# Puffer Data Analysis

**Tyler Niiyama**
New York University
tn2208@nyu.edu

**Wenbo Bao**
New York University
wb2128@nyu.edu

## 1 Introduction

Video streaming applications have revolutionized the internet landscape, presently accounting for nearly three-quarters of all internet traffic. One pivotal mechanism within this technology is Adaptive Bitrate Selection (ABR) algorithms. ABR strategically determines the quality or compression level for each fragment of video being transmitted, striking a balance between two primary metrics: video quality and stall time. While a high compression level ensures seamless transmission, high video quality may risk playback interruptions due to larger chunk sizes taking longer to reach the client.

Modern ABR algorithms employ statistical and machine learning techniques, considering a multitude of input signals to cater to a wide spectrum of client preferences. These algorithms take into account a variety of factors such as recent throughput, client-side buffer occupancy, and delay. However, an important realization is that the effectiveness of these learned algorithms is significantly influenced by the data environments they were trained on.

In addressing this issue, the authors of the Puffer study designed a public video-streaming website, Puffer, which randomly allocates each session to one of a set of ABR algorithms (Yan et al., 2020). The original Puffer experiment ran from January 2019 to October 2019, collecting 9 months of data they used to compare the performance of various ABR algorithms. The Puffer experiment has continued to run as of 2023, publishing daily plots of video quality vs. stall ratio. Analysis of the data collected in 2019 yielded several insightful observations:

- In practical scenarios, sophisticated algorithms based on control theory or reinforcement learning did not surpass the performance of simpler buffer-based controls.
- Relatively small sample sizes lead to substantial statistical margins of error in measuring algorithm performance.
- It was possible to robustly outperform existing schemes by amalgamating classical control with a machine learning predictor trained on real-time data.

Our research project aims to build on the foundation laid by the Puffer study, focusing on the data generated post-publication. We seek to address two key questions:

- How do different ABR algorithms compare in analyses not covered by the original Puffer paper?
- Given that the findings of the Puffer study predate the pandemic, we are interested in examining whether user behavior has been altered due to the pandemic, and the consequent impact of this change on the performance of different ABR algorithms.

## 2 Background

There has been significant prior research on the design of ABR algorithms, five of which are tested in the original Puffer experiment. BBA (Huang et al., 2014) is a buffer-based algorithm that uses the current capacity of the playback buffer to adjust video bitrate encoding, using capacity estimation only at startup. MPC and RobustMPC (Yin et al., 2015) are control theory approaches that use a throughput predictor and a predictive model that plans which sequence of video chunks to fetch at what rate. Pensieve (Mao et al., 2017) uses a neural network trained with reinforcement learning to make decisions on how to send chunks. Finally, Puffer's ABR algorithm, Fugu, uses a control theory approach similar to MPC, except instead of a throughput predictor, it uses a neural network to produce a probability distribution of transmission times.

| fields | sent | acked | buffer |
|---|---|---|---|
| time | ✓ | ✓ | ✓ |
| session_id | ✓ | ✓ | ✓ |
| expt_id | ✓ | ✓ | ✓ |
| video_ts | ✓ | ✓ | ✗ |
| size | ✓ | ✗ | ✗ |
| ssim_index | ✓ | ✗ | ✗ |
| cwnd | ✓ | ✗ | ✗ |
| in_flight | ✓ | ✗ | ✗ |
| rtt | ✓ | ✗ | ✗ |
| delivery_rate | ✓ | ✗ | ✗ |
| buffer | ✓ | ✓ | ✗ |
| cum_rebuf | ✓ | ✓ | ✓ |
| event | ✗ | ✗ | ✓ |

Table 1: important fields in raw CSVs

## 3 Data Description

We made use of the results released daily to the Stanford Puffer Experiment Results page[1], which contains all data collected since the experiment began in January 2019. The following are the relevant files used in our analysis.

