July 26, 2019

DYNAMIC TRANSMISSION CONTROL FOR IMPROVING MOBILE DATA USAGE

Tim Lin

Aaron Lee

DYNAMIC TRANSMISSION CONTROL FOR IMPROVING MOBILE DATA USAGE

ABSTRACT

This paper describes techniques for dynamically calculating values of one or more Transmission Control Protocol (TCP) memory buffer size variables (e.g., "tcp_mem," "tcp_rmem," and/or "tcp_wmem" variables), based on consideration of real-time network conditions, to achieve improved data usage of a mobile computing device. By dynamically determining and/or adjusting the values of TCP memory buffer size variables, the described techniques enable a mobile computing device (e.g., a mobile phone, tablet computer, wearable and/or headset device) to avoid sending too many in-flight packets that exceed network capacity, thereby reducing packet loss and the need for data retransmission from the mobile computing device. In some cases, the described techniques introduce and utilize a machine-learning model to predict suitable values of the dynamically determined TCP memory buffer size variables. The machine-learning model accepts a number of different features as inputs in order to produce a predicted output value of a memory buffer size variable. These features may include, for example, a specified time frame, real-time network allocated bandwidth, a geographic region (e.g., cell tower identifier or Global Positioning Satellite (GPS) location), and/or a packet loss rate, to name only a few examples.

DESCRIPTION

With the modern Internet, TCP is one of the main protocols that provides a reliable data exchange between communication devices. Various congestion control algorithms that have been widely deployed interpret packet loss as a signal indicating congestion. Network

congestion may occur when packets sent from a source device exceed the number of packets that a receiving device can handle, based on current network conditions. Packets may be stored in memory buffers on both the source and receiving devices, but even a temporary overflow of these buffers can lead to congestion, packet loss, data retransmissions, reduced data throughput, and/or performance degradation.

The Congestion Window (CWND) variable is one of the important TCP variables for rate control that determines the number of bytes that can be outstanding at any time for a TCP connection. More specifically, the CWND variable is a TCP state variable that limits the amount of data that a source device can send into the network via a TCP connection before receiving an acknowledgment from the receiving device. The use of the CWND variable may regulate data flow in TCP connections, minimize congestion, and improve network performance.

For a given connection or communication session, TCP typically increases the value of the CWMD variable over time to take advantage of available bandwidth between the source and receiving devices. However, once the size of the CWMD variable exceeds a certain value, the source device may send a volume of packets that exceeds current network capacity, which may result in dropped packets due to network congestion, and which may further then trigger data retransmission between the source and receiving devices in order to recover from packet loss. Once packet loss has occurred, TCP may drastically reduce the size value of the CWMD variable to avoid further loss.

In practice, one or more TCP memory buffer size variables (e.g., “tcp_mem,” “tcp_rmem,” and/or “tcp_wmem” variables) are variables that may be used to control and limit the growing size of the CWND variable. A properly determined size of TCP memory buffer size variables helps prevent packet loss and overshooting of in-flight packets, and thus mitigates the

problem of data retransmission. In current designs, the TCP memory buffer size variables typically have static or fixed values that are based on maximum theoretical throughput conditions for different service providers with various radio access technologies (e.g., Long-Term Evolution (LTE), Enhanced Data for Global Evolution (EDGE)). For example, Figure 1 below shows an assignment of various static/fixed TCP memory buffer sizes for various Radio Access Technology (RAT) for a certain carrier, and other carriers can provide their own configurations.

```
<!-- Configure mobile tcp buffer sizes in the form:
     rat-name:rmem_min,rmem_def,rmem_max,wmem_min,wmem_def,wmem_max
     If no value is found for the rat-name in use, the system default will be applied.
-->
<string-array name="config_mobile_tcp_buffers">
    <item>umts:131072,262144,1452032,4096,16384,399360</item>
    <item>hspa:131072,262144,2441216,4096,16384,399360</item>
    <item>hsupa:131072,262144,2441216,4096,16384,399360</item>
    <item>hsdpa:131072,262144,2441216,4096,16384,399360</item>
    <item>hspap:131072,262144,2441216,4096,16384,399360</item>
    <item>edge:16384,32768,131072,4096,16384,65536</item>
    <item>gprs:4096,8192,24576,4096,8192,24576</item>
    <item>1xrtt:16384,32768,131070,4096,16384,102400</item>
    <item>evdo:131072,262144,1048576,4096,16384,524288</item>
    <item>lte:524288,1048576,8388608,262144,524288,4194304</item>
</string-array>
```

Figure 1: RAT-based plus Carrier-based Configuration for TCP Buffer Sizes

In the example of Figure 1, two different memory buffer size variables are shown: “tcp_rmem” and “tcp_wmem”. Each variable may have three different values: a minimum value, a default value, and a maximum value. The “tcp_rmem” variable indicates the size of receive memory buffers used by a receiving device. The minimum receive memory buffer size is indicated by the value “rmem_min,” the default receive memory buffer size is indicated by the value “rmem_def,” and the maximum receive memory buffer size is indicated by the value “rmem_max.”

