**CS204 Project Report**

**1. Title of the project:** Comparison and analysis of different Quality of Service (QoS) techniques in a simulated network environment with different traffic types and analysis of the level of impact it creates as compared to not using QoS.

**2. Team members' names:**

Manoj Nagarajappa - mnaga024

Puneet Singhania- psing088

**3. Abstract:**

Our project performs the comparison and analysis of the impacts of different Quality of Service (QoS) techniques, such as priority queueing and token bucket filtering, on a simulated network environment. The challenge of effectively managing data traffic becomes crucial and interesting as a result of the growing complexity of modern networks and an increasing number of applications with diverse needs competing for shared resources. In order to simulate real-world network conditions, this project utilizes Mininet for network simulation and applies several Quality of Service (QoS) techniques to the hosts. Various types of traffic are generated and utilized for analysis and comparison. We use Wireshark to capture and analyze packets, which enables us to determine important performance metrics like latency, packet loss, and throughput using various QoS techniques. This project seeks to provide an in-depth understanding of QoS implementation, hence contributing to effective network resource management strategies. It does this by carrying out a thorough comparison of these metrics with and without the application of QoS techniques.

**4. Introduction:**

The main goal of this project is to analyze and compare various Quality of Service (QoS) techniques in a simulated network environment, with a focus on priority queueing and token bucket filtering. The primary objective is to monitor the effects that these techniques have on network performance metrics like latency, packet loss, and throughput, particularly when the network is subjected to a variety of traffic.

The project seeks to provide answers to the following questions:

1. What impact do QoS techniques like priority queueing and token bucket filtering have on a network's throughput, packet loss, and latency?

2. How effectively do these QoS techniques function when dealing with different kinds of traffic, such as bulk data transfers and web browsing?

3. What is the network's comparative performance without making use of QoS techniques?

Given how crucial it is to optimize network performance in today’s interconnected world, these questions are of the utmost importance. Network administrators and engineers can decide on network configuration and resource allocation with knowledge of the behavior and effects of various QoS techniques, resulting in effective and reliable network performance that is tailored to the necessities of various applications.

The project significantly contributes by performing a comparative analysis of many QoS techniques under the same network conditions, in contrast to existing or earlier works that typically focus on analyzing specific QoS techniques in isolation. Furthermore, it evaluates their performance not only in isolation but also against the baseline of a network without QoS optimization. This approach provides a more thorough viewpoint, enabling a greater understanding of the synergies and trade-offs between various QoS techniques and how they can be used most effectively in practical situations. Additionally, this project combines tools like Mininet for simulation and Wireshark for packet analysis, which enables a deeper, more useful insight into how QoS techniques are applied and how they perform.

**5. Related Works:**

The topic of Quality of Service (QoS) in networking has been the subject of numerous studies and research. These studies have devoted a large amount of time to understanding and enhancing particular QoS techniques.

For instance, priority queueing has been the subject of numerous studies. Priority queueing has been studied in the context of real-time traffic, such as Voice over IP (VoIP) and video conferencing, for its effectiveness in lowering latency and guaranteeing timely packet delivery. The ability of token bucket filtering to manage and restrict bandwidth for non-real-time traffic, hence lowering network congestion, has been the subject of other studies.

The evaluation of various QoS techniques for certain applications or environments, such as mobile networks, cloud computing, or IoT systems, has also attracted interest. Researchers have assessed, for instance, how QoS might be adapted for mobile networks to enhance the user experience by effectively managing limited bandwidth and addressing mobility issues.

Our project differs from previous works in a number of ways. First off, our project conducts a comparative study of various QoS techniques (priority queueing and token bucket filtering) under the same simulated network conditions, in contrast to the majority of the existing literature, which concentrates on individual QoS techniques. This comparison approach makes it possible to gain a deeper understanding of how various QoS techniques interact with one another and can be combined to improve and optimize network performance.

Second, our project evaluates the performance of QoS techniques for a variety of traffic kinds, including web browsing and bulk data transfer. This contrasts with a large number of existing studies, which frequently concentrate on specific uses or environments. Our project offers insights that are more broadly relevant to various network environments since it takes a broader variety of traffic types into account.

Finally, our project not only evaluates the performance of QoS techniques but also compares this with the performance of a network without any QoS optimization. This offers useful insights into the practical advantages and trade-offs of implementing QoS and serves as a baseline against which the effectiveness of various QoS techniques can be measured.

In terms of background, it’s important to understand that Quality of Service (QoS) in networking refers to the performance level of service provided by the network to applications, users, or data flows. Network resource management, traffic prioritization, and ensuring that network performance adheres to standards are all accomplished via QoS techniques. In order to provide reliable and efficient network performance across an extensive variety of applications, several QoS techniques are designed to deal with various challenges such as latency, constraints on bandwidth, packet loss, and jitter.

**6. Design:**

Our project makes use of a simulated network environment to perform an analysis of the impact of two QoS techniques: priority queueing and token bucket filtering. The design includes setting up a virtual network, generating various traffic types, applying QoS techniques, and analyzing the performance metrics.

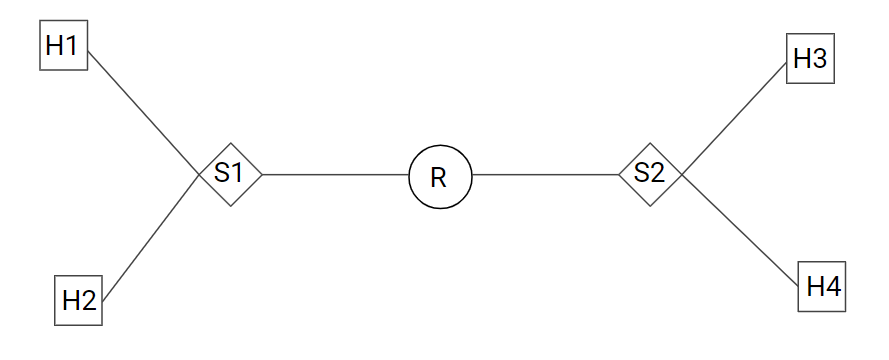


Figure 1: Network Topology

The network environment is simulated using Mininet, which allows us to create a custom network topology with multiple hosts and switches. We create a topology with four hosts and two switches where each switch is connected to two hosts and both the switches are connected to a single router. This simple topology is chosen to ensure a controlled environment for our experiments. Traffic control commands are employed to apply QoS techniques to the hosts. To simulate real-world network traffic, we use 'iperf' to generate bulk data transfer traffic and 'curl' or 'wget' to simulate web browsing. Wireshark is used to capture packets across the network for in-depth analysis. Data is extracted from the Wireshark captures, and performance metrics such as latency, packet loss, and throughput are calculated and compared.

The primary question we are looking into is how token bucket filtering and priority queueing impact latency, packet loss, and throughput in a network environment that is simulated with different traffic kinds. According to our hypothesis, priority queueing will be more successful in reducing latency and packet loss for time-sensitive traffic. On the other hand, we believe that token bucket filtering will perform better for bulk data transfers since it effectively manages bandwidth, hence lowering network congestion and packet loss. We also question how the network performs when QoS techniques aren't applied and how this baseline performance compares to the network when QoS techniques are already applied.

When the network is handling time-sensitive traffic, such as VoIP, and it's crucial to ensure minimal delay, priority queueing should perform the best. When there is a need for efficient bandwidth management and bulk data transfer, token bucket filtering is anticipated to perform well. In cases when non-prioritized traffic is resource-starved, priority queueing may not function well. Since token bucket filtering doesn't inherently prioritize such traffic, it could not be as efficient for scenarios requiring real-time communication. With this approach, the project is certain to be able to examine the effects of the QoS techniques under consideration and provide a baseline against which to compare their performance in various traffic conditions.

**7. Implementation:**

The implementation phase of our project involved setting up the simulated network environment, applying QoS techniques, generating traffic, and capturing data for analysis on Wireshark.

Setting Up Simulated Network:

We utilized Mininet to create a custom network topology consisting of four Linux-based hosts and two switches. Mininet's Python API was utilized to script the setup process, ensuring repeatability. Using our topology script(python topo.py), we started the network.

Applying QoS Techniques:

We employed Linux traffic control (tc) commands to apply QoS techniques to the virtual hosts in Mininet.

We implemented Priority Queueing using the `prio` qdisc, and Token Bucket Filtering using the `tbf` qdisc. For example,

h1 tc qdisc add dev h1-eth0 root handle 1: prio

h1 tc qdisc add dev h1-eth0 handle 1:0 tbf rate 1mbit burst 32kbit latency 400ms

Generating Traffic:

We used `iperf` to generate bulk data transfer traffic between the virtual hosts. On one host, we started an iperf server(h1 iperf -s &), while on another, we started an iperf client(h2 iperf -c h1 &). For the simulation of web browsing, we used `wget` to fetch web pages.

Capturing and Analyzing Data:

We utilized Wireshark to capture network traffic for analysis. To capture the traffic on each host, we used the following command:

tcpdump(tcpdump -i h1-eth0 -w h1.pcap). This generates a .pcap file that we can later examine using Wireshark. Using Wireshark's filters and statistics features, we extracted relevant data for calculating latency, packet loss, and throughput.

Libraries and Languages:

1. We used Python for scripting the custom network topology of the Mininet network.

2. Linux command-line tools such as `tc`, `iperf`, `curl`, and/or `wget` were utilized for configuring QoS, generating traffic, and fetching the page.

3. Wireshark was utilized for capturing and analyzing network traffic.

Challenges and Solutions:

1. Accurately determining latency was one challenge. This was resolved by computing the difference between the packet timestamps using Wireshark's timestamp feature.

2. Thorough implementation and configuration in Mininet were necessary to manage various traffic types and it was made sure they don't interfere with one another.

3. Ensuring the accuracy of results required multiple iterations of testing and fine-tuning of configurations.

Lines of Code:

The implementation consisted of approximately 100 lines of Python code for the custom network topology script and additional multiple shell scripts for traffic control configurations and traffic generation.

This implementation strategy made sure that the impact of various QoS techniques on network performance was assessed and compared in a systematic manner.

