A Conditional-Probability-Distribution Model for Bandwidth Estimation with Application in Live Video Streaming

Weijia Zheng

1Department of Information Engineering, CUHK
1Department of Mathematics, CUHK
1wjzheng@link.cuhk.edu.hk

Abstract

Experience of live video streaming can be improved if the video uploader have more accurate knowledge about the future available bandwidth. Because with such knowledge, one is able to know what sizes should he encode the frames to be in an ever-changing network. Researchers have developed some algorithms to predict throughputs in the literature, from where some simple hence practical ones. However, limitation remains as most current bandwidth prediction methods are predicting a value, or a point estimate, of future bandwidth. Because in many practical scenarios, it is desirable to control the performance to some targets, e.g., video delivery rate over a given target percentage, which cannot be easily achieved via most current methods.

In this work, we propose the use of probability distribution to model future bandwidth. Specifically, we model future bandwidth using past data transfer measurements, and then derive a probability model for use in the application. This changes the selection of parameters in application into a probabilistic manner such that given target performance can be achieved in the long run. Inside our model, we use the conditional-probability method to correlate past and future bandwidth and hence further improve the estimating performance.

Keywords
Network throughput estimation, conditional probability, relative frequency, live video streaming, empirical method

1 Introduction

With the development of smartphones and high-speed mobile data networks such as 3G and 4G/LTE, live video streaming has long become part of our lives in the entertainment or casual fields. In recent years, it has played a vital role in the workplace as well. As we all experienced in person, the year of 2020 witnessed Zoom’s significant increase [1] in the usage of remote work, distance education, and online social relations.

Given the importance of video streaming and the high peak bandwidth nowadays in mobile data networks, bandwidth fluctuation remains a challenging obligation for its unpredictable characteristic by its wireless nature, which may affect the quality of clients’ experience. Several prediction methods were established or used in the literature, such as arithmetic mean (AM), multiple linear regression (MLR), ARIMA, LSTM, etc. However, most of them predict a point estimate of future bandwidth, or gives a confidence interval with further assumptions, which is hard to guarantee its accuracy to a specified level in practice.

Motivated by the ideas of using an empirical conditional probability for prediction in financial [2] and transportation fields [3], this work contributes to establishing a conditional-probability model in bandwidth prediction. And a simulator of the uploading part of a live video streaming, referencing the one in [4], is implemented for simulation and demonstrating the feasibility of such a method. To mimic a real network environment, the simulator uses a packet-level with timestamp trace data measured from real-world network sources of 3HK 4G.

The rest of the paper is organized as follows. Section 2 introduces the settings of the problem. Some analyses and related works are presented in Section 3. In Section 4 we will derive the proposed encoding scheme in detail. Numerical results of it on different network environments are shown in Section 5, before conclusions are outlined in Section 6.

Besides, unless otherwise stated, all the bitrate variables are in Mbps, and all the time variables are in seconds in this paper.

2 Problem Background

In our study, to concentrate on the uplink part, we considered the uplink part exclusively with some further simplifications. We do not consider the downlink part of the streaming process in the simulator and the general scenario can be described as the following.

An uploader generates frames one by one with an equal time difference (i.e., 1/FPS second) in between. Assume that TCP is used, once a frame is sent from the uploader side, the uploader will not consider it anymore but will only take care of later frames. The uploader then starts transmitting some newest possible frame as buffer time
achieved. When the streaming process terminates, should there be any not-ever-sent frame, we count it as a frame loss.

\[ C_i = s_i/t_i, \]

where \( s_i \) is frame \( f_i \)'s size which is determined by the uploader itself. And \( t_i \) is how much time the uploader cost to process \( f_i \), which the uploader can always make a record on. When frame \( f_{i+1} \) needs to be transmitted, the uploader may predict how will the network soon (i.e., \( C_{i+1} \)) behave through utilizing all those pieces of knowledge about the network’s history it was obtained using the above way.

A study carried by Zou et al. [5] revealed that the optimal case happens when we can achieve

\[ C_{i+1} = \overline{C_{i+1}}. \]

Because then \( \forall j \), we can let \( f_j \) finish its transmission at \( t_g^{(j+1)} \), the exact time when frame \( f_{j+1} \) is just generated. By this, we are sending \( s_j \) as large as possible while not losing any single frame and introducing no delay.

