ICC Research Group
Internet-Draft
Intended status: Experimental
Expires: September 21, 2015

J. You
Huawei

March 20, 2015

Increasing TCP's CWND based on Throughput
draft-you-iccrg-throughput-based-cwnd-increasing-00

Abstract

With 4K technology, we can get more details compared to traditional HD screens of the same size. However, 4K content increases the demand for higher network throughput, which poses a big challenge for delivering 4K content by TCP. This document proposes a target throughput based method for increasing TCP's congestion window (cwnd). Experimental results show that the proposed method can reach the required throughput much more rapidly than traditional TCP congestion algorithms, such as Reno, Cubic. Moreover, this method prevents cwnd from increasing when cwnd reaching to the required throughput.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 21, 2015.

Table of Contents

1. Introduction

Ultra HD (4K), or Ultra High Definition, is the next big step in HDTV resolution. 4K introduced as a new projection standard for digital cinema, bring digital cinema experience home with more details, more colors, higher frame rate and smoother experience.

Basically, 4K content streaming requires throughput no less than 45Mbps. Table 1 shows the main parameters for different kinds of 4K services when using traditional TCP [RFC793] congestion algorithms.

Table 1: Paramters for 4K Services

	Burst of Speed	Packet Loss Rate	Packet Delay
Quasi 4K	> 30Mbps	< 5.6*10exp(-7)	RTT < 65ms
Basic 4K	> 60Mbps	< 5.6*10exp(-7)	RTT < 35ms
Ultra 4K	> 112Mbps	< 5.6*10exp(-7)	RTT < 22ms

4K video can be encoded using different methods, such as CBR (Constant Bit Rate), VBR (Variable Bit Rate). CBR encodes each frame of video with the same bit rate, however, VBR adjusts the amount of bits assigned to a frame depending on what the encoder believes is happening in the frame. So the bit rate fluctuates over time when using VBR. For example, one tested 4K video using VBR encoding, its average bit rate is about 40Mbps and its maximum bit rate is about 60Mbps.

TCP uses slow start to increase the congestion window after a connection is initialized. It may start with an initial window of 10MSS [RFC6928]. During the slow start phase, TCP increases the congestion window very rapidly until packet loss occurs. TCP will enter the congestion avoidance phase.

Given target throughput = 50Mbps, RTT = 60ms, then target window size = 259MSS. During the slow start phase, the cwnd increases exponentially per RTT, i.e. 10, 20, 40, 80, ... Assume a packet loss occurs when cwnd = 130MSS. TCP Reno [RFC3782] cuts its congestion window in half, i.e. 65MSS, and switches to the congestion avoidance phase. In congestion avoidance phase, TCP's send rate increases linearly, i.e. cwnd = cwnd+1. It takes 194 RTTs until the sender's congestion window reaches to 259MSS, which needs about 11.64s. On the whole, TCP Reno needs 198 RTTs, i.e. about 11.88s, to reach to 50Mbps from the beginning of a transfer.