### 3.1 Raw Data Files

There primary files released on the buffer websites are `video_sent_X.csv`, `video_acked_X.csv`, `client_buffer_X.csv`, `ssim_X.csv`, and `video_size_X.csv` where X represents the day when the data was collected. Our analysis are primarily based on the first three files, all important fields are displayed in Table 1. In `video_sent_X.csv`, each line represents a data point collected when the Puffer server sent a video chunk to a client. In `video_acked_X.csv`, Each line represents a data point collected when the Puffer server receives a video chunk acknowledgement from a client. In `client_buffer_X.csv`, each line represents a data point containing a message reported by the client, when certain events occur and on a regular interval.

### 3.2 Stream Statistics

When a Puffer client watches TV for the first time or reloads the player page, it starts a new "session", identified by `session_id` in the CSVs. When a client switches channels, it enters into a different "stream" but still remains in the same "session", which uses the same TCP connection. `stream_stats_X.csv` aggregates each day's packet-level statistics in the raw data files, including each stream's mean SSIM, mean delivery rate, startup delay, and stall time.

[1]https://puffer.stanford.edu/results/

### 3.3 Scheme Statistics

A scheme refers to a particular ABR algorithm. All the algorithms that have been tested since the start of the Puffer project are visualized in Figure 1. Each line in the `day_all_scheme_stats_X.csv` file summarizes the number of streams for the scheme, as well as its mean and 95% confidence interval for SSIM and stall ratio. This file has the highest level of daily aggregation and the least amount of raw information. However, since each raw data file can be on the order of 100 MB, the aggregated stream stats and scheme stats files are necessary for some analyses over large date ranges.

## 4 Experiments

We have computed the following experiments on the Puffer data set.

- Replication of the original paper's analysis on 2020 data
- Streaming metrics throughout the day across different years
  - Traffic
  - Stalls
  - Video quality
  - Delivery rate
- Stall occurrences over the length of the stream
- Correlations between rebuffering, stall time, delivery rate, round-trip time (RTT), and their variations
- Startup time comparisons between ABR algorithms

### 4.1 2020 Comparisons

First of all, we have replicated the main result of the Puffer Paper which is the values and confidence intervals of video quality (SSIM) and stall time (see Figure 3a). We aggregated the results of from `scheme_stats` files and calculated both mean and confidence intervals for SSIM and stall time. Our replicated results very well correspond to the original graph produced in the puffer paper. We also computed the same graph for 9 month periods starting in March 2020 and February 2022. Since the

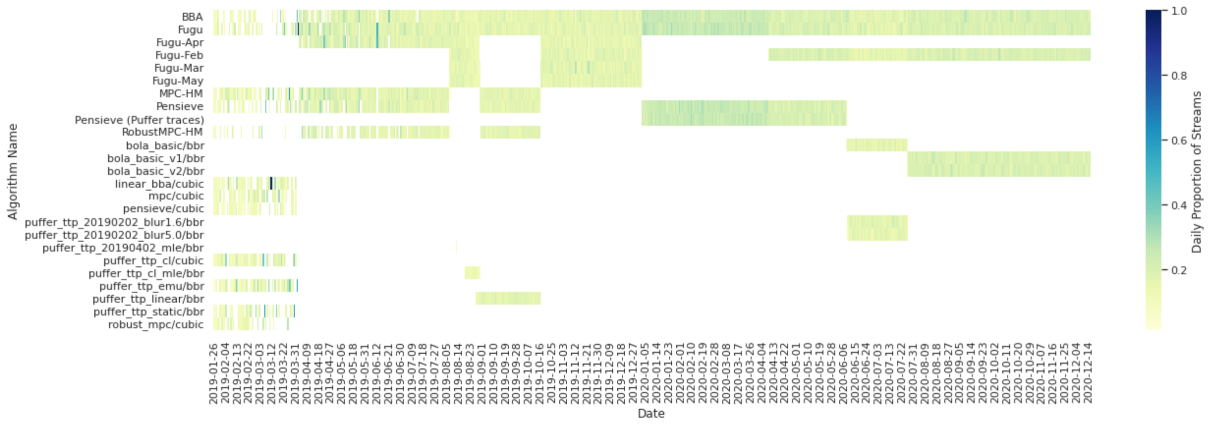


Figure 1: All algorithms tested on the Puffer platform from January 2019 to December 2020