Similarly, the “tcp_wmem” variable indicates the size of write memory buffers used by a source device. The minimum write memory buffer size is indicated by the value “wmem_min,” the default write memory buffer size is indicated by the value “wmem_def,” and the maximum write memory buffer size is indicated by the value “wmem_max.”

As shown in Figure 1, with various existing designs, all of the values of the “tcp_rmem” and “tcp_wmem” variables (e.g., the minimum, default, and maximum values) are hard-coded, fixed values based on maximum theoretical throughput conditions for different service providers with various radio access technologies (e.g., UMTS, HSPA, HSUPA, HSDPA, HSPAP, EDGE, GPRS, LXRTT, EVDO, LTS). However, these fixed and hard-coded values are less helpful in addressing the overshooting problem in live networks, because they are static, non-changing values that are based on maximum theoretical conditions. In practice, network bandwidth or capabilities may change with time, location and/or environment conditions. Static values of TCP memory buffer variables such as “tcp_rmem” and “tcp_wmem” cannot typically address all live network situations, and the use of static values may introduce packet retransmission due to the window overshooting problem identified above. For mobile users with limited mobile data plans, packet retransmission may unnecessarily consume traffic and may also reduce utilization efficiency.

For example, Figure 2 below shows a graph of time (seconds) versus window size (bytes) for live network activity, where each plot indicates the number of in-flight packets.

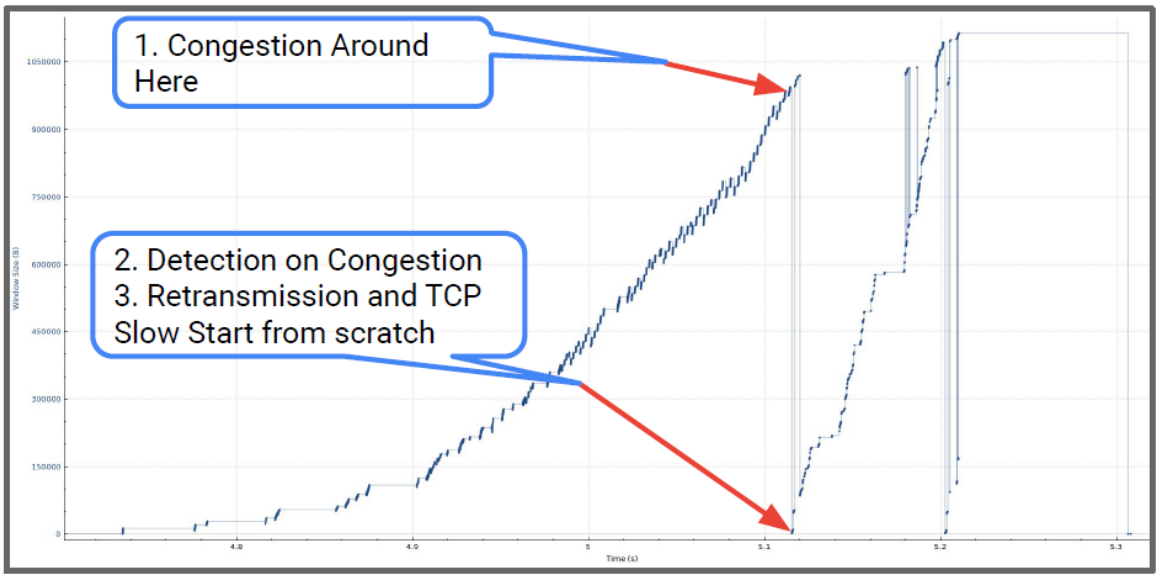

Figure 2: Time Versus Window Size For In-flight Packets

Over time, as shown in Figure 2, the window size (i.e., CWND) increases in size during packet transmission to take advantage of available bandwidth between the source and receiving devices. At point 1 ("1. Congestion Around Here"), the source device may send a volume of packets that exceeds current capacity, which may result in dropped packets due to network congestion. Once packet loss has occurred, the source device may drastically reduce the size value of the CWMD variable to avoid further loss ("2. Detection on Congestion") and perform data retransmission between the source and receiving devices in order to recover from packet loss ("3. Retransmission and TCP Slow Start from scratch"). As a result, packet retransmission due to window overshooting will impose additional data packet consumption. In the particular example of Figure 2, a 2.08MB upload task may actually, and unnecessarily, consume 2.53MB from a user's data plan and reduce utilization efficiency.