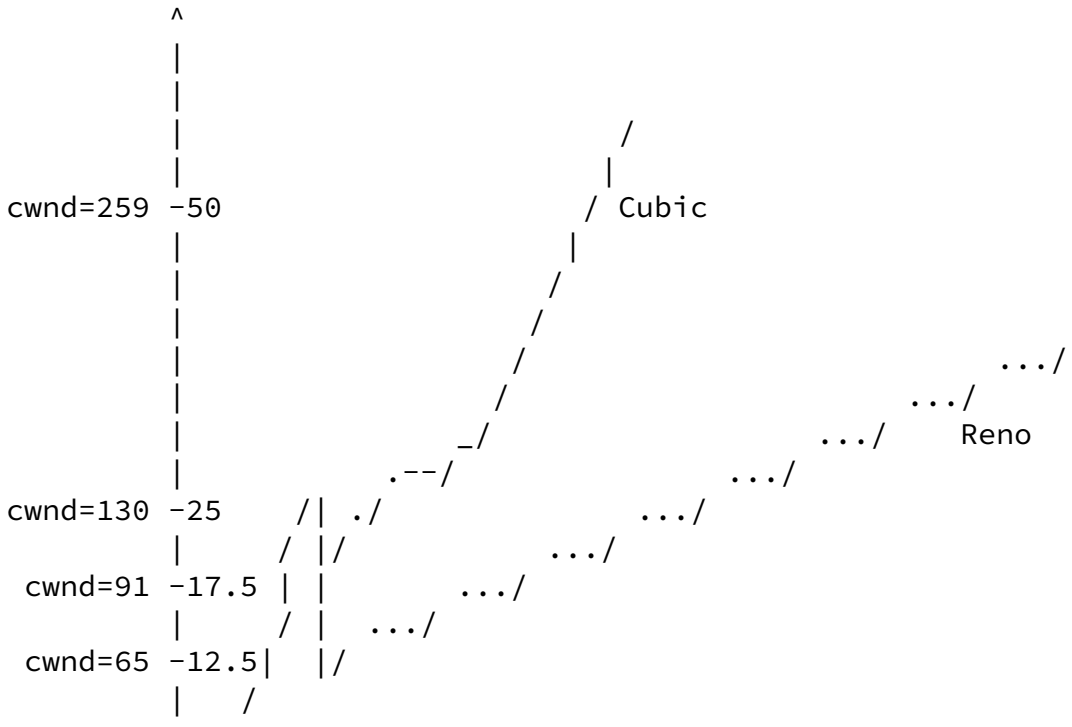
Another example is TCP Cubic [CubicTCP]. Similarly assume a packet loss occurs when cwnd = 130MSS, then the cwnd is reduced to (717/1024)*130 = 91MSS. The congestion window of Cubic is determined by the following function:

W(t) = C * (t-K)^3 + Wmax

where C is a scaling factor, t is the elapsed time from the last window reduction, Wmax is the window size just before the last window reduction, and

K = ((Wmax*beta)/C)^(1/3)

where beta is a constant multiplication decrease factor applied for window reduction at the time of loss event, i.e., the window reduction is beta*Wmax = (307/1024)*130 = 39MSS at the time of the last reduction. Calculate K=9.9 with C = 0.04, then 0.04*(t-9.9)^3+130 = 259, so t = 25. TCP Cubic needs 29 RTTs, i.e. about 1.74s, to reach to 50Mbps from the beginning of a transfer.



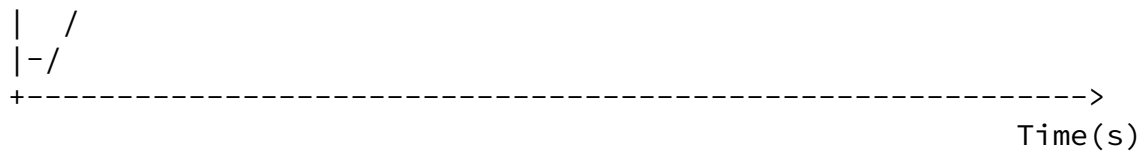


Figure 1: Cubic and Reno Window Curves

As shown in Figure 1, if packet loss occurs during the slow start phase, TCP Reno takes much longer time to reach the required throughput. TCP Cubic takes relatively shorter time compared to TCP Reno, but cwnd continues to increase even though it has reached to the target throughput.

This document proposes a target throughput based method for increasing TCP's congestion window. Experimental results show that the proposed method can reach the required throughput much more rapidly than traditional TCP congestion algorithms, such as Reno, Cubic. Moreover, this method prevents cwnd from increasing when cwnd reaching to the required throughput.

2. Terminology

This section contains definitions of term frequently used throughout this document.

4K: known as Ultra HD or UHD, is used to describe a new high resolution video format with a minimum resolution of 3840 x 2160 pixels in a 16 x 9 aspect ratio for any display.