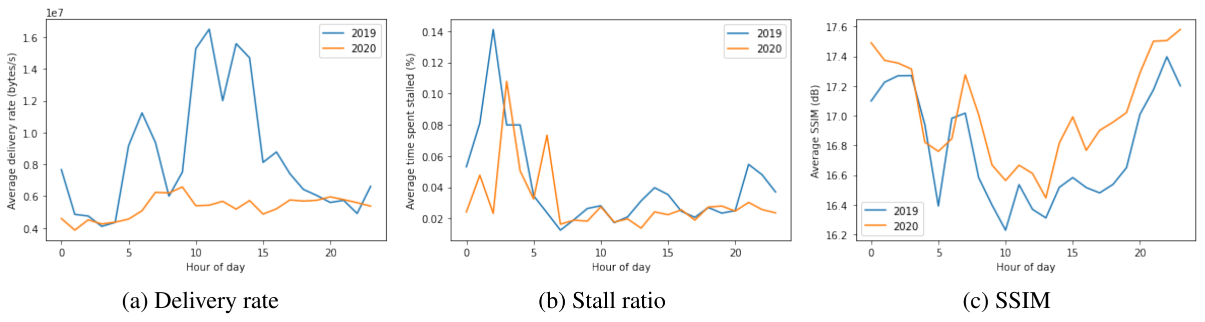

(a) Delivery rate (b) Stall ratio (c) SSIM

Figure 2: Fugu's average delivery rate/stall ratio/SSIM per hour in 2019 vs 2020

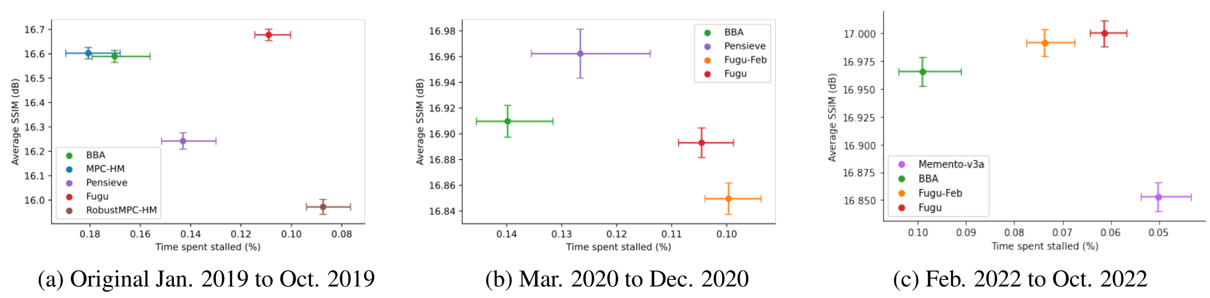
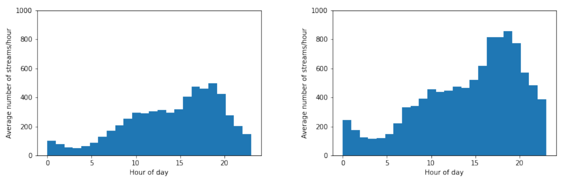

(a) Original Jan. 2019 to Oct. 2019 (b) Mar. 2020 to Dec. 2020 (c) Feb. 2022 to Oct. 2022

Figure 3: Replicated Puffer results

ABR algorithms in the experiment changed over time (see Figure 1), the only common schemes over all 3 time periods were Fugu and BBA. Although other algorithms were experimented with, the ones not displayed in Figure 3 either covered too short of a time period or did not have comparable performance to Fugu.