**8. Results & Analysis:**

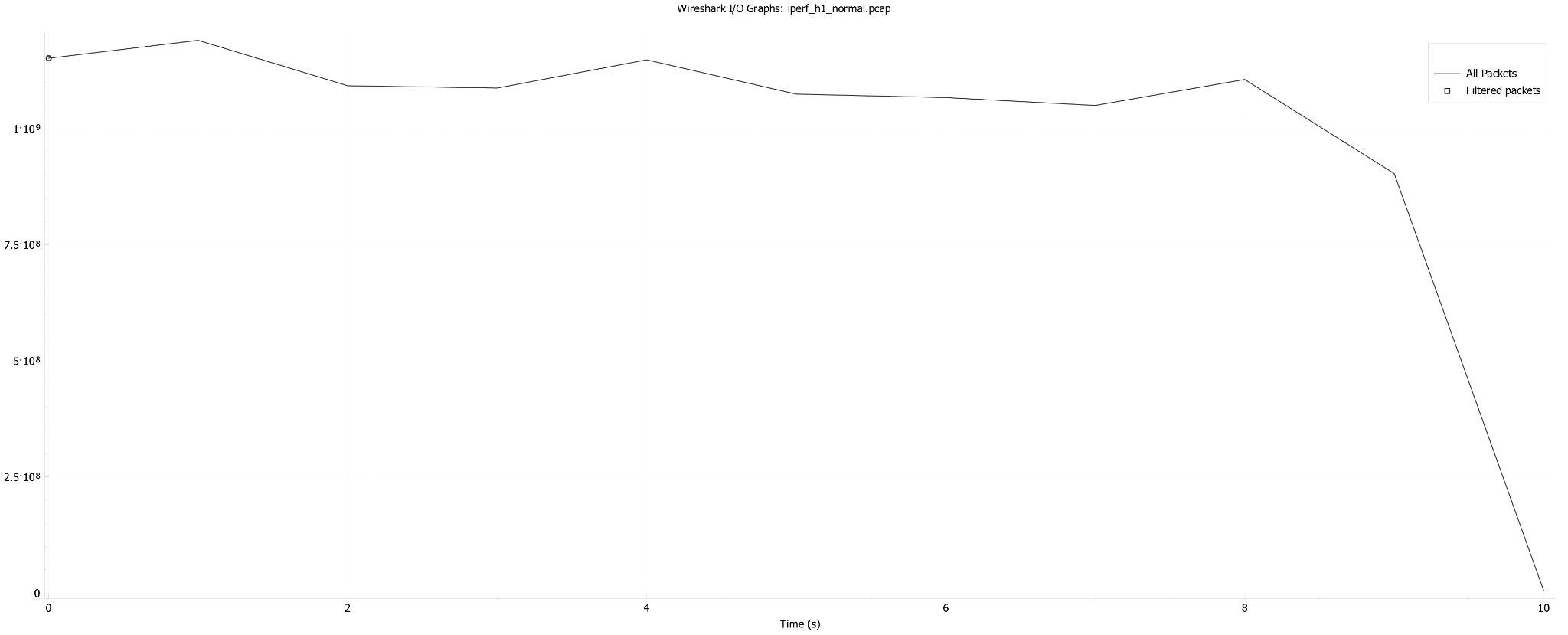


Figure 2: iperf h1 without QoS

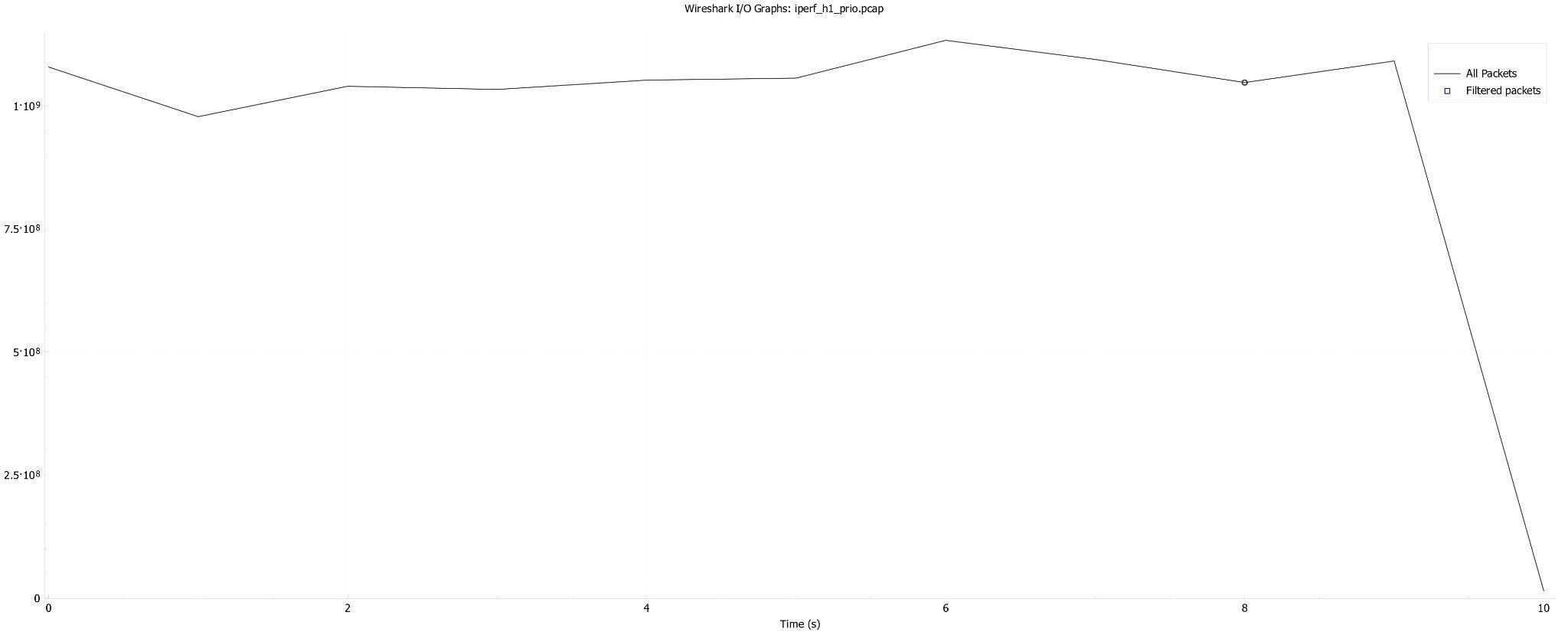


Figure 3: iperf h1 prio

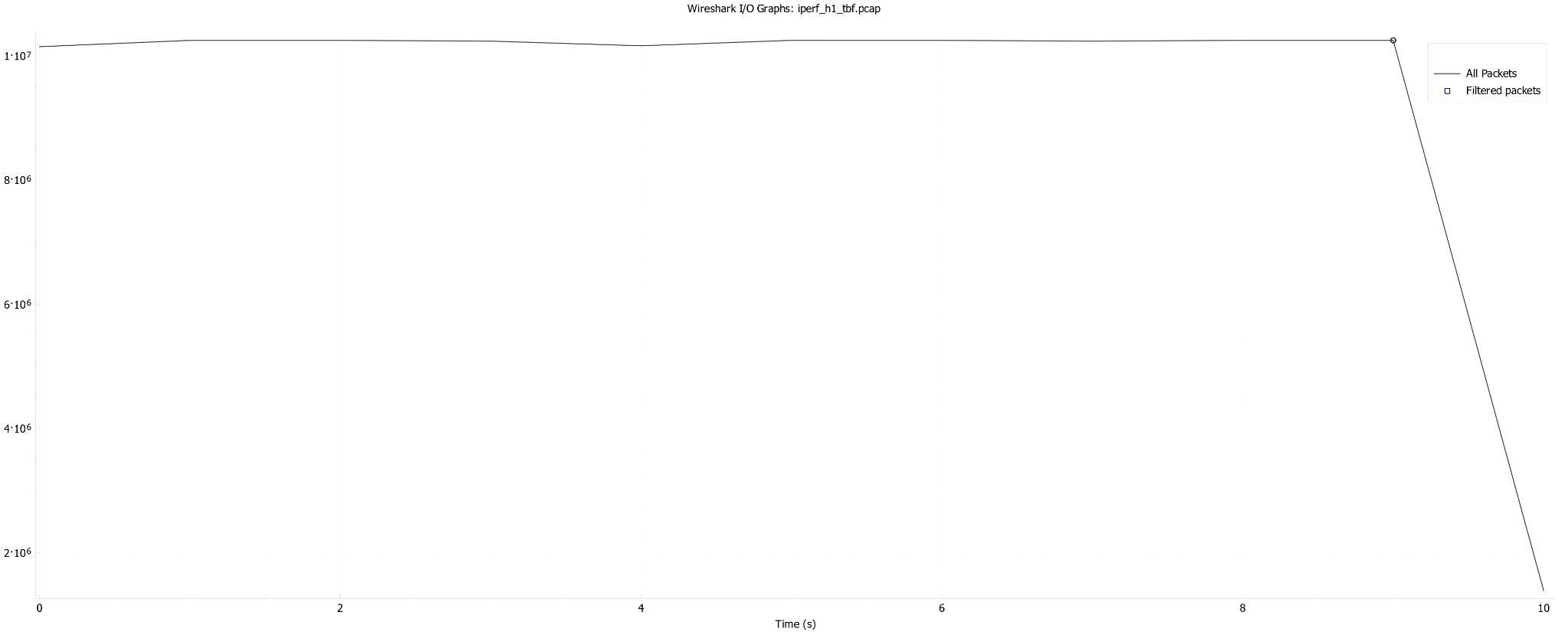


Figure 4: iperf h1 tbf

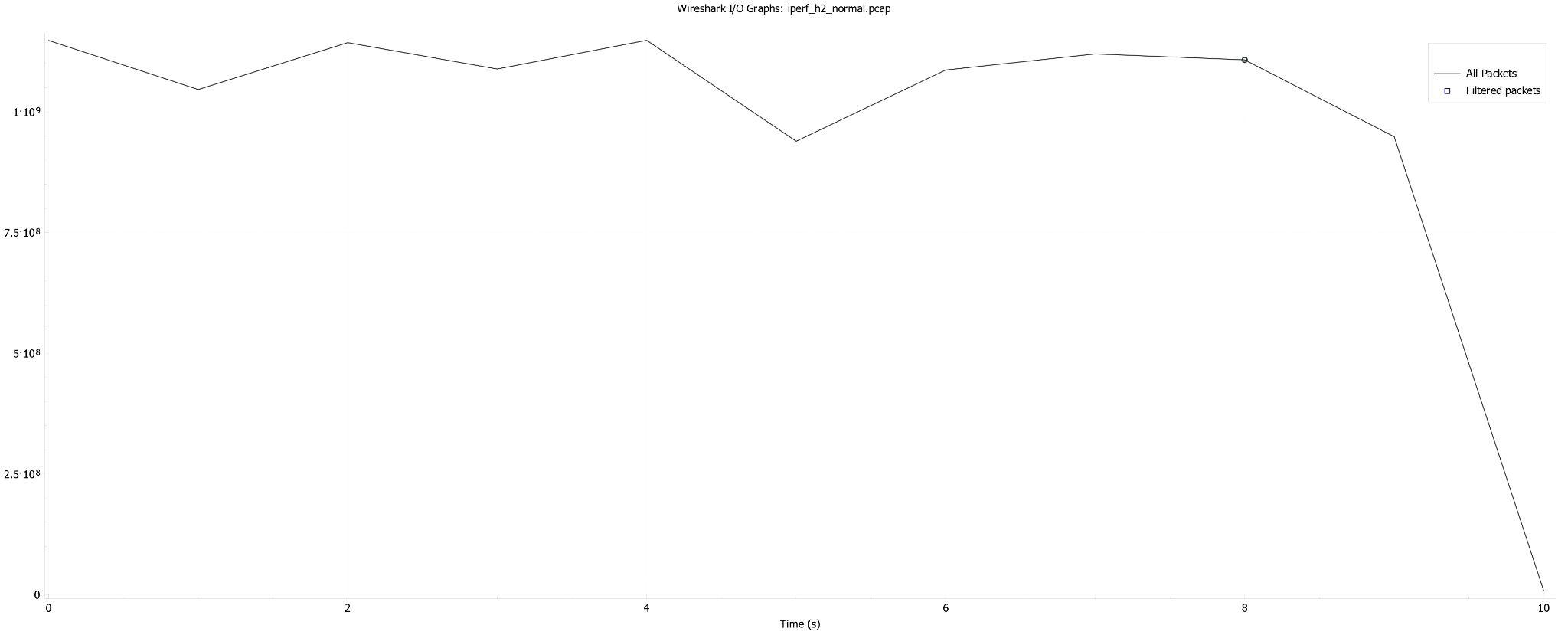


Figure 5: iperf h2 without QoS

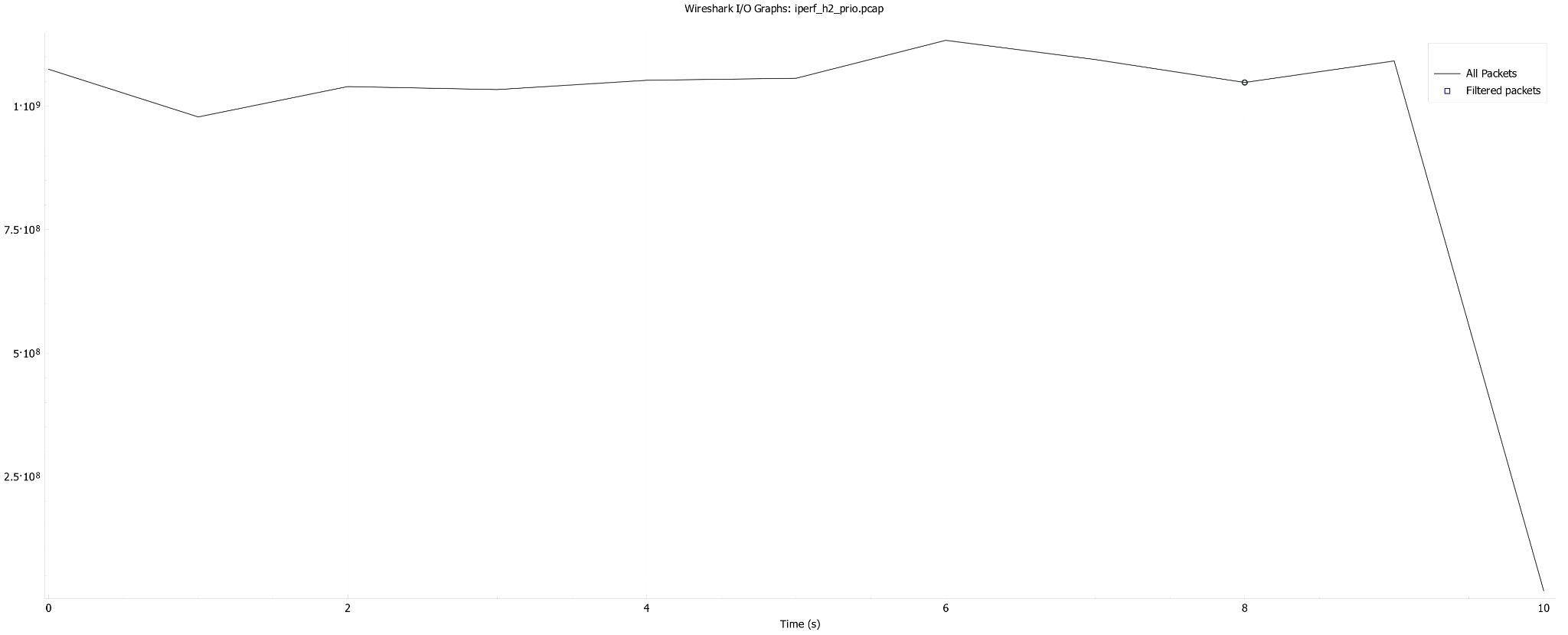


Figure 6: iperf h2 prio

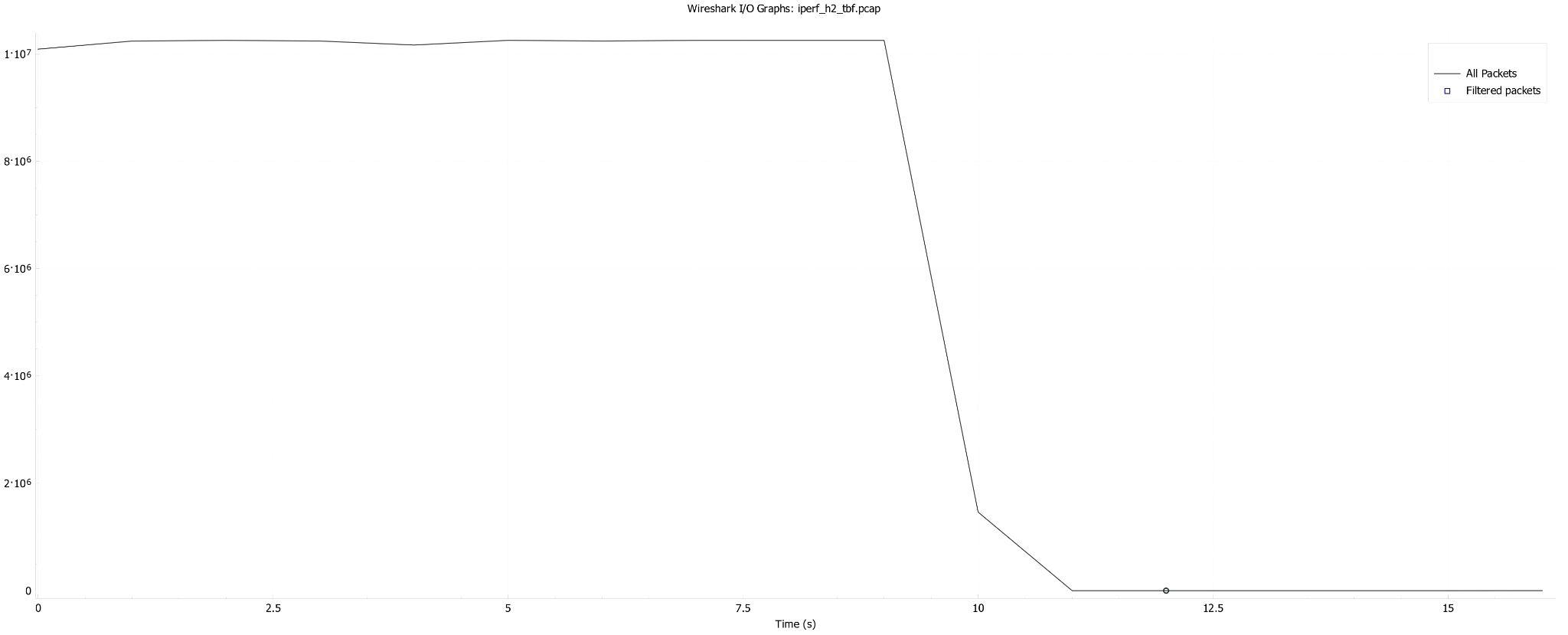


Figure 7: iperf h2 tbf

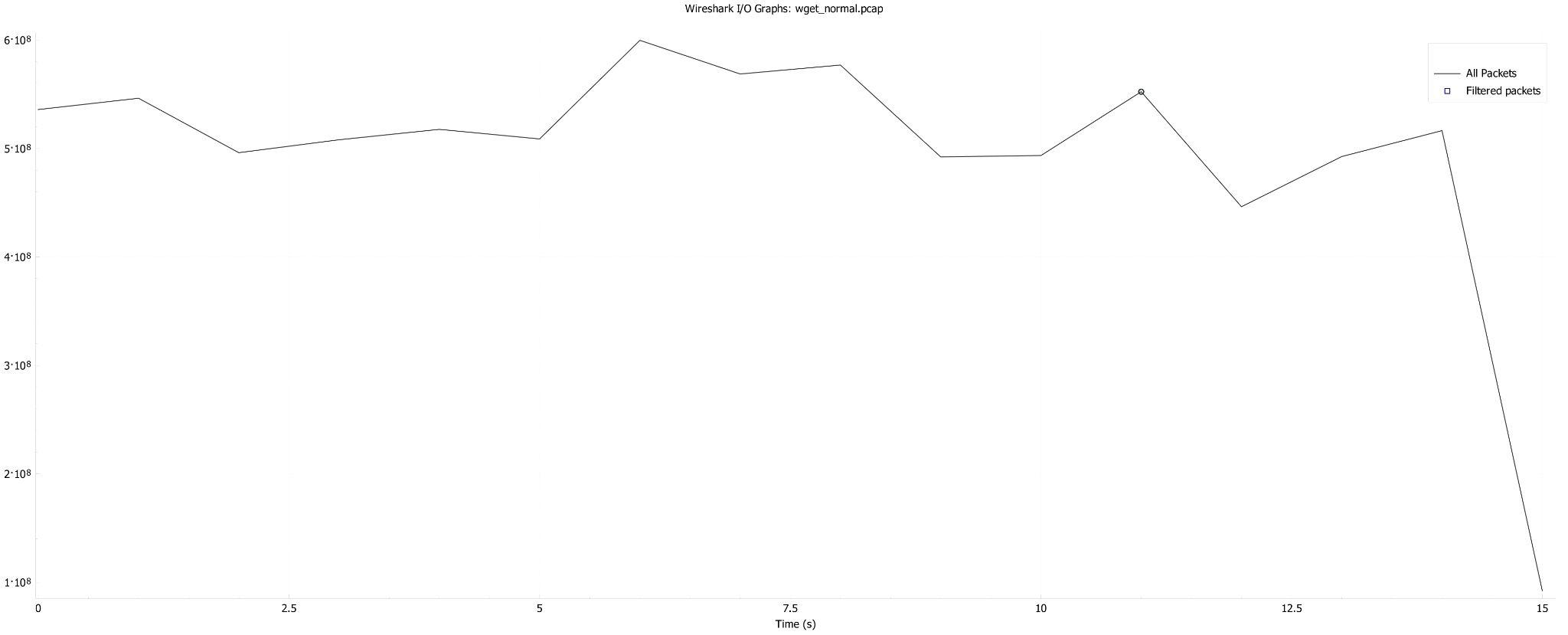


Figure 8: wget without QoS

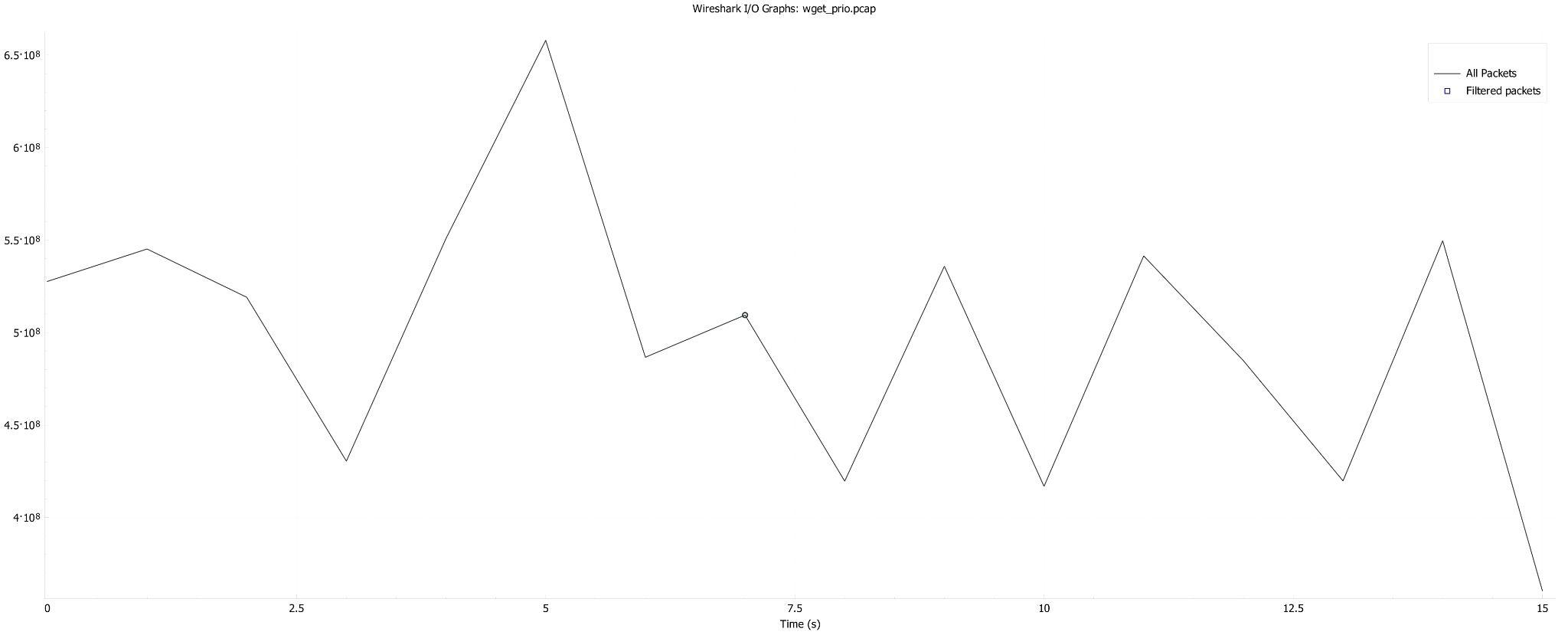