Besides, a minimal frame size \( s_{\text{min}} \), namely, \( \forall j, s_j \geq s_{\text{min}} \) is required. Because we cannot let \( s_j \) be arbitrarily small. We also set some fixed value \( t_B \) as our initial buffer time, which usually is some multiple of \( 1/\text{FPS} \). This allows the uploader to transmit some older frames that are still within the buffer time. Note that the buffer time may change with \( i \), since it may be eaten up by overestimating the throughputs in the past, which causes a longer transmission time than expected.

\[ \text{loss rate} := \frac{\# \text{ frames discarded during the } T \text{ period}}{\# \text{ frames generated during the } T \text{ period}} \]

and

\[ \text{(average) bitrate} := \frac{1}{T} \sum_i s_i. \]

An ideal algorithm will lead to low (close to zero) loss rate and large average bitrate. Both can be reached at the same time when the perfect prediction mentioned is achieved.

It is noticeable that the two metrics (loss rate and bitrate) can be easily optimized individually. By letting \( s_{\text{min}} \) to be very large or small values and encode every frame as size \( s_{\text{min}} \). This gives us an idea that to eliminate loss rate we do not necessarily have accurate knowledge of \( C_j \)'s, but some lower bounds will do. And there exists an upper limit of bitrate which we can use to check how an algorithm behaves.

Besides sophisticated methods explored in [6], [7] and [8], two seemingly-trivial throughput predicting algorithms got suggested in [9]. They are arithmetic mean (AM) and multiple linear regression (MLR). The AM algorithm returns a moving average of the past \( C_j \)'s. Suppose \( K \) is the number of past \( C_j \)'s we look back on, then one calculates

\[ C_{i+1} = \frac{1}{K} \sum_{k=1}^{K} C_{i+1-k} \]

as an estimate for \( C_{i+1} \). However, this assumes each \( t_{i+1-k} \)'s are the same, which cannot be guaranteed in reality, hence the (1) should be adapted to

\[ C_{i+1} = \sum_{k=1}^{K} C_{i+1-k} \cdot t_{i+1-k}, \]

Multiple linear regression has a general form (one may manipulate the formula to avoid multicolinearity)

\[ \hat{C}_{i+1} = b_0 + b_1 C_i + b_2 C_{i-1} + \ldots + b_K C_{i-K+1}, \]

where \( b_0 \) is the intercept and \( b_k \)'s are weights. The famous ARIMA, discussed in [7] and [8], and many others are variants of MLR. However, note that this also needs to be adapted in practice, since \( t_{i-K+1} \)'s may vary. These algorithms do not require much training and are relatively simple to apply, hence got suggested by many.

As one may be expected, in [6] and [8] LSTM also got some praise. However, as different people are using different datasets, and the results between different algorithms are quite matched in strength, it is hard to say there is a best algorithm or a game-changer in this topic.

## 3 Related Works

We use two simple metrics to measure how an encoding scheme performs during a time period of length \( T \) (in second). They are

\[ \text{loss rate} := \frac{\# \text{ frames discarded during the } T \text{ period}}{\# \text{ frames generated during the } T \text{ period}} \]

and

\[ \text{(average) bitrate} := \frac{1}{T} \sum_i s_i. \]

As one may be expected, in [6] and [8] LSTM also got some praise. However, as different people are using different datasets, and the results between different algorithms are quite matched in strength, it is hard to say there is a best algorithm or a game-changer in this topic.

## 4 Our Proposed Method

### 4.1 Motivation

For any fixed given time \( t \), we look back on the past and divide the past time into \( \tau \) equally sized non-overlapping intervals, i.e., \( (t-\tau,t], (t-2\tau,t-\tau], \ldots, (t-(j+1)\tau, t-j\tau] \ldots \) where \( j \in \mathbb{N} \).

\[ C_{j,\tau} \]

Denote \( C_{j,\tau} \) as the average throughput in the time interval \( (t-(j+1)\tau, t-j\tau) \). In other words, the previous \( j \)-th time interval of length \( \tau \) counting from \( t \). Since the throughput series indexed by \( j \), i.e., \( \{C_{j,\tau} | j \in \mathbb{N} \} \) forms a
stochastic process over time. It is conceivably to believe that the throughput is highly correlated with its past values during some past time intervals. Especially, we may expect more recent past throughput data to exhibit higher correlation with future throughput and is therefore more helpful in predicting the (short-term) future. Which is verified by [9] that for some fixed $\tau = 5$, $I(C_0,\tau;C_k,\tau)$ decreases as $k \in \mathbb{N}$ increases.

Motivated by the research works in predicting short term traffic speed, which utilize history data to generate a Markovian state transition matrix [10] or a conditional CDF [3] to predict future travel speed, we propose a conditional PDF based method that is able to predict short-term future throughput as well as control the streaming behaviour in terms of frame loss rate at the uploader side.

4.2 Implementation Procedures

To stay the loss rate low meanwhile not wasting too much available bandwidths, we try to find promising $s_i$’s.