For some reason, the version of Fugu trained in February 2019 has been included in experiments throughout 2022. The standard version of Fugu was retrained daily until October 2022. Fugu-Feb's comparable performance to the daily-retrained Fugu indicates that training on old network traces still generalize reasonably well, but fails to achieve as high a SSIM. Memento is a scheme introduced to the Puffer experiment in 2021 and achieved a lower stall ratio than Fugu at the cost of SSIM, but there's no publicly available information on its design. Pensieve achieves a higher SSIM than Fugu in 2020, but since it was only included in experiments until June 2020, it's hard to draw a meaningful conclusion without spanning most of the date range considering how network conditions change over time. Looking at the scales of the axes, it's clear that there's significant variation in network conditions, which we attempt to



(a) Original 2019 (b) 2020

Figure 4: Average number of streams per hour

explore next.

#### 4.1.1 Streaming metrics throughout the day

In Figures 4 and 2, we have generated graphs for average streams, delivery rate, stall ratio, and SSIM per hour. The horizontal axis of all graphs are the 24 hours in a day and the vertical axis is computed by taking the mean of corresponding fields in the `stream_stats` file over each hour.

Figure 4 shows the average number of streams per hour in 2019 and 2020. Assuming most users are in the same time zone as the experiment location (Stanford, CA), the distribution of usage looks reasonable across both time periods. High usage is from 15:00 - 23:00 when most people are finished working and low usage is from 0:00 - 6:00 when most people are asleep. The average number of streams per hour is noticeably higher in 2020 than in 2019, likely because more people are at home and able to watch TV.

In Figure 2a, the average delivery rate is fairly constant in 2019. However, in 2020, the delivery rate is much more variable, peaking in the middle hours of the day. In addition to the changed distribution, the magnitude of the delivery rate is significantly higher. Note that delivery rate is averaged over streams, so this isn't due to the increased number of streams shown in Figure 4.

In Figure 2b, both time ranges demonstrate that stalls are around 2x - 5x more likely between 00:00 and 05:00 than during other time periods. This is a surprising results, as one would expect there to be more stalls in the late afternoon hours containing the most traffic. In addition, despite 2020 having more stream traffic than 2019, stall ratios are similar.

In Figure 2c, SSIM patterns are similar across both date ranges, with 2020's SSIMs being slightly higher. The dip in SSIM at around noon is interesting, since it doesn't align with traffic patterns in Figure 4, but resembles an inverted plot of 2020's delivery rate distribution. Since delivery rate is measured in bytes/sec, an inverse relationship is unexpected since a high delivery rate should result in high quality video chunks.

### 4.2 Feature Correlations

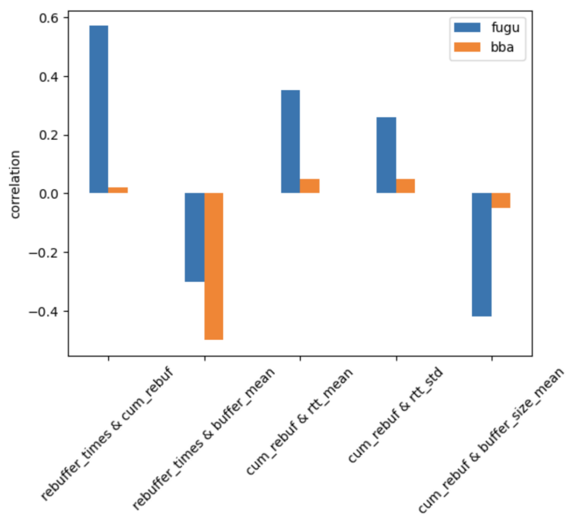


Figure 5: Correlation between stream stats

Figure 5 illustrates the relationship between total rebuffering time and several variables: rebuffering frequency, average buffer size, average round-trip time (RTT), and RTT variation for each stream under the Fugu and BBA protocols over a one-month period from June 12, 2021, to July 12, 2021. While the BBA protocol exhibits no discernible correlation between these variables, the Fugu protocol demonstrates a weak to moderate correlation. As anticipated, the frequency of rebuffering, along with average and variation in RTT—a reflection of poor network connectivity—exhibits a positive correlation with stall time. Conversely, the average client buffer size is negatively correlated with stall time. This is because a larger playback buffer suggests superior network capacity and lower latency. This data implies that the Fugu protocol may have a superior capability to decide when to rebuffer, thereby handling slower streams more effectively. The correlation between other pairs of variables are quite similar and expected. The table for the entire list of feature correlations is in the appendix.