Figure 9: wget prio

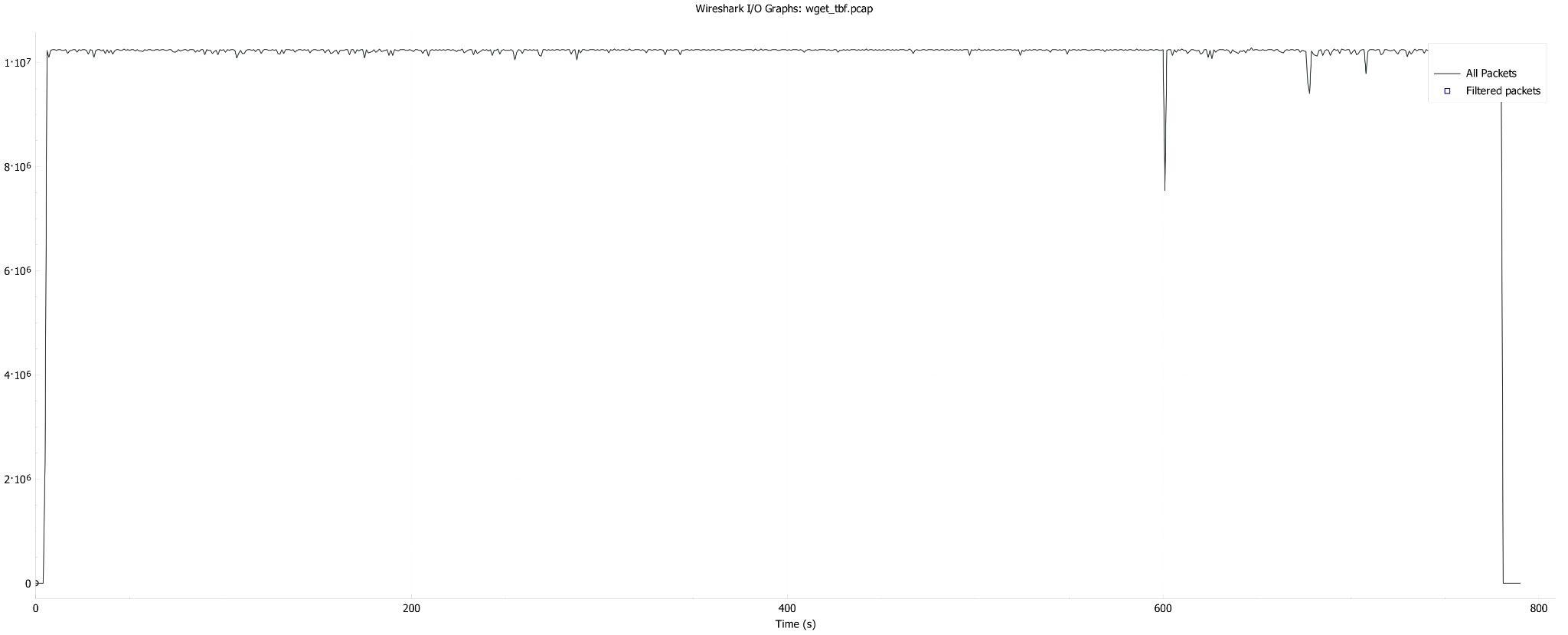


Figure 10: wget tbf

1. Latency:

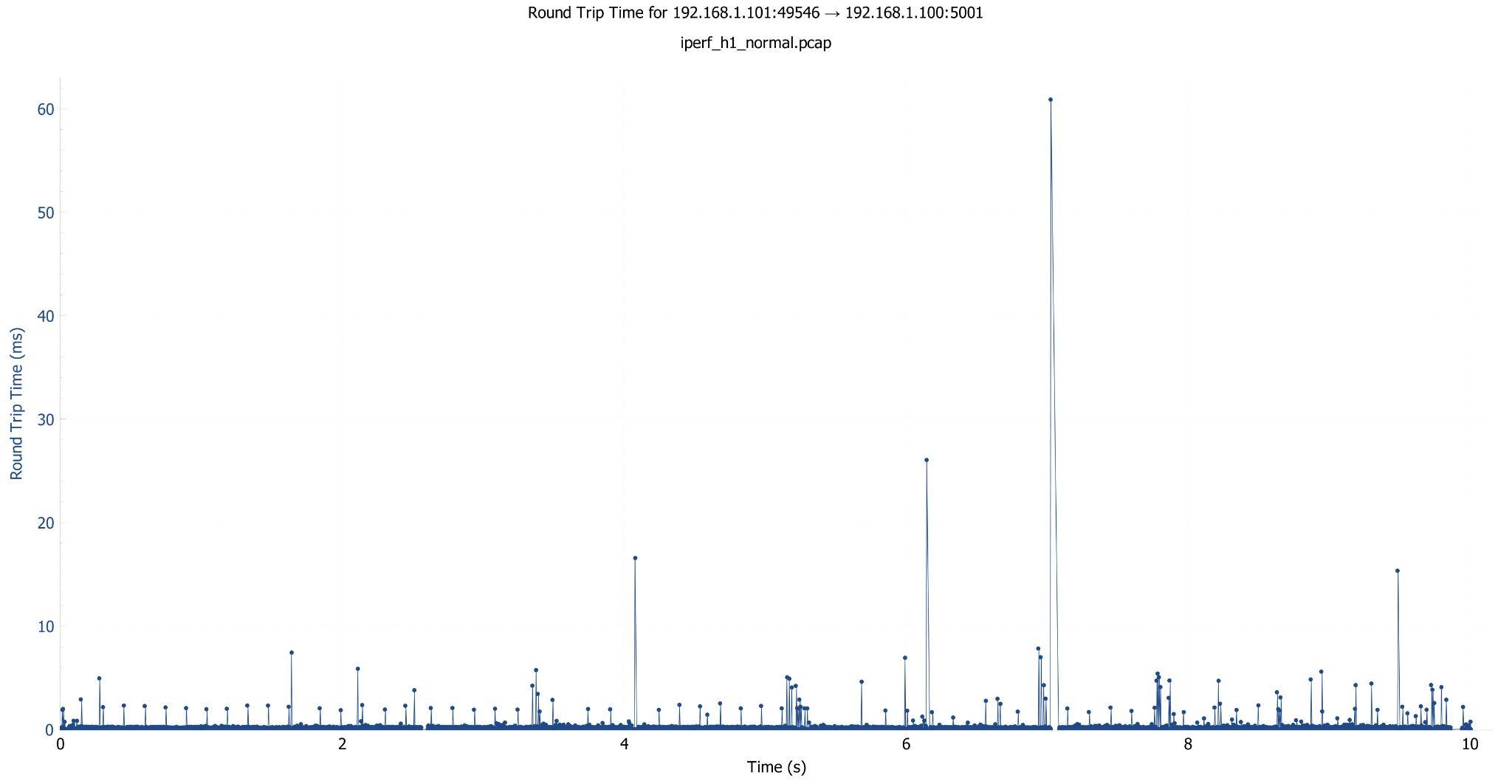


Figure 11: Latency: iperf without QoS

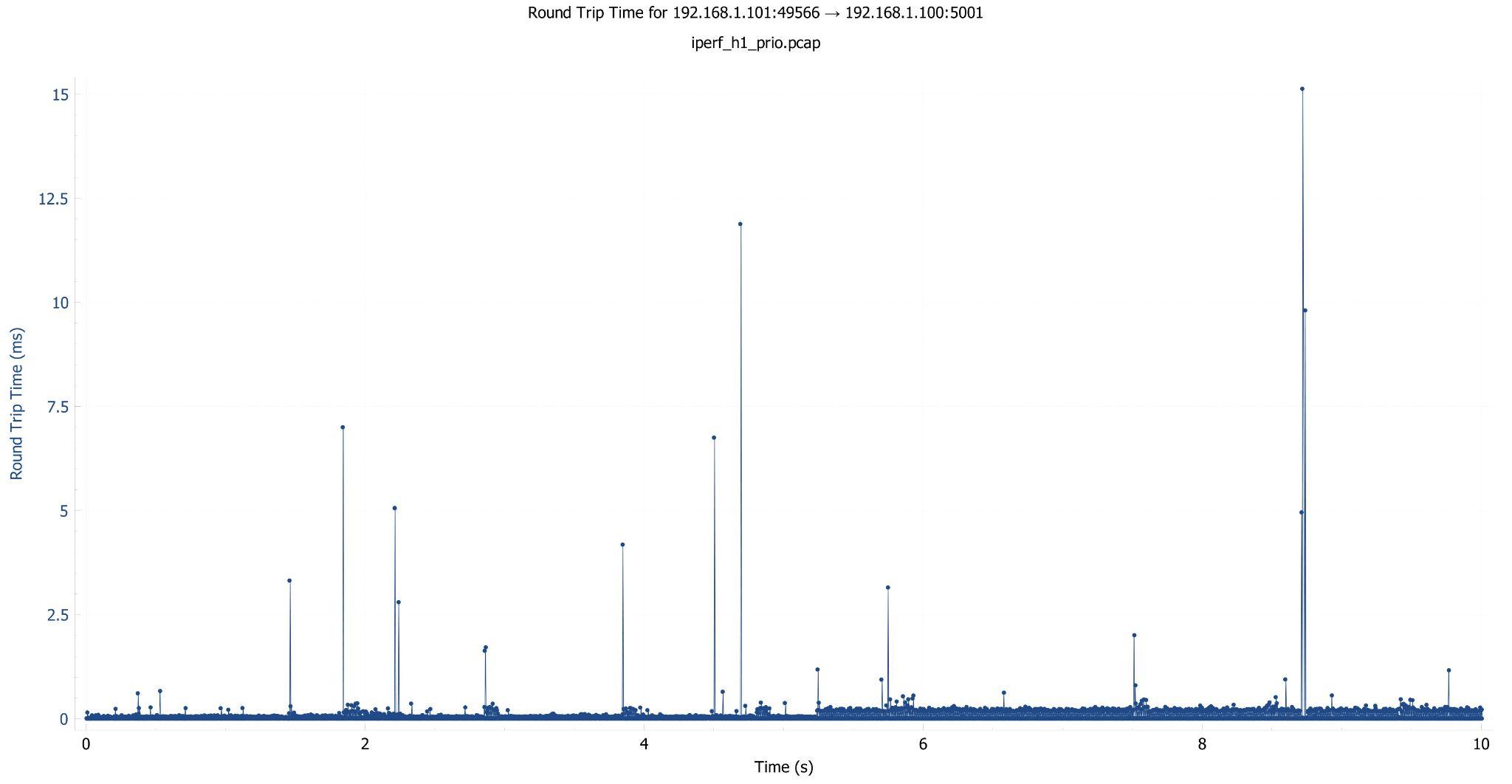


Figure 12: Latency: iperf prio

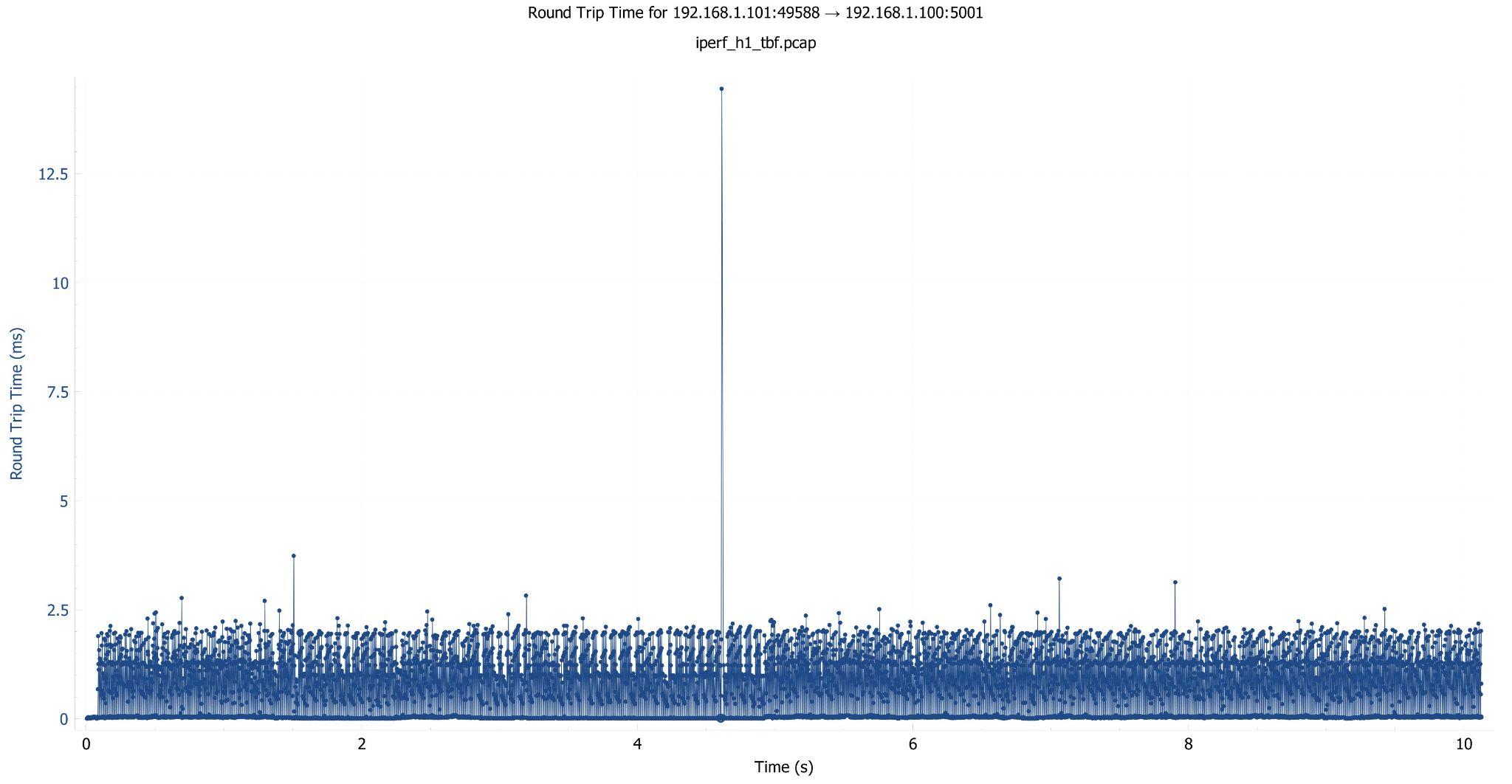


Figure 13: Latency: iperf tbf

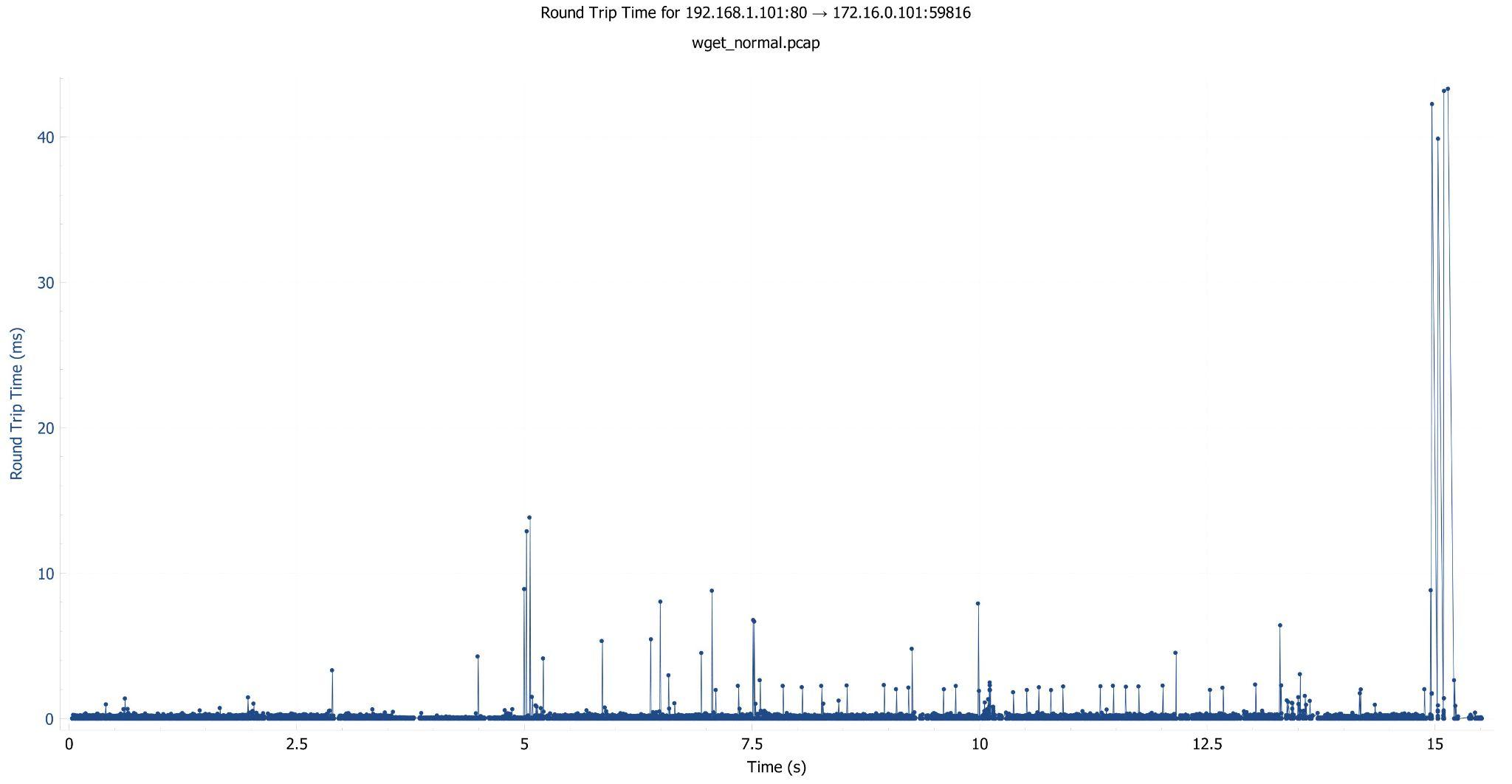


Figure 14: Latency: wget without QoS

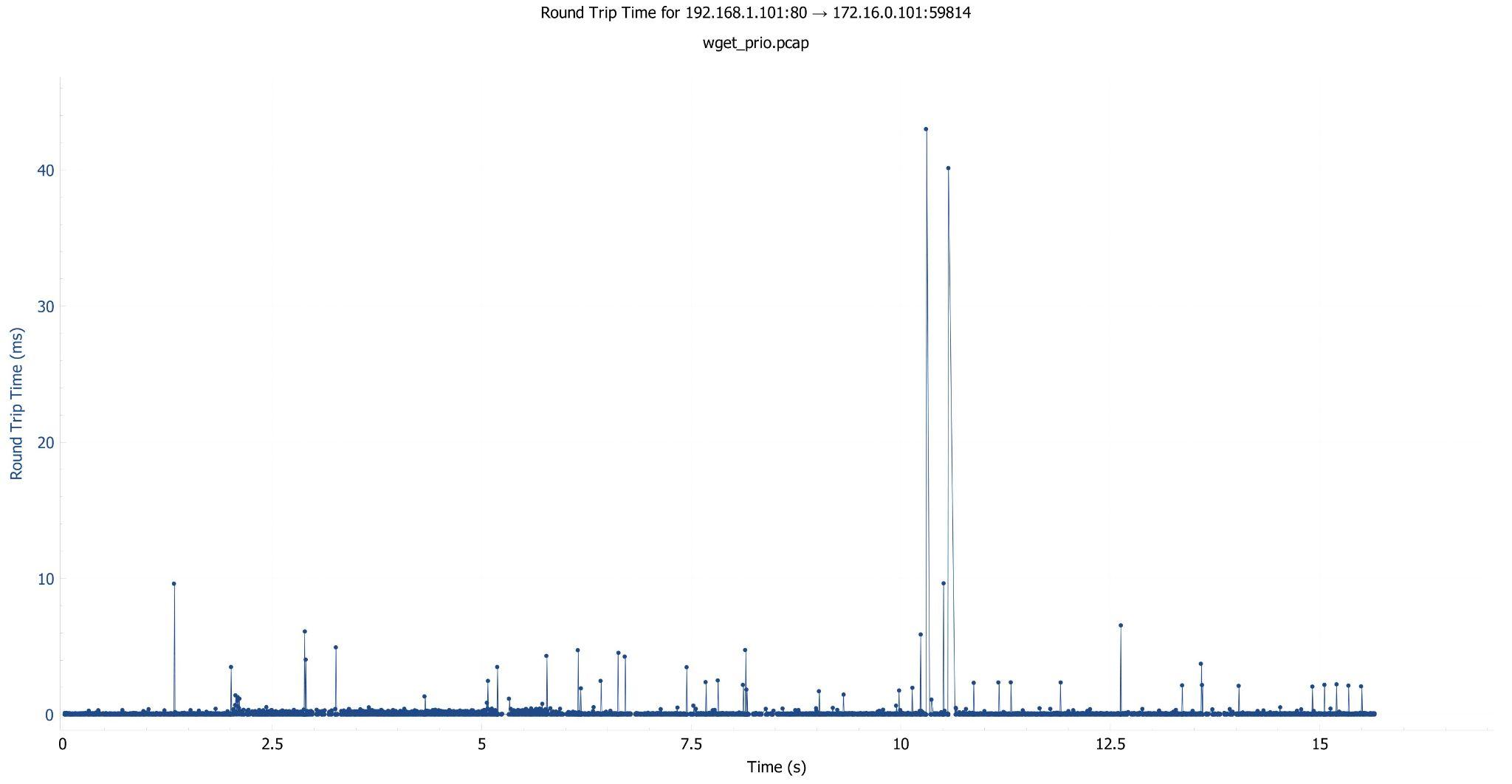
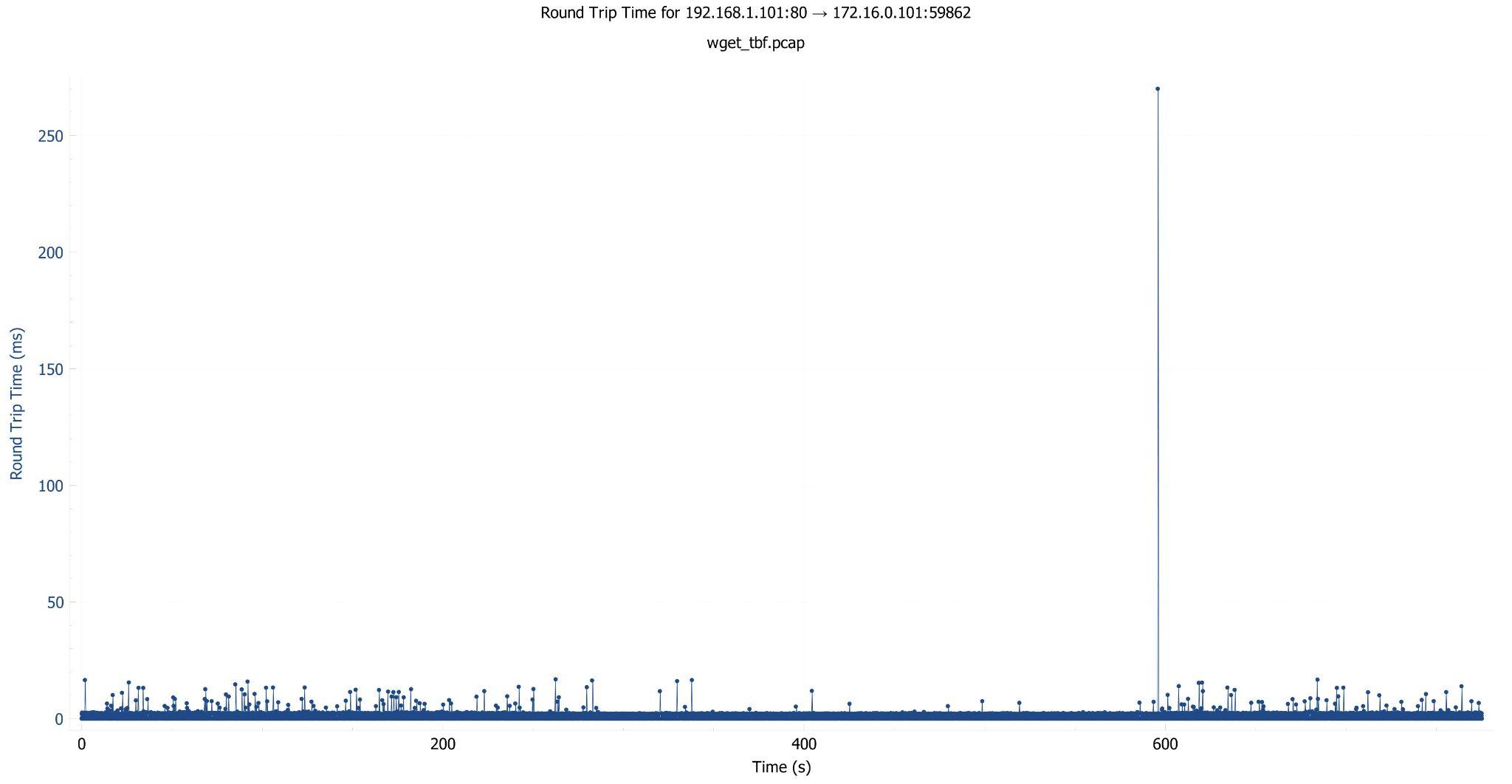


Figure 15: Latency: wget prio

Figure 16: Latency: wget tbf

As you can observe in the figures, for iperf, priority queueing has the lowest latency. Token bucket filtering has a slightly higher latency, while the baseline has the highest latency. As you can observe in the figures, for wget, prio significantly reduces latency compared to tbf or without QoS.

The reader learns that priority queueing is effective in reducing latency, which is crucial for real-time applications and time-sensitive traffic. It confirms the hypothesis that priority queueing performs well for time-sensitive traffic.

This answers the question about the impact of priority queueing on latency and supports the notion that it's beneficial for scenarios where minimal delay is critical. This also relates back to the question regarding how different QoS techniques affect latency for different types of traffic.

2. Packet Loss:

As you can observe in the figures, for iperf, packet loss is significantly lower with tbf as compared to prio. The highest packet loss could be observed without QoS. As you can observe in the figures, for wget, packet loss varies significantly between different QoS techniques and is quite lower with tbf.

The reader learns that token bucket filtering is effective in controlling the data rate, thereby reducing congestion and packet loss and therefore ensuring data integrity. This is particularly important for non-real-time applications where ensuring the delivery of all packets is more critical than timely delivery.

This answers the question regarding how QoS techniques can be used to manage bandwidth and packet reliability depending on traffic types.