TCP: Transmission Control Protocol

RTT: Round-Trip Time

CBR: Constant Bit Rate

VBR: Variable Bit Rate

3. Increasing CWND based on Throughput

This document proposes a target throughput based method for increasing TCP's congestion window. The main steps are shown in Figure 2.

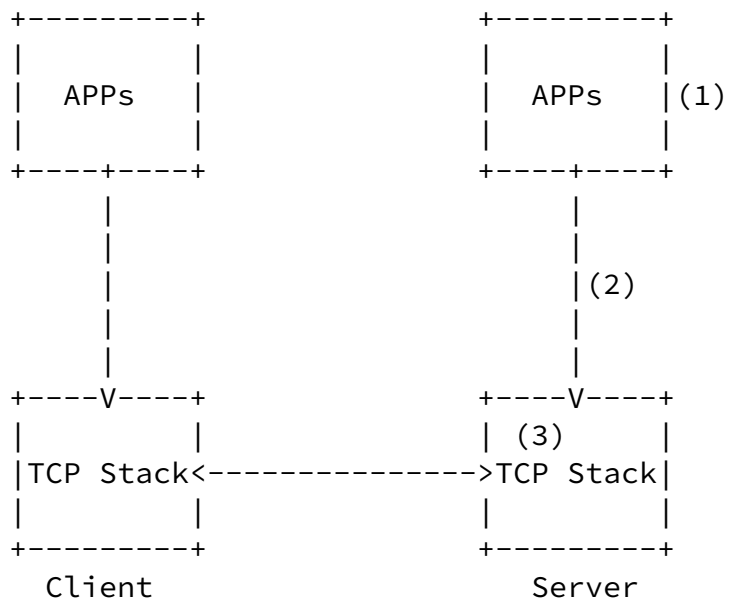


Figure 2: Increasing cwnd based on Throughput

Step 1: APP calculates the target throughput; take 4K VBR as an example,

TARGET_THROUGHPUT = e * BR

where e>1, is a multiplication factor.

Step 2: Extend TCP socket option: setsockopt(); add a new parameter: TARGET_THROUGHPUT, which will be transferred to TCP protocol stack.

Step 3: Calculate the increase factor alpha:

alpha = TARGET_THROUGHPUT * RTT - cwnd;

The increase factor for cwnd is calculated according to RTT and TARGET_THROUGHPUT. TARGET_THROUGHPUT is input by APP, while RTT is estimated based on current technology, which reflects the congestion state of the network. Hence,

cwnd = cwnd + alpha

The cwnd can be adjusted for every RTT, as shown in Figure 3.