To address these issues, the techniques described in the present paper enable the use of dynamically determined values of TCP memory buffer size variables that account for real-time network conditions. By dynamically determining and/or adjusting TCP memory buffer size

variables, the described techniques enable a mobile computing device to avoid sending too many in-flight packets over and above current network capacity, even during times of network congestion, thereby reducing data retransmission from the mobile computing device. As a result, the disclosed techniques can help reduce packet loss and data retransmission issues. In certain cases, such as that shown below in Figure 4, the disclosed techniques may also increase data throughput and performance. In certain other cases, data throughout may be maintained or slightly lowered in order to prevent packet loss.

In some cases, the described techniques introduce and utilize a machine-learning model to predict suitable values of the TCP memory buffer size variables. The machine-learning model can be trained with labeled datasets based on historical throughput data, which may be converted or otherwise used to determine associated TCP memory buffer size information for inclusion in the model. After the machine-learning model is trained, it is configured to accept a number of different features as inputs in order to produce predicted output values of one or more memory buffer size variables (e.g., "tcp_mem," "tcp_rmem," and/or "tcp_wmem" variables). These features may include, for example, a specified time frame, such as particular times, hours, days, weeks, months, or other time periods or ranges. These input features may also include real-time network allocated bandwidth information for a given network, and/or or geographic region information (e.g., cell tower identifier information or Global Positioning Satellite (GPS) location information) that is associated with the source device, receiving device, or network area in general. Additional example features may include real-time packet loss rates, modem bandwidth information, and/or other physical layer bandwidth information associated with the source device, receiving device, or network area in general

Figure 3, as shown below, includes a conceptual diagram illustrating various aspects of the techniques in the present paper.

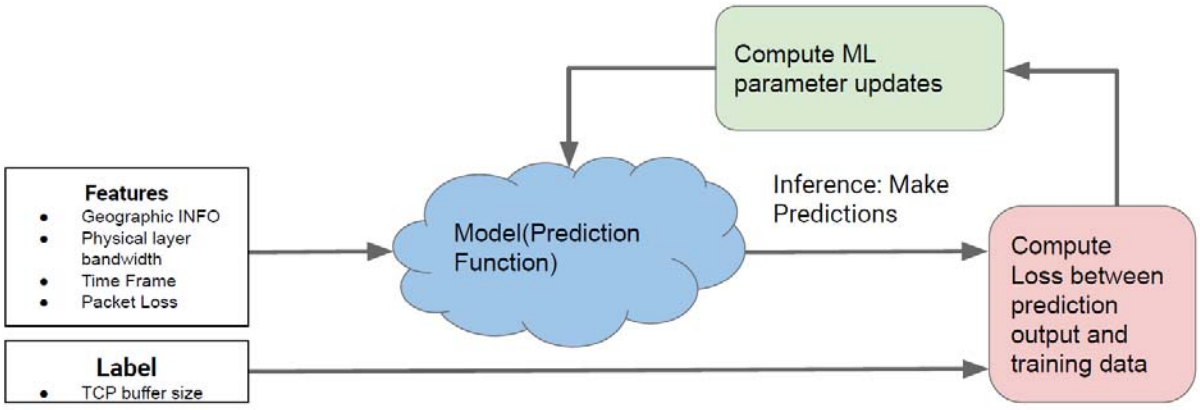

Figure 3: System For Predicting Values of Memory Buffer Variables

As shown in Figure 3, the machine-learning model may provide a prediction function that is configured to make predictions or inferences of the dynamically determined values of TCP memory buffer variables. The machine-learning model can be trained with labeled datasets used to determine associated TCP memory buffer size information used in the model, and the model is configured to accept a number of different features as inputs in order to produce predicted output values of the memory buffer size variables, such as described above.

The system may also refine the model based on a computed loss between the predicted output and the initial training data to determine how well the predicted values achieved overall lower packet loss and retransmission rates within the system. In some examples, the predicted or otherwise dynamically determined values of the TCP memory buffer variables may be maximum values that are used in the system (e.g., dynamically determined “rmem_max” and/or “wmem_max” values), which are used instead of the pre-defined, static values illustrated in

Figure 2. Based on the computed loss between the predicted output and the training data, the system can refine the machine-learning model by computing parameter updates that are periodically fed back into the model in real time. As a result, the disclosed techniques provide dynamically determined values of TCP memory buffer size variables to fit existing, real network conditions, and help reduce packet transmission to enable mobile users to more effectively make use of their data plans.