3. Throughput:

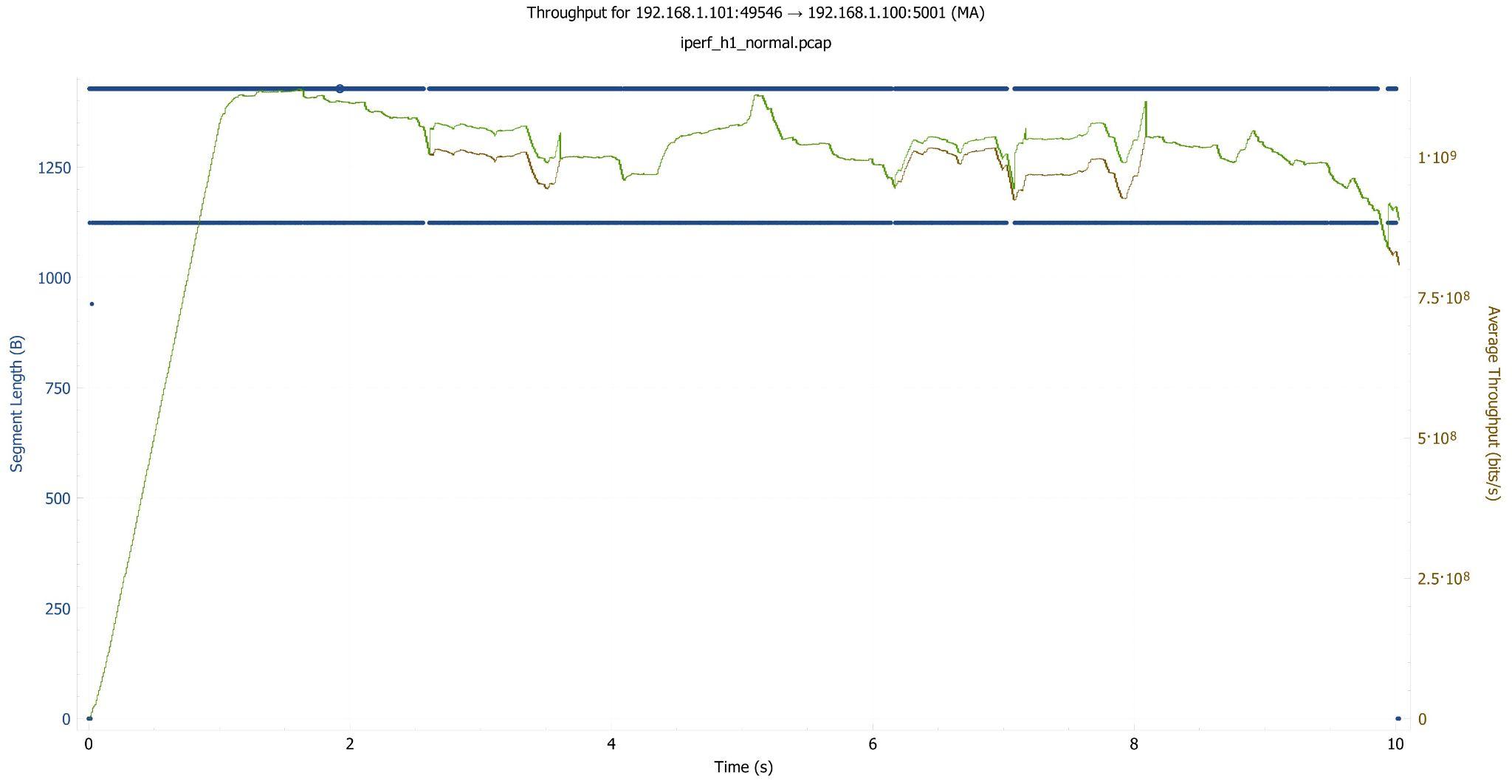


Figure 17: Throughput: iperf without QoS

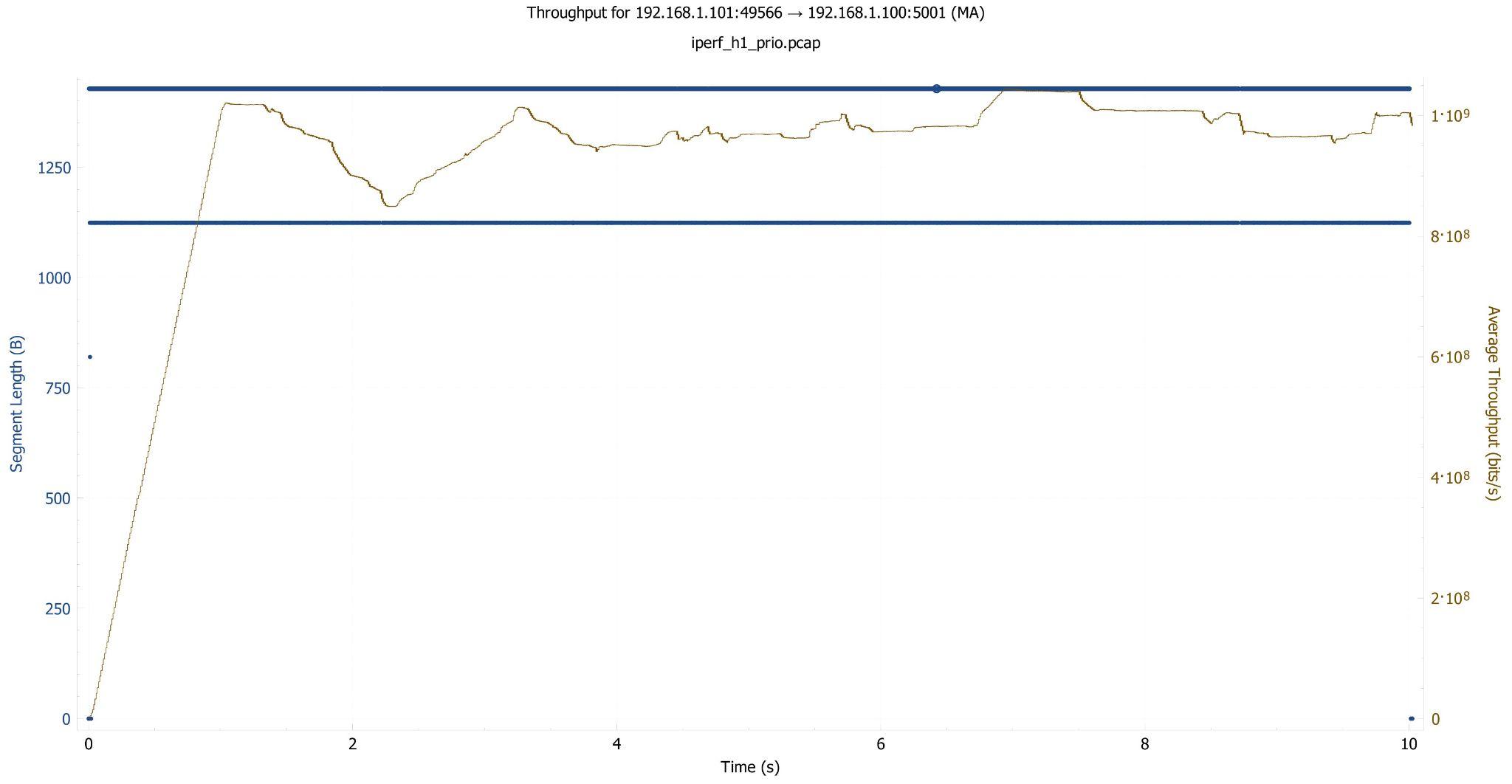


Figure 18: Throughput: iperf prio

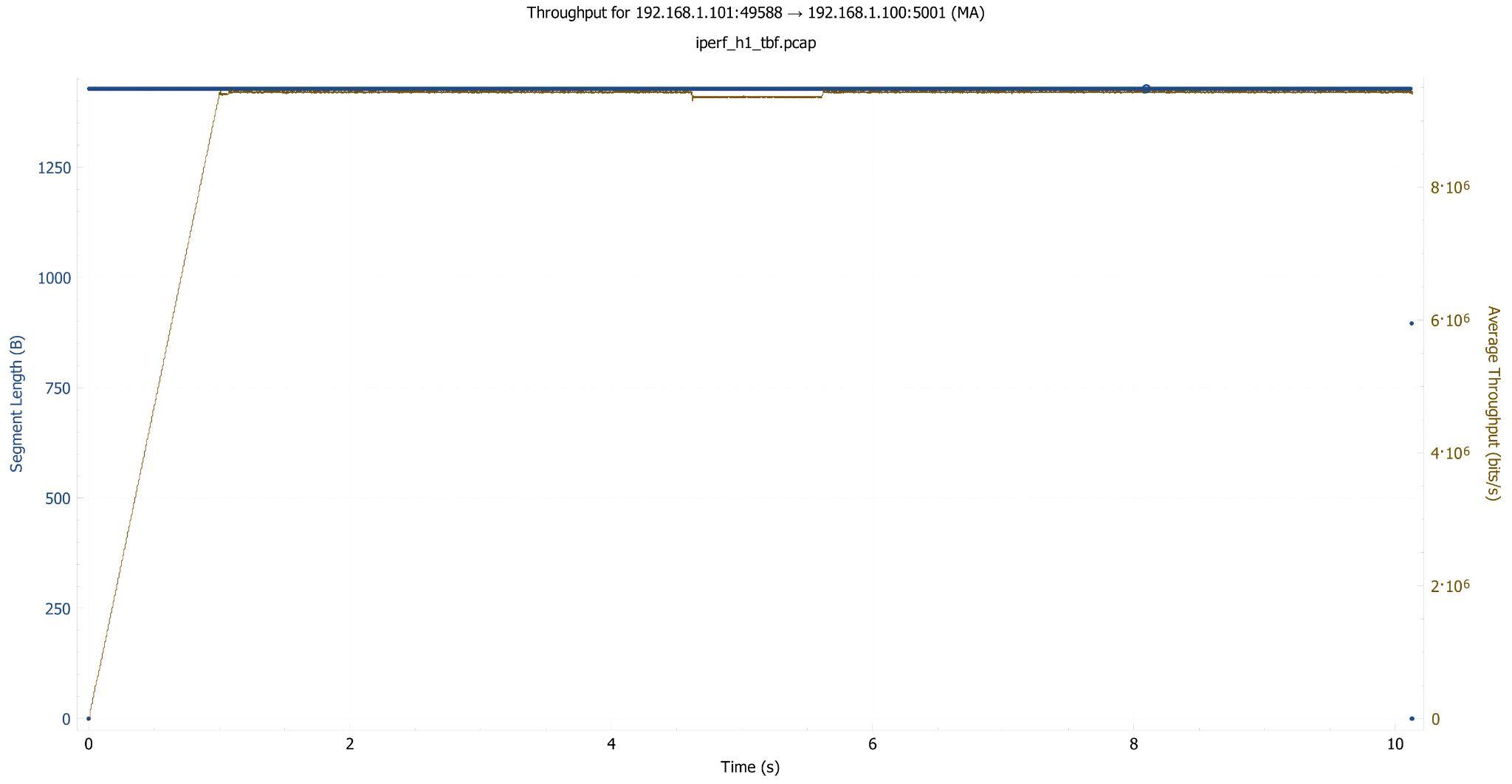


Figure 19: Throughput: iperf tbf

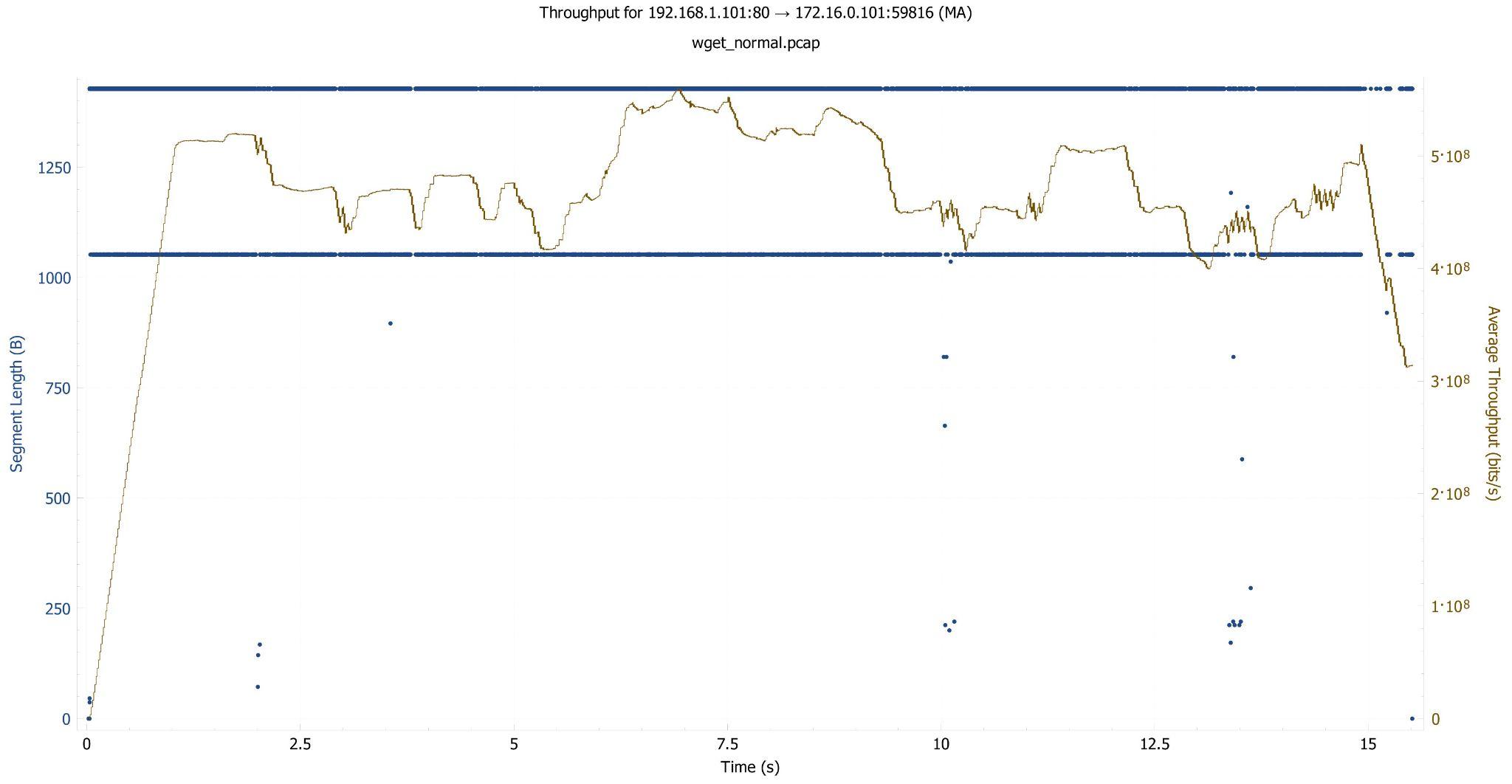


Figure 20: Throughput: wget without QoS

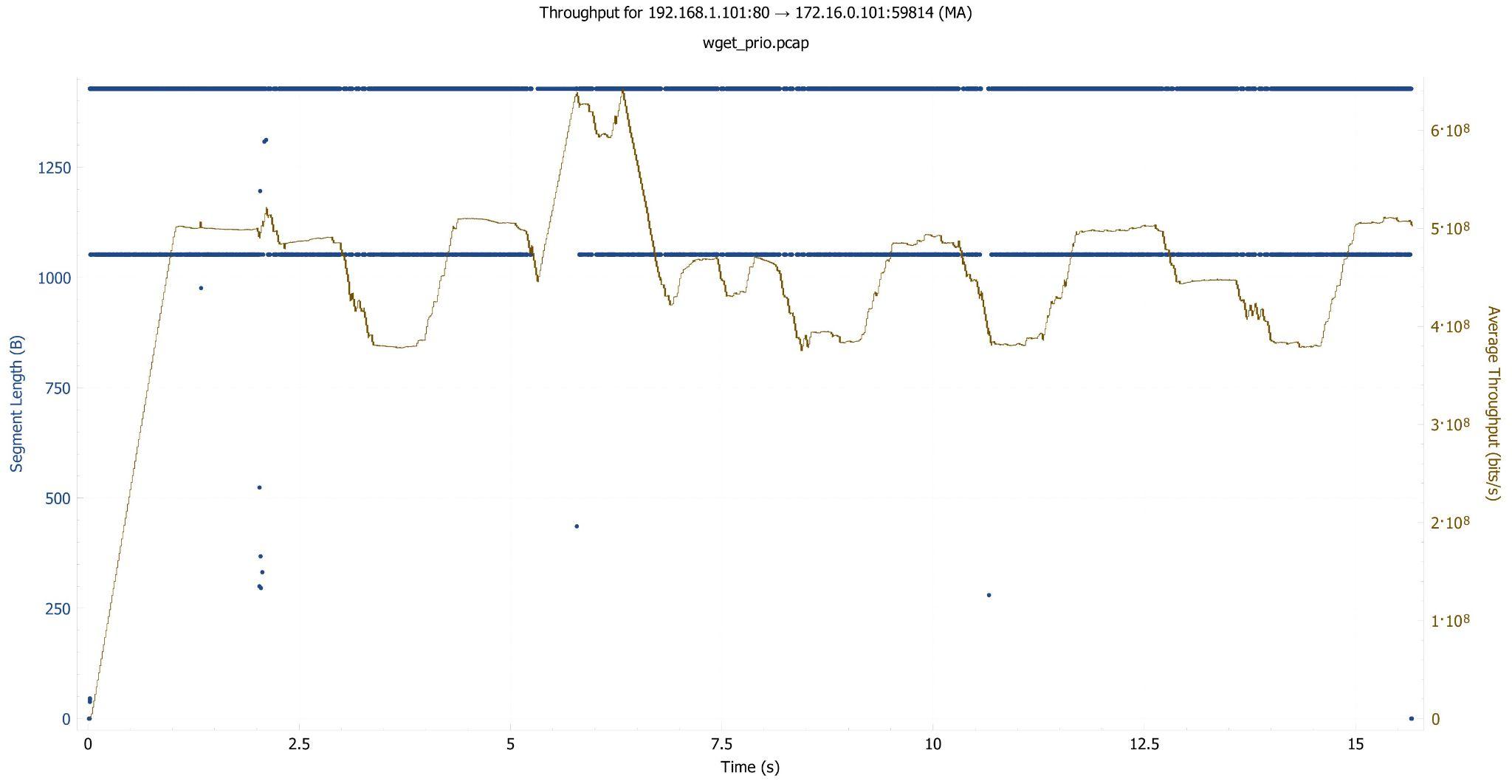


Figure 21: Throughput: wget prio

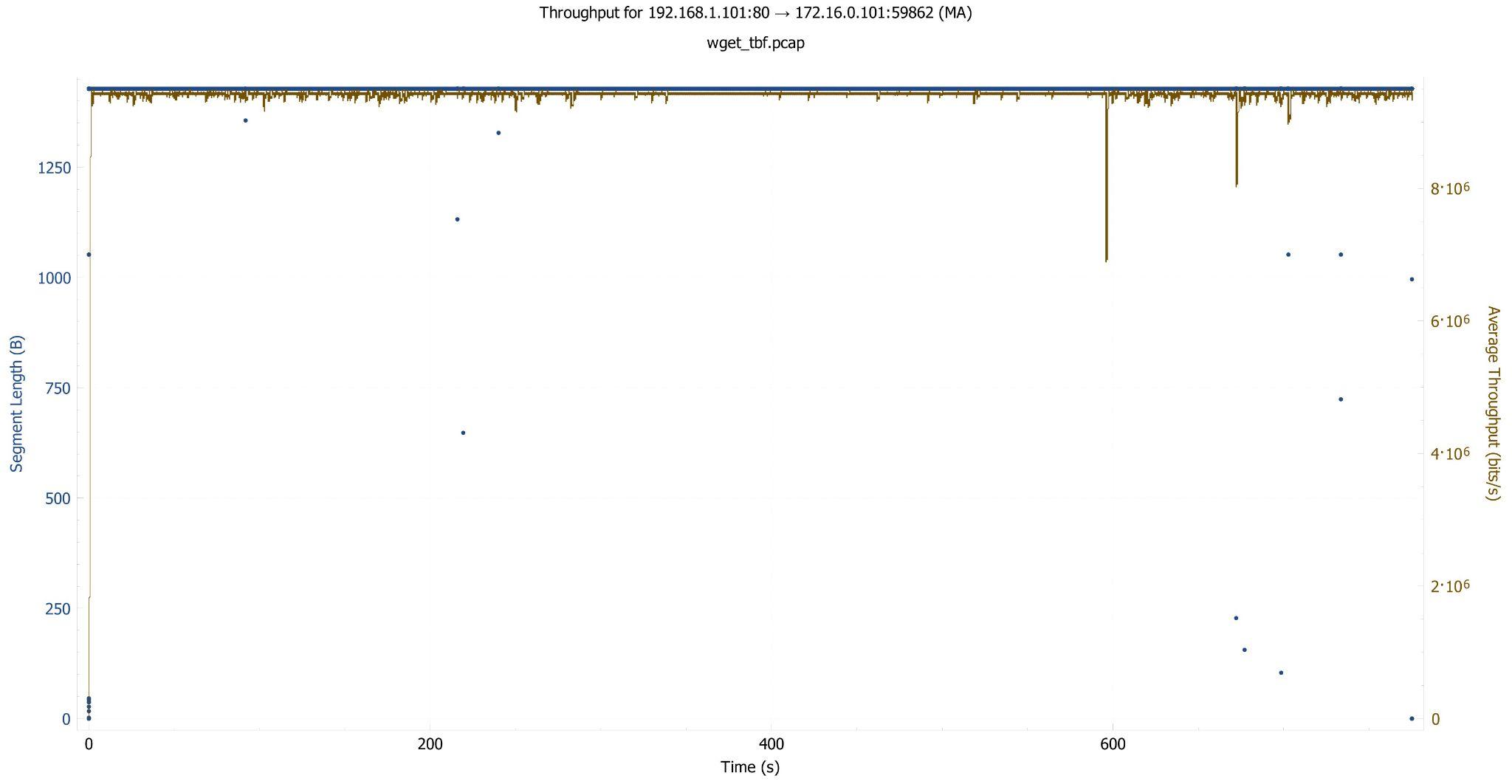


Figure 22: Throughput: wget tbf

For iperf, throughput is the highest when using tbf, followed closely by prio, and the lowest without QoS. For wget, throughput is not significantly affected by the QoS technique used but shows a slight increase with prio.

The reader learns that applying QoS techniques can effectively utilize network resources to maximize data transfer rates. The reader also learns that tbf is particularly effective in increasing throughput for bulk data transfers by efficiently managing network resources, whereas prio can slightly improve throughput for web browsing traffic.

This addresses the question of how QoS techniques influence throughput as well as optimize network resources to maximize data transfer rates and confirms the hypothesis that token bucket filtering is well-suited for scenarios that involve bulk data transfers.

These results offer insightful information about how priority queueing and token bucket filtering perform in various network scenarios. Token bucket filtering is adept at handling bandwidth for bulk data transfers, minimizing packet loss, while priority queueing is more advantageous for time-sensitive traffic, providing minimal latency. Compared to a network without QoS optimizations, both methods increase overall throughput. These results are essential for network managers and engineers in order to make knowledgeable choices about QoS configurations based on the particular requirements of their network traffic.

