor and its name is smooth part. The variable linux_min_win is set when there occurs a loss in packet and it is set.

NS2’s TCP Linux implementation of bictcp

The differences specified in the paper between Linux and NS2 [17] has been eliminated by creating a new design of NS2 TCP Linux. The NS2 TCP linux implementation uses TCP in NS2 with Linux Congestion control interface [17].

The boundary conditions and the algorithms used in the system implementation are referred from [6]

Based on the four conditions referred from [6], the value of the parameter BICTCP_B can be well understood. In the TCP Linux implementation standard version it is considered as 4, a constant value. The normal working of BIC algorithm has been explained using the flowchart.

![Figure-1](image)

Figure-1. The flow diagram explaining normal working of BIC TCP.

The following pseudo code is the congestionAvoid part of NS2’s implementation of bictcp. This implementation is equivalent of the above explained Linux tcp bic implementation. The function “On_bictcp_cong_avoid” will be called during each successful acknowledgment and for each call of the function “On_bictcp_cong_avoid”, the function “bictcp_update” will calculate ca_cnt according to the four boundary conditions mentioned in [6].

```
bictcp_update(cwnd, bictcp_b)
{if(target_win - cwnd > bictcp_b)
   ca_cnt= (cwnd * bictcp_b)/ target_win;
   if(| target_win - cwnd | < bictcp_b)
      ca_cnt = (cwnd * Smooth_Part)/ bictcp_b;
   if(target_win - cwnd < bictcp_b)
      ca_cnt = (cwnd * (bictcp_b -1))/ (cwnd target_win); return ca_cnt;}
```

Figure-2. The bictcp_update function.

The following Figure-3 is the standard bic_tcp congestion avoidance function.
Proposed bic with adaptive window increase (BIC_AWI)

The following function is modified for the implementation of BIC_AWI shown in Figure-4.

```c
On_BIC_AWI_tcp_cong_avoid() {
    increment = 0;
    prev_rtt = 0;
    diff_rtt = 0;
    now_rtt = rtt;
    diff_seq_rtt = now_rtt - prev_rtt;
    //find running average of rtt
    prev_rtt = (now_rtt + prev_rtt)/2;
    if(!tcp_is_cwnd_limited()) return;
    if(cwnd <= ssthresh) {
        cwnd += tcp_slow_start();
    } else {
        ca_cnt = bictcp_update(cwnd, BICTCP_B);
        if(cwnd_cnt >= ca_cnt) {
            if(cwnd < cwnd_clamp) {
                cwnd_cnt = 0;
                cwnd = cwnd + ada_inc;
            } else {
                cwnd_cnt += 1;
            }
        }
    }
}
```

Figure-4. The On_bictcp_cong_avoid function.

Proposed dynamic parameter estimation based BIC (BIC_DPE) algorithm

The following function referred in Figure is modified version of BIC_AWI.

```c
On_BIC_DPE_tcp_cong_avoid() {
    increment = 0;
    prev_rtt = 0;
    diff_rtt = 0;
    now_rtt = rtt;
    diff_seq_rtt = now_rtt - prev_rtt;
    //find running average of rtt
    prev_rtt = (now_rtt + prev_rtt)/2;
    if(!tcp_is_cwnd_limited()) return;
    if(cwnd <= ssthresh) {
        cwnd += tcp_slow_start();
    } else {
        //set the decrement factor of BICTCP_B
        //with respect to the acceleration and
deceleration of rtt
        if(diff_rtt < 0)
            decA = 1; else decB = 0;
        bictcp_B = BICTCP_B - decA;
        ca_cnt = bictcp_update(cwnd, bictcp_B);
        if(cwnd_cnt >= ca_cnt) {
            if(cwnd < cwnd_clamp) {
                cwnd_cnt = 0;
                cwnd_cnt += cwnd_cnt + 1;
            }
        } else {
            cwnd_cnt = 0;
        }
    }
}
```

Figure-5. The On_BIC_DPE congestion avoidance function.

In the above modified function, during handoff the cwnd will not get increased. The flag UnderTermLinkHandoffFlag will be set true from another layer of the protocol stack during handoff.

The onTermHandoffTimer referred is a timer function which will be called periodically and check whether the connection on the terminal is going to handoff to another satellite or not. During the period of handoff, the UnderTermLinkHandoffFlag will be set as true.

```c
void onTermHandoffTimer {
    UnderTermLinkHandoffFlag = IsUnderHandoff();
    if(UnderTermLinkHandoffFlag) {
        printf("\n%0.2f : Term Link handoff\n");
        ScheduleTermHandoffTimerAt(HandoffCheckInterval)
    }
}
```

Figure-6. The timer function which periodically checks handoff event.