For example, Figure 4 below shows a graph of time (seconds) versus window size (bytes) for live network activity utilizing the techniques of the present paper, where each plot indicates the number of in-flight packets.

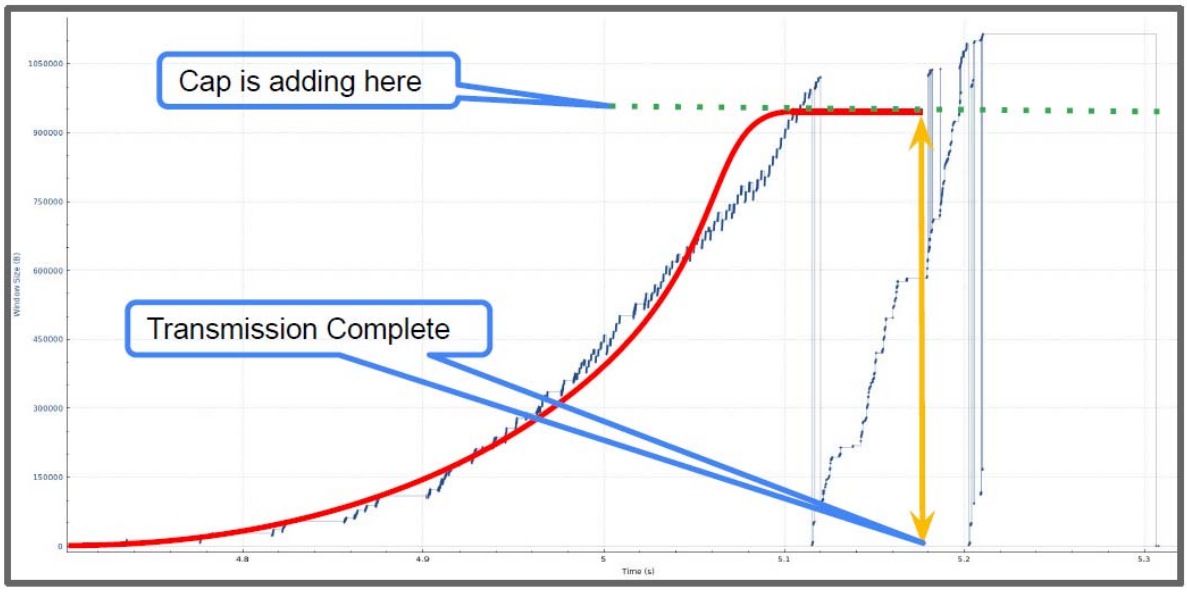

Figure 4: Time Versus Window Size Using Dynamic Buffer Sizes

Similar to Figure 2, Figure 4 shows that the window size (i.e., CWND) increases in size during packet transmission to take advantage of available bandwidth between the source and receiving devices. However, through use of the present techniques to dynamically determine the values of

the TCP memory buffer size variables (e.g., "tcp_mem," "tcp_rmem," and/or "tcp_wmem" variables), the techniques provide a cap or ceiling on the size of one or more of the TCP memory buffer sizes. This, then, effectively creates an overall cap on the value of the window size, as shown in Figure 4 by the dotted green line ("Cap is adding here"), given that one or more of the buffer size values may be used in calculating the window size value. By capping the value of the window size at a particular threshold, the source device is unable to increase the value of the window size beyond this threshold over time, as indicated by the red curve, which will then prevent overshooting, packet loss, and packet retransmission, even in view of existing network congestion. Once transmission is complete, as indicated by the yellow arrow in Figure 4, the system may revert the window size value to an initial value (e.g., an initial value determined by TCP), for future data transmissions between the source and receiving devices. As a result, by dynamically determining and/or adjusting TCP memory buffer size variables, the described techniques enable a mobile computing device to avoid sending too many in-flight packets over and above current network capacity, even during times of network congestion, thereby reducing data retransmission from the mobile computing device. Fewer packet retransmissions will result in a more efficient use of users' mobile data plans, cost savings over time, and improved speeds for completion of data transmissions due to fewer lost packets.

REFERENCES

1. K. Hiraki and J. Kuroda, "Adaptive TCP Congestion Control Algorithm Using Multi-Layer Perception," June 1, 2019.

2. P. Barford, M. Mirza, J. Sommers, and X. Zhu, "A Machine Learning Approach to TCP Throughput Prediction," June 12-16, 2007.