**9. Lessons Learned:**

We learned a great deal about the practical use of QoS techniques and how they affect network performance. We learned the significance of thorough design and tuning for attaining the desired outcomes. On the performance metrics, even slight configuration modifications could have a big effect. We came to understand the advantages of a systematic approach to data collection and analysis. The accuracy and consistency of the data collection were crucial for the validity of the results we obtained. We also learned that there is no one-size-fits-all approach to network performance optimization. Customized setups and configurations are necessary for optimal performance in various scenarios and traffic types.

One obstacle was ensuring accurate and consistent data capture and analysis. This was overcome by carefully carrying out the entire setup, taking a tcpdump, and examining the data and outcomes on Wireshark.

Another challenge was dealing with the complexity of the traffic control commands and configurations. In order to grasp the implications of different commands and configurations, we thoroughly researched and tested them.

Interpreting the data and relating it to the underlying network behavior presented further challenges for us. So, we better understood the findings by plotting graphs and discussing potential interpretations.

As a team of two, Manoj Nagarajappa and Puneet Singhania, we effectively divided the workload to make optimal use of our strengths and ensure a collaborative effort.

1. Puneet focused on: Setting up the Mininet environment and coding the network topology. Implementing the priority queueing technique and configuring traffic control settings. Data analysis and interpretation, particularly focusing on latency metrics.

2. Manoj focused on: Implementing the token bucket filtering technique. Generating different traffic types using different traffic generation tools. Data analysis and interpretation, with a focus on packet loss and throughput metrics.

We held regular meetings to discuss our progress, challenges, and findings. This ensured that both members were aware of the overall project status and could provide input and support as needed. Additionally, we collaboratively created the presentation and wrote the final report, combining our insights and interpretations into a cohesive document.