This terminal handoff flag will control the behavior of the tcp agent. So, during handoff, the congestion window size will not be increased any further (but if any loss occurs, the congestion window will get reduced from the loss event handler). In future works, we may try to set the cwnd_cnt and the cwnd as zero, exactly
at the event of hand off, to avoid the risk of dropping packets during handoff.

4. SIMULATION AND VISUALIZATION OF IRIDIUM SATELLITE CONSTELLATION

The satellite network constellation used in this paper is Iridium. The study on the basic explanation for Iridium satellite constellation has been taken from [16].

Parameters of Iridium satellite constellation [10]

The parameters of Iridium satellite network has been referenced from [10], [7].

Software used for satellite communications simulation [2939, 12]

Network simulator (NS-2)

NS-2 is a discrete event network simulator. NS2 can be used for exact satellite network simulation with a detailed modeling of radio frequency characteristics.

Satellite handoff modelling in NS-2 [12]

NAM is the Network visualization tool used in NS2 and it nevertheless supports satellite network visualization. The other open source softwares used in this paper which supports visualization of satellite constellations are

- Sat-plot-scripts (perl scripts)
- SaVi (Satellite Constellations Visualization)
- Geomview

Sat-plot-scripts [13]

Perl scripts are used to visualize ns satellite constellation configurations, both as a complete snapshot at a given instant and the path a packet travels.

SaVi [14]

SaVi, the Satellite Visualization tool [14], is a used for visualizing and animating the movement of satellites and their coverage.

Features of SaVi includes

- 3D visualization of satellites in orbit around the Earth
- Satellites footprints on earth’s surface can be displayed
- Fraction of the Earth's surface covered by the constellation can be also computed.

Geomview [15]

Geomview is an open source Object Oriented Graphics Library used for geometry viewing.

The simulated Iridium Constellation and Terminals- Plotted using “Sat-plot-scripts” is shown in my previous paper [7].

The Iridium Constellation Map created using the “SaVi” tool has been shown in my previous paper [7].

5. SIMULATION AND ANALYSIS OF IRIDIUM SATELLITE CONSTELLATION

The TCP connection setup [8] [7]

A part of the moving satellite constellation of dumbbell configuration has been referenced from [7].

The important traffic parameters has been referenced from [7]

<table>
<thead>
<tr>
<th>Type</th>
<th>Bic, Bic_AWI</th>
</tr>
</thead>
<tbody>
<tr>
<td>PacketSize</td>
<td>1448</td>
</tr>
<tr>
<td>Initial Window Size</td>
<td>30000</td>
</tr>
<tr>
<td>TCP Sources</td>
<td>3</td>
</tr>
<tr>
<td>TCP Sink</td>
<td>3</td>
</tr>
<tr>
<td>Ground Terminal</td>
<td>2</td>
</tr>
<tr>
<td>Application</td>
<td>FTP</td>
</tr>
</tbody>
</table>

Table-1. Performance metrics [7].

<table>
<thead>
<tr>
<th>Contention Window (cwm) Time vs cwm</th>
<th>Better cwm results in better throughput and minimum EED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Round Trip Time (ETT)</td>
<td>Minimum RTT, better TCP performance</td>
</tr>
<tr>
<td>End to End Delay (EED)</td>
<td>Time taken for transfer and propagation</td>
</tr>
<tr>
<td>Delay / Packet Delay</td>
<td>Delay between 2 successive packets from sender to receiver</td>
</tr>
<tr>
<td>Jitter / Packet delay variation (PDV)</td>
<td>difference in EED between selected packets in a flow.</td>
</tr>
<tr>
<td>Packet Delivery Ratio (PDR)</td>
<td>Ratio between number of packets sent and received</td>
</tr>
<tr>
<td>Throughput</td>
<td>Number of packets arriving at the destination at a given time limit</td>
</tr>
<tr>
<td>Dropped Packets.</td>
<td>Number of packets failed to reach the destination while network transfer</td>
</tr>
<tr>
<td>Total Sent and Received Bytes</td>
<td>Number of data sent from the sender and received by the receiver</td>
</tr>
</tbody>
</table>

6. RESULTS AND DISCUSSIONS

The performance of the normal BIC congestion control algorithms, improved BIC_AWI algorithms and improved BIC_DPE has been evaluated under Iridium satellite constellation. The below mentioned following bar and line graphs shows the comparative performance of the three algorithms under Iridium satellite constellation.

