Optimizing Data Capture and Delivery WAN Performance

Albert Tsai* and Ricardo Turcios*

ABSTRACT. — Congestion control algorithms are used in computer networks to regulate and manage the flow of data traffic, preventing network congestion and ensuring efficient data transmission. Due to their conservative nature, traditional congestion control algorithms often demonstrate suboptimal performance in high-latency and lossy network environments, as they do not fully leverage the available network bandwidth. This article discusses the progress made in optimizing the performance of the Data Capture and Delivery (DCD) application, a vital data management component within the Deep Space Network (DSN), over a wide-area network (WAN) characterized by a high-latency and lossy network environment. The methodology employed involves setting up a controlled testing environment that accurately replicates the challenging network conditions. Then through systematic experimentation, different congestion control algorithms are evaluated based on their ability to maintain optimal throughput while effectively handling random packet loss. Key performance metrics such as throughput, latency, and loss rate are measured and analyzed to assess the algorithms' effectiveness. Preliminary results indicate that the Bottleneck Bandwidth and Round-Trip Time (BBR) algorithm exhibits superior performance in high-latency and lossy environments when compared with other algorithms. However, continued testing and analysis are required to validate these initial observations and draw robust conclusions, and further refinement of the algorithm is recommended for network performance optimization.

I. Introduction

A computer network consists of a set of devices connected by a communication link. These devices can be computers, servers, switches, routers, or any other networking equipment. Data are transmitted from one device to another in the form of packets. In a network with limited bandwidth, the transmission of large volumes of data can cause congestion because the volume of data being transmitted through a network exceeds the network's capacity to handle it. When this occurs, the quality of the communication decreases because congestion results in a bottleneck where data transmission is delayed or even blocked entirely. This can lead to reduced network performance, increased latency, and packet loss. Congestion control is a technique used to prevent network congestion from occurring and to ensure that network performance is maintained at an optimal level.

* Mission Control Systems Section.

The research described in this publication was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

Congestion control algorithms work by monitoring the traffic flow and regulating the rate at which data are transmitted. These algorithms detect congestion by analyzing factors such as packet loss and delay. *Packet loss* is identified when the sender fails to receive acknowledgment messages from the receiver confirming the successful reception of transmitted packets, and *delay* is measured through the calculation of round-trip time, which is the time taken for a packet to travel from the sender to the receiver and for the acknowledgment message to come back. Once congestion is detected, the algorithm adjusts the transmission rate to alleviate the congestion. The goal of congestion control is to maximize network performance while avoiding congestion. There are several congestion control algorithms available, including Reno, Vegas, and New Reno. These algorithms have different approaches to detecting and mitigating congestion. For example, Reno detects congestion by monitoring packet loss, while Vegas uses delay as an indicator of congestion. In summary, congestion control is a technique used to regulate the flow of data through a network and prevent congestion from occurring. Congestion control algorithms are used to monitor network traffic and adjust the transmission rate to ensure that network performance is optimized.

The Deep Space Network (DSN), consisting of an international array of antennas that supports interplanetary spacecraft missions, plays a vital role in commanding our spacecraft and retrieving their data on Earth. It comprises three antenna facilities spaced at equal distances from each other and located in Goldstone, California; Madrid, Spain; and Canberra, Australia. The strategic placement of these facilities enables continuous communication with spacecraft as Earth rotates. Within the expansive framework of the DSN, the Data Capture and Delivery (DCD) application serves as the backbone for managing the flow of data. DCD is tasked with receiving, storing, and transmitting data between spacecraft, antenna facilities, and mission teams. In fulfilling this crucial role, DCD leans on computer networks and congestion control algorithms to transport captured spacecraft data from antenna sites to Jet Propulsion Laboratory (JPL).