### 4.3 Stall Occurrences

Minimizing stalls is one of the major goals of ABR algorithms, and it's something that modern ABR algorithms do relatively well. The Puffer paper cites that only 4% of streams had any stalls. Even of the streams that experience rebuffering, stalls compose a fraction of a percent of the total stream time.

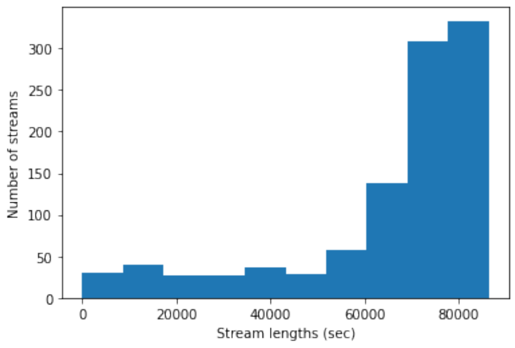


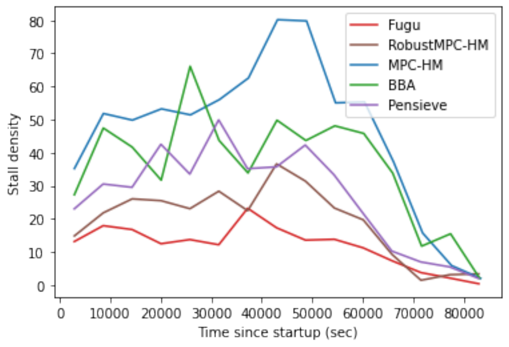
Figure 6: Stalled stream lengths

Figure 6 shows the distribution of stream lengths for streams that experienced rebuffering during the original experiment's date range. It makes sense that long streams are more likely to experience stalls, but it's interesting that the distribution is so skewed considering that the vast majority of streams are short. The presence of so many 60,000+ second streams (16+ hours) is strange, however.

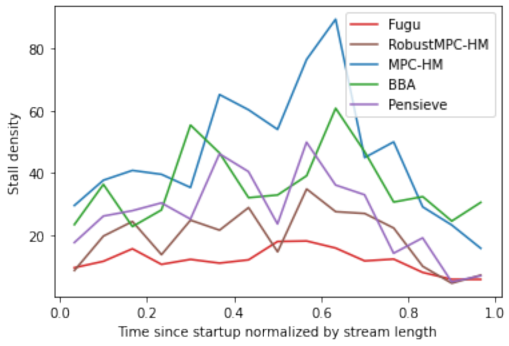
Figure 7a was produced using data from the original experiment's date range and shows the distribution of stall occurrences over a stream. Figure 7b shows the distribution of stall occurrences normalized by stream length. Unsurprisingly, schemes with higher stall ratios like BBA and MPC-HM generally experience more stalls than schemes like RobustMPC and Fugu. If the length of streams were more evenly distributed, Figure 7a would likely have a higher density on the left side. But from the normalized plot, we can see that stall occurrences are generally evenly distributed along the length of a stream; there are no early stalls to push through or end stalls that drive users to stop streaming. This doesn't include startup time, which will be covered next.

### 4.4 Startup Time Comparison

Figure 8 shows a distribution of startup times for different schemes during the original experiment's date range. Startup times in general are very low (less than 1 sec) but have a long tail. Most schemes have a similar distribution of startup times. Notably, Pensieve has a longer tail of startup times than the other schemes. Maybe Pensive's reinforcement learning agent doesn't generalize as well to the beginning of streams, or Pensieve's chunk-level simulator doesn't accurately simulate real-world packet arrival.