Contention window dynamics

The size of the contention window with respect to time is used as an evaluation parameter for BIC, BIC_AWI and BIC_DPE congestion control algorithms. The simulation time is 300 seconds with a time gap of 0.5 seconds. The line graph depicts the graphical format of the
evaluation metrics under the Iridium satellite constellations. As shown in this chart, the performance of the two proposed algorithms are better than normal bic.

**Figure-7.** The contention window dynamics.

**Round trip time estimation**

Round Trip time is taken as an important parameter as it refers to the time taken for a signal to sent plus the length of the time for acknowledgement has to be considered. The graph shows the RTT of 3 congestion control algorithms.

**Figure-8.** The round trip time estimation.

**Number of bytes sent**

The total number of bytes sent by the BIC is 17304936, by BIC_AWI is 18716052 and by the BIC_DPE is 20913618 is depicted in the below bar graph.

**Figure-9.** The total sent bytes.

**Number of bytes received**

The total number of received bytes by the BIC is 17304936, by BIC_AWI is 18716052 and by the BIC_DPE is 20913618 is depicted in the below bar graph.

**Figure-10.** The total received bytes.

**Average throughput**

Throughput refers to the amount of bytes received by the destination. The average throughput is calculated by the throughput per unit of time by the formula \( \frac{\text{recvdSize}\times \text{duration}}{8/1000} \). The average throughput for BIC is 461.46 kbps and for BIC_AWI is 499.09 kbps and for BIC_DPE is 557.70 kbps which shows that BIC_DPE throughput is improved with respect to other 2 algorithms. The figure shows an improved throughput with respect to BIC_DPE Algorithm.

**Figure-11.** The average throughput.

**Packet delivery ratio / fraction**

The ratio of the number of packets delivered to the destination with respect to the packets sent by the source. The values are 99.38, 99.06 and 99.49 with respect to BIC, BIC_AWI and BIC_DPE.

**Figure-12.** The average PDF.
Total dropped packets
The total dropped packets with respect to BIC is 181 and with respect to BIC_AWI 299 and with respect to BIC_DPE is 195. The BIC_DPE algorithm drops just 14 packets higher than BIC and it is shown in Figure-13.

\[ \sum \text{ (arrive time – send time) / } \sum \text{ Number of connections} \]
The average EED of BIC is 441.30 ms, BIC_AWI is 451.54 and BIC_DPE is 452.57 ms.

Delay
It is the difference between the receiving time and sending time. The AWK script processes the trace file and delay is calculated. Average delay is:

\[ \text{Total Delay / total packet count} \]

The average delay of BIC is 24.36 ms, BIC_AWI is 22.38 and BIC_DPE is 21.07 ms. The below depicted Figure-14 shows average inter packet Delay over time. The performance in terms of delay in the case of proposed BIC_DPE is lower than the normal bic in most of the locations in this graph. That is why the proposed BIC_DPE was able to send and receive much packets and providing higher throughput than bic.

The end to end delay
The total time taken by the packet to reach the destination inclusive of route discovery delay, data queue is referred as end-to-end delay.

Jitter analysis
Packet delay variation (PDV) is the difference in end-to-end delay between selected packets in a flow with any lost packets being ignored (RFC 3393). The effect is referred to as jitter1.

\[ \text{Jitter1} = \text{jitter1} + \text{e2eDelay - prev_e2eDelay} \]

The average jitter1 of BIC is 7.63 ms jitter, BIC_AWI is 7.42 ms jitter and BIC_DPE is 7.66 ms jitter.

Figure-18 shows the Time vs Jitter1 (packet delay variation or PDV) and

Figure-20 shows the analysis between time Vs jitter1 and jitter2.
The above results shows that the proposed BIC_DPE performed very well under Iridium satellite constellation.

7. CONCLUSIONS

According to the results arrived, the overall performance of the proposed BIC_DPE was good under Iridium constellations compared to BIC and BIC_AWI algorithms. With respect to cwin and rtt dynamics, BIC_DPE realized better performance. The performance in terms of delay of BIC_DPE was lower than the normal bic. That is why the proposed BIC_DPE was able to send and receive much data and providing higher throughput than normal bic.

In the design of the update method of BIC, BICTCP B parameter is kept constant during the optimum TCP window size findings. In this model, it been dynamically changed with respect to acceleration and deceleration of RTT and the results had considerable improvements compared to BIC. Our future work will address this possibility and try to provide another improved congestion control algorithm for satellite networks.

REFERENCES