Suppose we want the loss rate to be no larger than some fixed $\epsilon \in (0, 1)$. Now is time $t_s^{(i)}$, frame $f_i$ starts to transmit. By discussions in section 2, $f_i$ will be discarded if

$$t_s^{(i)} + s_i/C_i > t_{g}^{(i+1)} + t_{b}^{(i)}.$$  

(3)

Recall that $C_i$ is defined to be the average throughput during $f_i$’s transmission. $t_{b}^{(i)}$ (not $t_{g}$ in general) is the available buffer time at this moment.

Let the probability of $f_i$ not to be discarded be no less than $1 - \epsilon$. Then we obtain

$$P(t_s^{(i)} + s_i/C_i \leq t_{g}^{(i+1)} + t_{b}^{(i)}) \geq 1 - \epsilon,$$

which is equivalent to

$$P(C_i(t_{b}^{(i)} + T_i) \geq s_i) \geq 1 - \epsilon,$$

(4)

where

$$T_i := |t_{g}^{(i+1)} - t_{s}^{(i)}| +.$$ 

In order to enhance the bandwidth efficiency, we choose an $s_i$ such that the equality in (4) is taken, i.e.,

$$P(C_i(t_{b}^{(i)} + T_i) \geq s_i) = 1 - \epsilon.$$  

(5)

Since the uploader keeps a log of past throughputs of the network, it is possible to construct a time series

$$\{C_j(t_{b}^{(i)} + T_i) \mid j = 0, 1, \ldots\}$$

(6)

Where $C_j(t_{b}^{(i)} + T_i)$ denotes the average throughput in interval $\{t_{s}^{(i)} - (j+1)(t_{b}^{(i)} + T_i), t_{s}^{(i)} - j(t_{b}^{(i)} + T_i)\}$. That is, the uploader constructs an array $\hat{s}$, whose $j$-th entry is how many Mb of data can be transmitted in the $j$-th interval with length $(t_{b}^{(i)} + T_i)$ counting back from $t_{s}^{(i)}$ following $t$.

Define $s_{k,\tau} := C_{k,\tau} \cdot \tau$. (5) then changes to

$$P(s_{-1,t_{b}^{(i)} + T_i} \geq s_i) = 1 - \epsilon.$$  

(7)

According to (6) and the discussions following it, the uploader maintains an array

$$\hat{s} := \{s_{0,t_{b}^{(i)} + T_i}, s_{1,t_{b}^{(i)} + T_i}, \ldots, s_{J-1,t_{b}^{(i)} + T_i}\}$$

as a time-series, which represents a relative frequency of $s_{-1,t_{b}^{(i)} + T_i}$ in terms of its realizations in (7). However, to reach (7), we let the probability be conditioning on its predecessor, $s_{0,t_{b}^{(i)} + T_i}$, which is known as that’s the first element of $\hat{s}$. Thus (7) can be replaced by

$$P(s_{-1,t_{b}^{(i)} + T_i} \geq s_i \mid s_{0,t_{b}^{(i)} + T_i} = \text{some known value}) = 1 - \epsilon.$$  

(8)

Define

$$s_{0}^{*} := \{s_{-1,t_{b}^{(i)} + T_i} \in \hat{s} \mid s_{0,t_{b}^{(i)} + T_i} \approx s_{0,t_{b}^{(i)} + T_i}\}$$

(9)

Remember that $s_{-1,t_{b}^{(i)} + T_i} = C_{-1,t_{b}^{(i)} + T_i} \cdot (t_{b}^{(i)} + T_i)$ is a proper size for $f_i$, which we want to determine by estimating $C_{-1,t_{b}^{(i)} + T_i}$ (plays the role as $C_i$ in (4)). Hence $P(s_{-1,t_{b}^{(i)} + T_i} \mid \hat{s}_{0,\tau} = \text{some known value})$ can be approximated by $s_{0}^{*}$ since the $s_{0}^{*}$ is its one conditional relative frequency.

By (8), one should take

$$s_i = Q_{X}(s_{-1,t_{b}^{(i)} + T_i} | \hat{s}_{0,\tau} = \text{quantile}(\hat{s}_{0,\tau}, \epsilon)) \approx \text{quantile}(\hat{s}_{0,\tau}, \epsilon)$$

(9)

as our chosen frame size $s_i$ for $f_i$, where $Q_X(\gamma)$ denotes the $\gamma$ quantile of random variable $X$. And $\text{quantile}(\hat{\alpha}, \gamma)$ is the numpy quantile function in Python, which returns the value at the $\gamma$-th quantile value of $\hat{\alpha}$.