II. Limitation

Traditional congestion control algorithms are designed to take a conservative approach and significantly reduce throughput when indication of congestion (such as a packet loss) is encountered to avoid congesting the network. Due to their conservative nature, traditional congestion control algorithms often demonstrate suboptimal performance in high-latency and lossy network environments, as they do not fully leverage the available network bandwidth. The DSN encounters challenges posed by such limitations. We experience notable decreases in throughput when transmitting data from the DSN antenna sites to JPL due to random losses in the wide-area network (WAN). This notable reduction in throughput is attributed to the conservativeness of traditional congestion control algorithms, which struggle to adapt optimally to the random-loss conditions in the network. However, the DCD application, tasked with facilitating the process of receiving, storing, and transmitting data between spacecraft and individual mission teams, has a more aggressive requirement and needs an algorithm that uses the entire available bandwidth as much as possible. Thus, we explore two approaches to optimize the performance of the WAN that connects oversea complexes to JPL. The first approach involves identifying an existing congestion control algorithm that is better suited for the

DSN. The second approach is to modify an existing congestion control algorithm, aiming to utilize the entire available bandwidth as efficiently as possible, even in high-latency and lossy network environments.

III. Method 1 and Results

The first step in accomplishing the above goal is to understand the current state of the network and the challenges that exist, which involves identifying the bandwidth, latency, and loss rates of the network. Once these have been identified, a controlled testing environment that accurately replicates the challenging network conditions was set up using Netropy, a network simulation tool used to emulate various network conditions such as latency, bandwidth, and packet loss. We proceeded by exploring which existing congestion control algorithms are better suited for DSN through systematic experimentation using iPerf, a network testing tool that measures network throughput.

Different congestion control algorithms are evaluated based on their ability to maintain optimal throughput while effectively handling random packet loss. Key performance metrics such as throughput, latency, and loss rate are measured and analyzed to assess the algorithms' effectiveness. Test results indicate that the Bottleneck Bandwidth and Round-Trip Time (BBR) algorithm exhibits superior performance in high-latency and lossy environments when compared with other algorithms. Notably, BBR exhibited exceptional performance across all test scenarios involving a lossy network environment, outperforming all other algorithms by a significant margin as presented in Table 1 to Table 4 below. BBR's exceptional performance is expected as the algorithm is renowned for its impressive resilience to high loss rates, boasting an ability to accommodate up to 15% loss. BBR's tolerance to high loss rates stems from its unique approach of not relying on packet loss as an indicator of congestion. This allows BBR to overcome the limitations of traditional congestion control algorithms, resulting in improved performance even in lossy network environments. However, BBR's design to maximize throughput and minimize delay can lead to aggressive behavior in terms of bandwidth consumption. In scenarios where fairness among different flows or users is a critical concern, BBR might not be the most suitable choice. In addition, BBR is unavailable in many existing operating systems, so transitioning to BBR may require difficult adjustments that outweigh the benefits.

Table 1. Simulating Madrid Deep Space Communications Complex (MDSCC) with 93 Mbps and 1% loss rate.

Throughput	Algorithm (sender)	Latency	Loss
64.6 Mbits/sec	BBR	160ms	1%
0.832 Mbits/sec	Reno	160ms	1%
0.886 Mbits/sec	Cubic	160ms	1%
1.95 Mbits/sec	Scalable	160ms	1%
0.652 Mbits/sec	Highspeed	160ms	1%
3.89 Mbits/sec	Hybla	160ms	1%
6.56 Mbits/sec	Illinois	160ms	1%
7.21 Mbits/sec	Westwood	160ms	1%

Table 2. Simulating MDSCC with 93 Mbps and 0.5% loss rate.

Throughput	Algorithm (sender)	Latency	Loss
69.7 Mbits/sec	BBR	160ms	0.5%
1.27 Mbits/sec	Reno	160ms	0.5%
1.44 Mbits/sec	Cubic	160ms	0.5%
2.52 Mbits/sec	Scalable	160ms	0.5%
0.976 Mbits/sec	Highspeed	160ms	0.5%
5.75 Mbits/sec	Hybla	160ms	0.5%
9.52 Mbits/sec	Illinois	160ms	0.5%
13.1 Mbits/sec	Westwood	160ms	0.5%

Table 3. Simulating MDSCC with 93 Mbps and 0.1% loss rate.

Throughput	Algorithm (sender)	Latency	Loss
73.9 Mbits/sec	BBR	160ms	0.1%
3.12 Mbits/sec	Reno	160ms	0.1%
4.63 Mbits/sec	Cubic	160ms	0.1%
9.41 Mbits/sec	Scalable	160ms	0.1%
2.77 Mbits/sec	Highspeed	160ms	0.1%
13.7 Mbits/sec	Hybla	160ms	0.1%
21.7 Mbits/sec	Illinois	160ms	0.1%
37.5 Mbits/sec	Westwood	160ms	0.1%

Table 4. Simulating MDSCC with 93 Mbps and 0% loss rate.