(a) Stall occurrences



(b) Normalized stall occurrences

Figure 7: Stall occurrences within stream

## 5 Limitations

One of the main contributions of the original Puffer paper was the huge amount of data gathered from real-world experiments. The large sample size results in narrow enough confidence intervals to draw meaningful conclusions from highly variable network data. Ideally, all of our results would have been derived from the raw data files produced by their experiments, but a dataset of 9 months of raw packet data would be several terabytes large, too large for us to process. Because of this, we had to rely on the aggregated `stream_stats_X.csv` and `day_all_scheme_stats_X.csv` files for several of our analyses. This leads to a few issues:

- Reduced date ranges on analytics (Figure 5)
- Aggregation of aggregated summaries (Figure 3)

### 5.1 Reduced date ranges

To analyze packet-level data, there was no other option than to use the raw data files. Since each data file can be on the order of 100 MB, we had to use a date range smaller than the 9 months used in the original experiments. These results are therefore

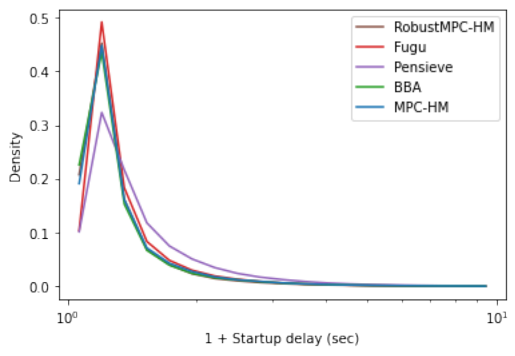


Figure 8: Startup delay

more subject to variation of network conditions, client usage, and scheme assignment.

### 5.2 Aggregation of aggregates

Each day's `day_all_scheme_stats_X.csv` file contains day-level summaries of each scheme's number of streams, SSIM mean/confidence interval, and stall ratio mean/confidence interval. Aggregating the daily means of these values over a larger date range is easy: take the weighted average of the mean SSIMs/stall ratios with weights corresponding to stream time. However, finding aggregate confidence intervals is more complicated, especially when stall ratios are not normally distributed. The original Puffer paper solved this by sampling stall ratios from buckets. Our simulation of this sampling wasn't entirely accurate, so the confidence intervals are slightly different from what was shown in the original paper.

## 6 Conclusion

This study has furthered our understanding of Adaptive Bitrate Selection (ABR) algorithms by revisiting and extending the Puffer study with a particular focus on data generated after the original publication. Our investigation covered a range of analytical perspectives, including comparisons of ABR algorithms, feature correlations, stream metric variations over time, and an exploration of stall occurrences and startup times.

Our analysis of the data generated from the Puffer study after 2019 has led to several significant insights into the performance of ABR algorithms over time. Fugu, in particular, continues to perform well into 2022. The performance gap between different ABR algorithms has diminished compared to 2019, suggesting the field is maturing and converging in its methods. But this observation may also imply *learning in situ* may not be good enough. We noticed that there were more streams in 2020 compared to the previous year, with most streams occurring in the evening. The SSIM and stall ratio remained consistent between 2019 and 2020, but this did not hold for the delivery rate, which saw more significant variations. This trend might be related to pandemic-induced flattening of the network utilization where the peaks and troughs aren't that different, which warrants further exploration.

We also found that different ABR algorithms demonstrated varied correlations between network and buffer metrics and stall time, suggesting that different algorithms may have different sensitivities to these factors. Additionally, stall occurrences were uniformly distributed during a stream, indicating that modern ABR algorithms are managing to maintain consistent performance throughout the duration of a stream. Our analysis also revealed similar startup time distributions across all schemes. This suggests that, regardless of the underlying algorithm, startup times are less affected by the specific scheme. The limitations of our study are primarily due to the significant volume of data generated by the Puffer study.

Our extended analysis of the Puffer experiment data has yielded valuable insights into the real-world performance of ABR algorithms, the impact of network and buffer conditions on streaming quality, and the effect of user behaviour changes brought about by external factors like the pandemic. As video streaming continues to dominate internet traffic, the importance of robust, efficient, and adaptable ABR algorithms will only grow, making this an exciting field for further research and and the Puffer study's dataset provides an excellent resource for this ongoing work.