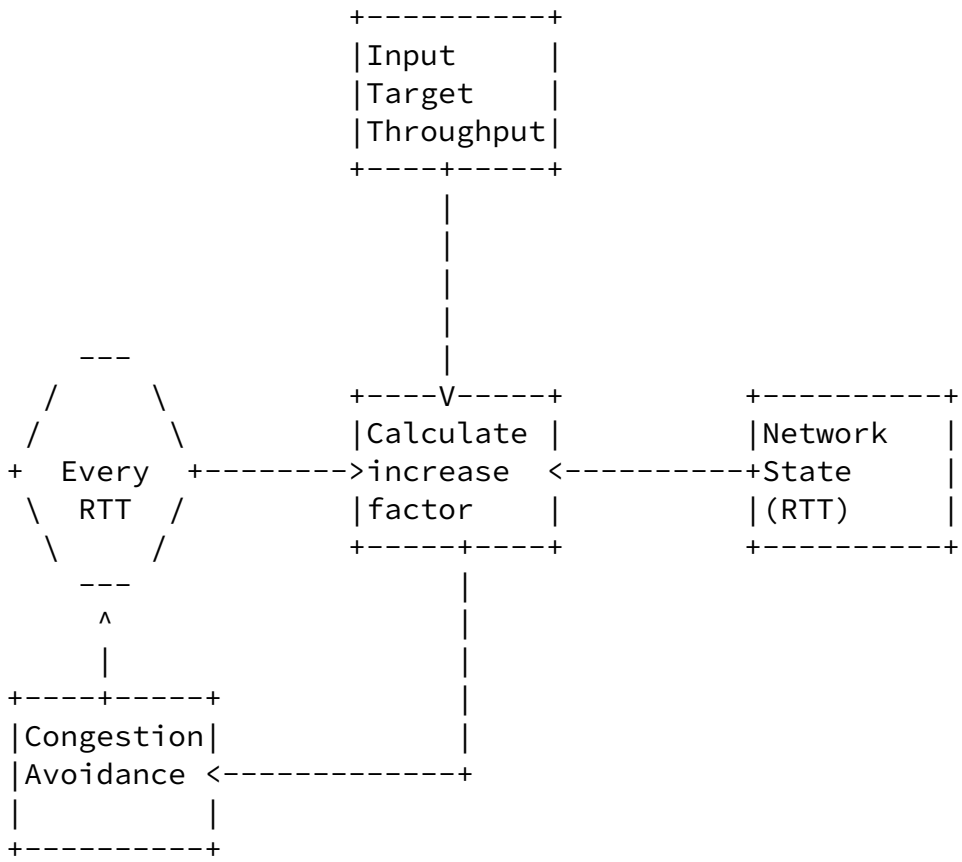


Figure 3: Procedures for Increasing cwnd

3.1. Window Growth Function

Alpha is an increase factor which is determined by the following function:

alpha = TARGET_THROUGHPUT * RTT - cwnd

However, protection mechanism needs to be considered for alpha, for example, when RTT is longer, the fast growth of cwnd may deteriorate the delay or congestion. So a threshold T is defined to balance the congestion state of the network. When BaseRTT =< RTT <= T; the proposed alpha function will be applied, otherwise, current window growth function (such as Reno, Cubic) will be applied, as shown in Figure 4.

In Figure 4, BaseRTT is estimated as the minimum observed RTT for the connection. Assume the TARGET_THROUGHPUT provide by the APP is no larger than the actual network capacity, then

TARGET_cwnd = TARGET_THROUGHPUT * BaseRTT

When cwnd reaches to TARGET_cwnd, Alpha is zero with RTT = BaseRTT.