The algorithm of deciding $s_i$ is articulated in Algorithm 1.

After $s_i$ is determined, then by the end of $f_i$’s transmission, $t_i$ will be available. Then we update the frame-level history data $s$ and $t$ by appending $s_i$ and $t_i$ respectively, and update the buffer time based on if $t_i < 1/FPS$ or $t_i > 1/FPS$. Thus we are ready for the next round of transmission.

One may note that in (4), to be more conservative, one can change $t_{b}^{(i)}$ by any smaller value $\alpha t_{b}^{(i)}$ with $\alpha \leq 1$. And then all the following steps should change correspondingly.

5 Simulation Results

The following section shows the results of the proposed method. The explanatory variables are the two metrics, minimal frame size $s_{\text{min}}$ and initial buffer time $t_B$. We ran simulations on two different networks. For simplicity, we will only compare with AM algorithm, with its parameter in (2), $K = 5, 16, 128$ respectively. And we always let the target loss rate to be $\epsilon = 0.05$. For all plots whose $x$-axis is $t_B$, we let the $s_{\text{min}}$ fixed to be some small value. Since in a live video streaming, the $t_B$ cannot set be too large, we let it be up to 0.1s. Similarly, too large loss rate like 20% or above are not meaningful, we will focus on when the loss rate is below 20%.

Finally, the training time for our algorithm is 120 seconds, and the FPS = 60.

The below groups of figures show how our conditional probability method behaves in terms of loss rate and bitrate when $s_{\text{min}}$ and $t_B$ increases individually. The network (call it network 1) in Fig. 3 and Fig. 4 has a mean of throughput $\approx 12$ Mbps. And the network (network 2)
Algorithm 1: Determination scheme of $s_i$

Data: updated frame-level sizes: $s = [s_0, s_1, ..., s_k]$, updated frame-level timestamp: $t = [t_0, t_1, ..., t_k]$, buffer time at the moment, $t_i^{(j)}$, time before $t_i^{(j+1)}$, $T_i = [t_i^{(j+1)} - t_i^{(j)}]$, how many intervals do we trace back, $J \in \mathbb{N}^*$, loss rate target, $\epsilon \in (0, 1)$.

Result: Maximal frame size $s_i$ s.t. (5) holds.

\[
\hat{s} \leftarrow []
\]

\[
j \leftarrow 0
\]

while $\text{Len}(\hat{s}) < J$

\[
\hat{s} \text{ append } (F(j) - F(j))
\]

\[
j \leftarrow j + 1
\]

end

$s_{i0} \leftarrow [\hat{s}[n-1]]$ in $\hat{s}$ for $\hat{s}[0] \approx \hat{s}[n]$

($\approx$ can be defined as within 0.05 variation)

$s_i \leftarrow \text{quantile}(\hat{s}_{[0]}, \epsilon)$

return $s_i$

procedure $F(j)$

\[
U \leftarrow \max\{u | \sum_{m=U}^{k} t_m \geq j \cdot (t_b^{(i)} + T_i)\}
\]

\[
t_{\text{Res}, U} \leftarrow \sum_{m=U}^{k} t_m - j \cdot (t_b^{(i)} + T_i)
\]

\[
L \leftarrow \max\{l | \sum_{m=U}^{L-1} t_m \geq t_b^{(i)} + T_i - t_{\text{Res}, U}\}
\]

\[
t_{\text{Res}, L} \leftarrow j \cdot (t_b^{(i)} + T_i) - t_{\text{Res}, U} - \sum_{m=U}^{L-1} t_m
\]

\[
F_j \leftarrow \frac{t_{\text{Res}, U}}{t_L} \cdot s_U + \frac{t_{\text{Res}, L}}{t_L} \cdot s_L + \sum_{m=U}^{L-1} s_m
\]

return $F_j$

end procedure

Figure 3: Network 1. Fix $s_{\text{min}}$ be small, bitrates and loss rates as $t_B$ increases.

Figure 4: Network 2. Fix $s_{\text{min}}$ be small, bitrates and loss rates as $t_B$ increases.

Note that when $t_B$ is small, our method gives lower bitrate comparing with the AM algorithms, which is reasonable since that is the trade off for having a controlably (low) loss rate. Because AM algorithms are mean-bitrate pursuing algorithms and ours is not. As $t_B$ increases, our method's bitrate increases while the loss rate remains fluctuating around 0.05. When $t_B$ reaches 0.1s (= 6/FPS), its bitrate is leveraged to the maximal level. Which is just as we expected.

Figure 5: Fix $t_B = 1$FPS, bitrates and loss rates as $s_{\text{min}}$ increases.