Throughput	Algorithm (sender)	Latency	Loss
84.4 Mbits/sec	BBR	160ms	0%
80.1 Mbits/sec	Reno	160ms	0%
87.9 Mbits/sec	Cubic	160ms	0%
84.3 Mbits/sec	Scalable	160ms	0%
87.1 Mbits/sec	Highspeed	160ms	0%
81.6 Mbits/sec	Hybla	160ms	0%
87.8 Mbits/sec	Illinois	160ms	0%
88.0 Mbits/sec	Westwood	160ms	0%

IV. Method 2 and Result

In addition to comparing existing congestion controls algorithms, we built a customized kernel with modified Transmission Control Protocol (TCP) modules to implement and evaluate a custom congestion control algorithm derived from the Scalable algorithm. Modifying TCP modules allowed us to have more control and flexibility in enhancing the algorithm's aggressiveness by adjusting the parameters controlling additive increase and multiplicative decrease. By default, the Scalable algorithm increments the congestion window by 1 for every 100 acknowledgments and reduces it by a factor of 1/8 during loss events. To boost the WAN throughput, we tuned these parameters to accelerate the

congestion window growth (incrementing by 1 for every acknowledgment) during additive increase, while reducing its decrease (by a factor of 1/(2^30)) during packet loss.

However, upon testing the modified algorithm with the rebuilt kernel, we discovered a decrease in throughput contrary to our expectations. Moreover, the results highlighted an increase in packet retransmission, suggesting that the algorithm's excessive aggressiveness caused the congestion window to consistently exceed the available bandwidth. As a consequence, higher packet loss occurred, resulting in decreased throughput. To address this problem, we imposed an upper bound for the congestion window size to prevent the algorithm from exceeding the available bandwidth. This adjustment effectively resolved the issue of the algorithm overshooting its congestion window size, allowing the algorithm to achieve a substantial increase in throughput as illustrated in Table 5 and Table 6 below. However, while the modified Scalable algorithm shows improved throughput performance, it still falls short of the optimal throughput achieved by the BBR algorithm. Thus, BBR is selected as the preferred algorithm due to its superior performance.

Table 5. Default Scalable performance.

Throughput	Algorithm (sender)	Latency	Loss
1.95 Mbits/sec	Scalable	160ms	1%
2.52 Mbits/sec	Scalable	160ms	0.5%
9.41 Mbits/sec	Scalable	160ms	0.1%
84.3 Mbits/sec	Scalable	160ms	0%

Table 6. Modified Scalable performance.

Throughput	Algorithm (sender)	Latency	Loss
38.9 Mbits/sec	Scalable	160ms	1%
48.8 Mbits/sec	Scalable	160ms	0.5%
67.2 Mbits/sec	Scalable	160ms	0.1%
86.9 Mbits/sec	Scalable	160ms	0%

V. Integration

We then proceeded with exploring viable methods for executing the DCD application using the BBR algorithm. Currently, DCD operates on Solaris 11, while BBR is only available on Red Hat Enterprise Linux (RHEL) and Ubuntu, both of which are derived from the Linux operating system. Thus, we explored several approaches to enable the execution of DCD with this algorithm. In our exploration, we considered the option of using port forwarding via Secure Shell (SSH) protocol to streamline the transmission of application data through a virtual machine running on RHEL. With this setup in place, we can leverage BBR's availability on RHEL, thereby facilitating the utilization of BBR as the default congestion control algorithm for the DCD application. However, we encountered a limitation with SSH, as it has a fixed SSH window size of 2MB. Upon testing, we confirmed that this constraint leads to a bottleneck in throughput and negatively impacts WAN performance.

We overcame this challenge by leveraging port forwarding via SOcket CAT (SOCAT), a command-line utility that establishes two bidirectional byte streams and transfers data between them. This approach allows us to bypass the constraints of having a fixed SSH window size. Additionally, to ensure a viable solution for situations where no alternative is available, we developed our own port forwarding application by utilizing DSN's Data Transport Subsystem library, which handles data transport between endpoints by facilitating the establishment of connections as well as the reading and writing of messages between them. Following the integration of BBR into the existing infrastructure, we reconducted performance testing on the algorithm using the DCD application instead of iPerf to better simulate real operation. A summary of test results is provided in Table 7 and Table 8 below. BBR significantly outperformed its preliminary testing with iPerf, achieving throughputs of 65Mbps, 75 Mbps, and 90 Mbps at 10%, 5%, and 1% loss rate, respectively (Table 8).