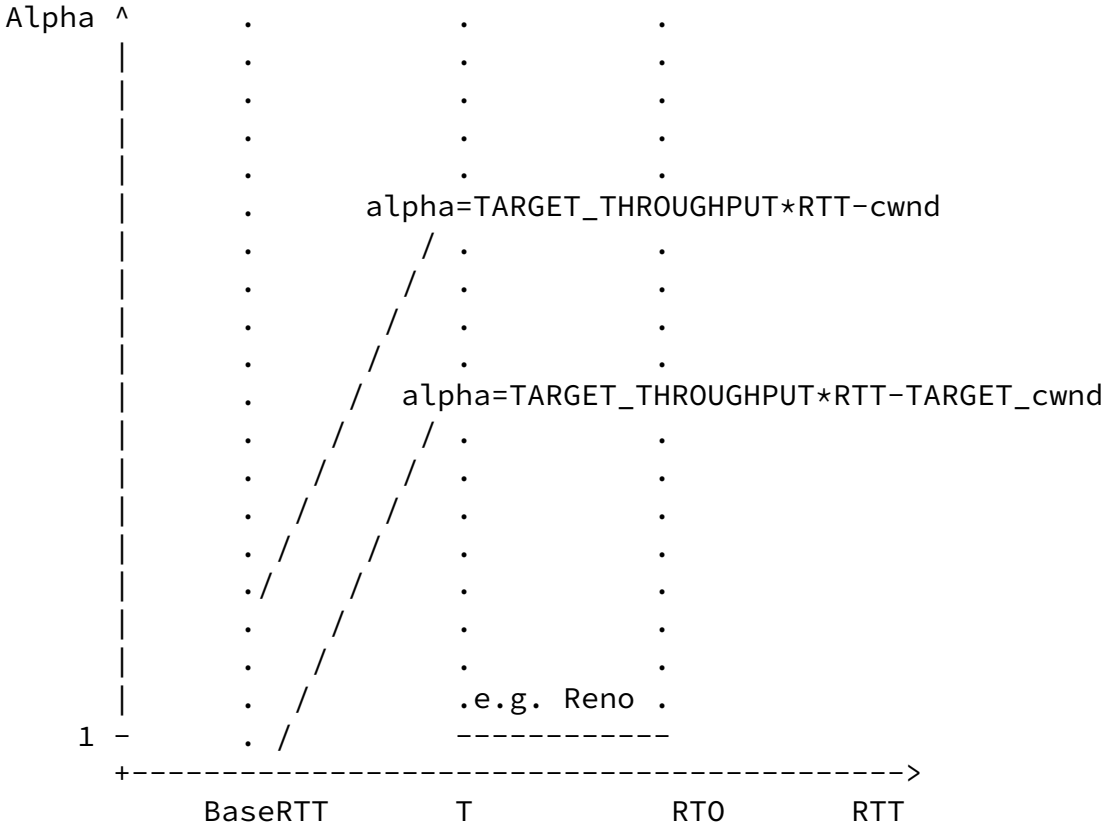


Figure 4: Alpha during Congestion Avoidance

When T < RTT <= RTO, alpha will is determined by current congestion algorithm, such as TCP Reno. RTO is the TCP retransmission timeout time. Suppose we still use the above-mentioned alpha formula, i.e. alpha = TARGET_THROUGHPUT * RTT - cwnd, then alpha will be larger. However, at that time, the state of network is not good as RTT is long. If still keeping large increase pace, it may deteriorate congestion. So when RTT is during this range, alpha should be in

inverse proportion to RTT. For example, when RTT = RTO, the network

is very congested; alpha could be 1 MSS.

3.2. Threshold

Threshold T is defined to balance the congestion state of the network. It has two main purposes:

- o prevent alpha from fast increasing in case TARGET_THROUGHPUT is larger than the actual network capacity. APP may overestimate the network capacity, and provide the incorrect TARGET_THROUGHPUT.

- o adjust T flexibly so as to adapt to different network.

When T = RTO, it shows the network capacity is enough to meet the TARGET_THROUGHPUT. When T = BaseRTT, the proposed method doesn't work.

4. Implementation Considerations

Given target throughput = 50Mbps, RTT = 60ms, then target window size = 259MSS. Assume initial cwnd = 10MSS, during the slow start phase,

```
alpha = TARGET_THROUGHPUT * RTT - cwnd
      = 50Mbps*60ms -10 MSS
      = 249MSS
```

Only one RTT is needed to reach to the target throughput. After, alpha will be zero if RTT is not changed. If packet loss occurs, TCP will cut its congestion window in half like Reno, i.e. to 130 MSS, and switches to the congestion avoidance phase. In congestion avoidance phase, the cwnd increases still according to:

```
alpha = TARGET_THROUGHPUT * RTT - cwnd
      = 50Mbps*60ms -130MSS
      = 129MSS
```

Still one RTT is needed to come back to the target throughput.