Figure 6: Fix $s_{\text{min}}$ be small, bitrates and loss rates as $t_B$ increases.

The above Fig. 5 and 6 shows when fix $t_B$ to be 1/FPS, how bitrate and loss rate changes as $s_{\text{min}}$ increases in network 1 and 2 respectively. As expected again, our method

in Fig. 5 and 6 has a mean of throughput $\approx 8$ Mbps. We compare with AM algorithms and the marginal version (does not use conditioning) of our method.
suffers a bitrate loss comparing with the AMs as the trade
off for controlling the loss rate to be under $\epsilon = 0.05$, which
can be observed as a flat line in the plots. However, it
should be surprising comparing with the minimal frame
scheme and the marginal probability method’s behaviour.
Conditional probability method is really reaching some
balance between loss rate and bitrate, under any given
target $\epsilon$ and buffer time $t_B$. When the $s_{\text{min}}$ goes large,
then all algorithms will be taken over by $s_{\text{min}}$. That is
why all algorithms "converges" together in the right end of
Fig. 5 and 6.

From the above results, the conditional probability
method is able to control certain loss rate target, while
its bandwidth efficiency can be relatively high.

6 Conclusion
This work proposed and demonstrated a history-based
short-term bandwidth predicting method called condi-
tional probability method, which especially has a practical
value in live video streaming.

Rather than doing a (point) estimation on the future
throughput, conditional probability method uses the em-
pirical conditional relative frequency to find the largest
frame size while eliminating the probability of frame loss
under any networks. It is expected to give good perfor-
mances in reality via its simulation results under different
packet-level testing sets.

Possible improvements such as giving a more concrete
theoretical proof of this method, a faster implementation
of the algorithm in both space and time complexities, and
ways to obtain more reliable past data are expected in
future works to further enhance this method’s efficiency
and robustness.

References
[1] NASDAQ. "Zoom Video Communications, Inc. to Join the NASDAQ-100 Index Beginning April 30, 2020", April 23, 2020. [Online]. https://www.nasdaq.com/press-release/zoom-video-communications-inc.-to-join-the-nasdaq-100-index-beginning-april-30-2020

[2] R. R. Joss, "Analyzing Investment Data Using Condi-
tional Probabilities: The Implications for Investment Forecasts, Stock Option Pricing, Risk Premia, and CAPM Beta Calculations," Journal of Financial and Economic Practice, vol. 6, No. 2, 2006.

[3] Z. Zhu, L. Tang, C. Xiong, X. Chen and L. Zhang,
"The conditional probability of travel speed and its
application to short-term prediction", Transportmet-
rica B: Transport Dynamics, 7.1, pp. 684-706, 2019,
doi: 10.1080/21680566.2018.1467289

[4] G. Yi and 8 others, "The ACM Multimedia 2019
Live Video Streaming Grand Challenge", MM ’19:
Proceedings of the 27th ACM International Confer-
ence on MultimediaOctober pp. 2622–2626, 2019,
doi:https://doi.org/10.1145/3343031.3356083

[5] X. K. Zou, J. Erman, V. Gopalakrishnan, E. Hale-
povic, R. Jana, X. Jin, J. Rexford and R. K. Sinha,
"Can Accurate Predictions Improve Video Streaming
in Cellular Networks?," in Proc. 16th Int. Workshop
Mobile Computing Systems and Applications (Hot-
Mobile’15), Santa Fe, USA., Feb. 2015, pp. 57-62.

[6] H. Na, Y. Shin, D. Lee and J. Lee, "LSTM-based
throughput prediction for LTE networks", ICT Ex-
press, 2021.

[7] D. Lee, D. Lee, M. Choi and J. Lee, "Prediction of
Network Throughput using ARIMA," 2020 Interna-
tional Conference on Artificial Intelligence in Infor-
mation and Communication (ICAIC), pp. 1-5, 2020,
doi: 10.1109/ICAIC48513.2020.9065083.

[8] H. Elsherbiny, H. M. Abbas, H. Abou-zeid, H. S. Has-
sanein and A. Noureldin, “4G LTE Network Through-
put Modelling and Prediction," 2020 IEEE Glob.
Commun. Conf.

[9] Y. Liu and Y. B. Lee, "An Empirical Study of
Throughput Prediction in Mobile Data Networks," 2015 IEEE Glob. Commun. Conf., pp. 1–6, 2016, doi:
10.1109/glocom.2015.7417858.

[10] Y. Qi and S. Ishak, "A Hidden Markov Model for
short term prediction of traffic conditions on free-
ways", Transportation Research Part C: Emerging
Technologies, vol. 43, pp. 95-111, 2014.