## References

Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. 2014. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. *SIGCOMM Comput. Commun. Rev.*, 44(4):187–198.

Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. 2017. Neural adaptive video streaming with pensieve. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '17, page 197–210, New York, NY, USA. Association for Computing Machinery.

Francis Y. Yan, Hudson Ayers, Chenzhi Zhu, Sadjad Fouladi, James Hong, Keyi Zhang, Philip Levis, and

Keith Winstein. 2020. Learning in situ: a randomized experiment in video streaming. Stanford University, Tsinghua University.

Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. 2015. A control-theoretic approach for dynamic adaptive video streaming over http. *SIGCOMM Comput. Commun. Rev.*, 45(4):325–338.

## A Appendix

| Features | Fugu | BBA |
|---|---|---|
| rebuffer_times & cum_rebuf | 0.571807 | 0.027174 |
| rebuffer_times & buffer_size_mean | -0.296854 | -0.500066 |
| rebuffer_times & buffer_size_std | 0.083257 | 0.067621 |
| rebuffer_times & delivery_count | 0.061316 | 0.131781 |
| rebuffer_times & delivery_rate_mean | -0.043585 | -0.192072 |
| rebuffer_times & delivery_rate_std | 0.024082 | -0.154173 |
| rebuffer_times & rtt_rate_mean | 0.158484 | 0.225505 |
| rebuffer_times & rtt_rate_std | 0.084304 | 0.110536 |
| cum_rebuf & buffer_size_mean | -0.424771 | -0.054727 |
| cum_rebuf & buffer_size_std | 0.157449 | 0.045725 |
| cum_rebuf & delivery_count | 0.045707 | -0.012339 |
| cum_rebuf & delivery_rate_mean | -0.124119 | -0.029075 |
| cum_rebuf & delivery_rate_std | -0.076491 | -0.024313 |
| cum_rebuf & rtt_rate_mean | 0.353258 | 0.053313 |
| cum_rebuf & rtt_rate_std | 0.260036 | 0.046776 |
| buffer_size_mean & buffer_size_std | -0.404139 | -0.314443 |
| buffer_size_mean & delivery_count | 0.119063 | 0.095653 |
| buffer_size_mean & delivery_rate_mean | 0.433047 | 0.480951 |
| buffer_size_mean & delivery_rate_std | 0.305227 | 0.400978 |
| buffer_size_mean & rtt_rate_mean | -0.592613 | -0.607429 |
| buffer_size_mean & rtt_rate_std | -0.369557 | -0.374137 |
| buffer_size_std & delivery_count | -0.254835 | -0.236946 |
| buffer_size_std & delivery_rate_mean | -0.323844 | -0.360481 |
| buffer_size_std & delivery_rate_std | -0.227855 | -0.285981 |
| buffer_size_std & rtt_rate_mean | 0.355153 | 0.357718 |
| buffer_size_std & rtt_rate_std | 0.394870 | 0.450587 |
| delivery_count & delivery_rate_mean | -0.057842 | -0.065187 |
| delivery_count & delivery_rate_std | -0.046347 | -0.067720 |
| delivery_count & rtt_rate_mean | -0.057103 | -0.036056 |
| delivery_count & rtt_rate_std | -0.073654 | -0.066886 |
| delivery_rate_mean & delivery_rate_std | 0.936218 | 0.953994 |
| delivery_rate_mean & rtt_rate_mean | -0.429647 | -0.481755 |
| delivery_rate_mean & rtt_rate_std | -0.264405 | -0.322108 |
| delivery_rate_std & rtt_rate_mean | -0.334841 | -0.413105 |
| delivery_rate_std & rtt_rate_std | -0.193856 | -0.263274 |
| rtt_rate_mean & rtt_rate_std | 0.667444 | 0.683500 |

Table 2: All feature correlations