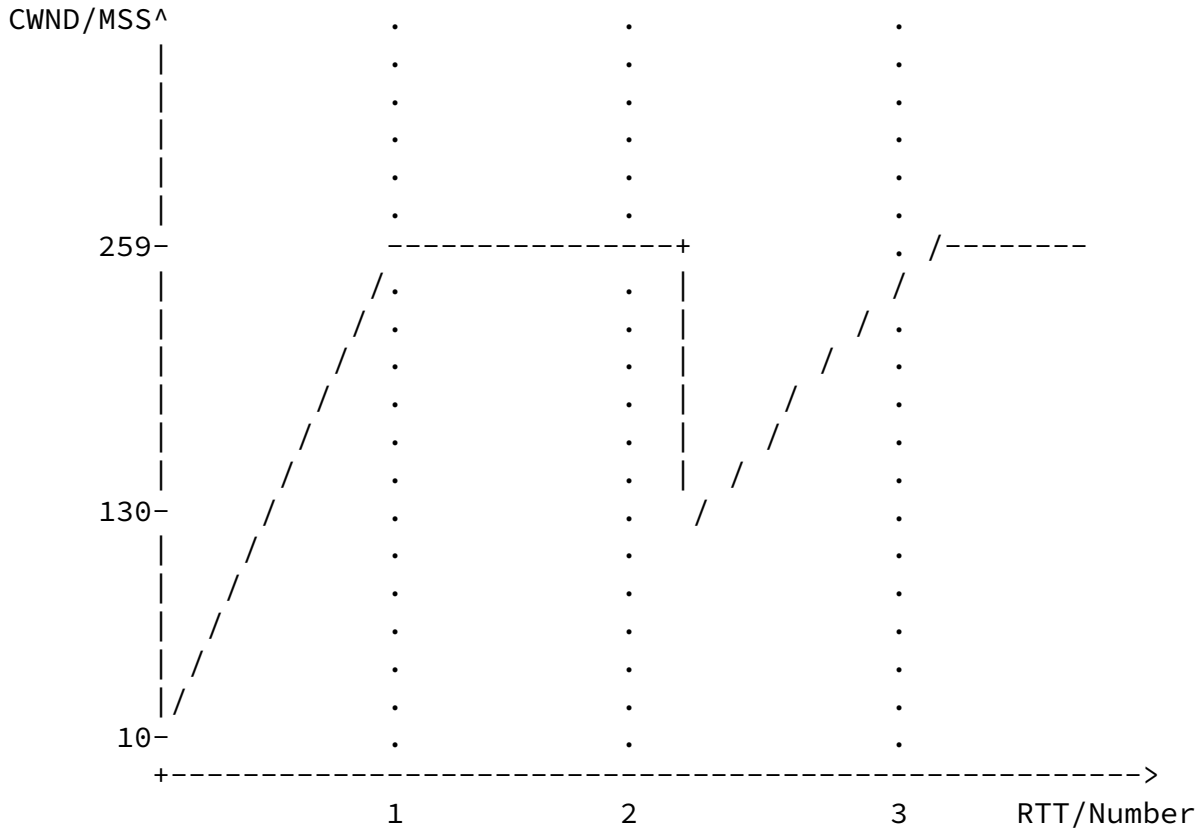


Figure 5: Window Curve with Packet Loss using Proposed Method

During the slow start phase, the growth of cwnd is not effected by packet loss if the ACK for first packet is received. In the first RTT, cwnd has already increased to the target cwnd when the sender receives the ACK for the first packet. If packet loss occurs in the third RTT, cwnd is reduced to half. Then in the following RTT, the cwnd rapidly increases to the target cwnd.

In order to support the alpha function in the sender, the receiver needs to set the receiving window according to the TARGET_THROUGHPUT too; otherwise the sender's cwnd may be limited by the receiving window.

5. IANA Considerations

This document has no actions for IANA.

6. Security considerations

This document introduces a new method to configure the congestion window for TCP connections. This method facilitates application developers to tune TCP for their benefits. But it also has the

possibility that packet loss may be caused by inappropriate cwnd setting if application overestimates the network capacity.

7. Acknowledgement

TBD.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3782] Floyd, S., Henderson, T., and A. Gurtov, "The NewReno Modification to TCP's Fast Recovery Algorithm", RFC 3782, April 2004.

[RFC6928] Chu, J., Dukkipati, N., Cheng, Y., and M. Mathis, "Increasing TCP's Initial Window", RFC 6928, April 2013.

[RFC793] Postel, J., "Transmission Control Protocol", RFC 793, September 1981.

8.2. Informative References

[CubicTCP] Ha, S., Rhee, I., and L. Xu, "CUBIC: A New TCP Friendly HighSpeed TCP Variant", 2008.

Author's Address

Jianjie You
Huawei
101 Software Avenue, Yuhua District
Nanjing 210012
China

Email: youjianjie@huawei.com