Table 7. BBR performance in preliminary testing with iPerf.

Throughput	Algorithm	Latency	Loss
84.4 Mbits/sec	BBR	160ms	0%
73.9 Mbits/sec	BBR	160ms	0.1%
69.7 Mbits/sec	BBR	160ms	0.5%
64.6 Mbits/sec	BBR	160ms	1%

Table 8. BBR performance using DCD application.

Throughput	Algorithm	Latency	Loss
93 Mbps	BBR	160ms	0%
91 Mbps	BBR	160ms	0.1%
90 Mbps	BBR	160ms	0.5%
90 Mbps	BBR	160ms	1%
75 Mbps	BBR	160ms	5%
65Mbps	BBR	160ms	10%

Lastly, we tested BBR on a different network setup with a capacity of 300 Mbps to assess its performance in future missions. The results are displayed in Table 9 below. BBR demonstrated tolerance to a 1% loss rate while achieving the mission requirement of 170 Mbps. This is a substantial improvement from the current algorithm, Highspeed, which can only tolerate a loss rate of 0.001%.

Table 9. BBR performance using DCD application and 300 Mbps network capacity.

Throughput	Algorithm	Latency	Loss
300 Mbps	BBR	160ms	0%
225 Mbps	BBR	160ms	0.1%
215 Mbps	BBR	160ms	0.5%
205 Mbps	BBR	160ms	1%
145 Mbps	BBR	160ms	5%
108 Mbps	BBR	160ms	10%

VI. DSN Requirements Testing

While average throughput is important, mission requirements focus on the latency in data delivery, which is more sensitive to intermittent drops below the input rate. However, these intermittent drops might not always be evident during throughput testing. Therefore, testing the expected operational data rates and verifying the latency are the ultimate tests that validate our ability in meeting the specified requirements. Additionally, due to the dynamic nature of the loss rate on the WAN, setting a specific target loss rate is challenging. Thus, the objective is to maximize the tolerance for loss rate as much as possible.

For James Webb Space Telescope (JWST), the expected data transmission rate is 31 Mbps, and the latency for the transmission should be kept under 180 seconds. The current algorithm, Highspeed, is able to tolerate a loss rate of 0.1% while meeting this requirement. After testing, we verified that BBR is able to tolerate a 14% loss rate while meeting this requirement, demonstrating an over 100 times improvement.

In addition to testing BBR using JWST, we also evaluated the algorithm's performance in future missions. DSN anticipates a bandwidth increase to 300 Mbps in the future and expects spacecraft that utilize this bandwidth. To prepare for this, we evaluated how this solution performs under future network capacities. For the future Lunar Gateway mission, the expected data transmission rate is 100 Mbps (113 Mbps with telemetry header overhead); however, a data transmission rate of up to 150 Mbps (170.1 Mbps with telemetry header overhead) is permitted. Thus, we performed the requirement testing at a rate of 170.1 Mbps, with the latency for the transmission kept under 2 seconds. The current algorithm, Highspeed, is able to tolerate a loss rate of 0.001% while meeting this requirement. In contrast, BBR can tolerate a 1.1% loss rate, achieving an over 1000 times improvement.

VII. Conclusion

In conclusion, through systematic experimentation and analysis, we compared various existing algorithms and their suitability for DSN. Our findings indicate that the BBR algorithm emerged as the most promising choice, demonstrating superior throughput and tolerance to lossy network environments. The modified Scalable algorithm also shows improvements but fell short of the optimal performance achieved by BBR. Furthermore, we successfully incorporated BBR into the existing network infrastructure by utilizing port forwarding through a virtual machine running RHEL, and BBR demonstrated its ability to meet current and future DSN requirements tests while tolerating higher loss rates, making it a better choice over the current algorithm.

Acknowledgments

The authors would like to express our gratitude for the financial support received from NASA JPL and Caltech as their investment has made this research possible. In addition, the authors would like to thank Jeff Berner for his valuable review comments on this work.