

# Statistical Learning Based Congestion Control for Real-time Video Communication

Tongyu Dai, Xinggong Zhang, Yihang Zhang, and Zongming Guo, *Member, IEEE*

**Abstract**—With the increasing demands on interactive video applications, how to adapt video bit rate to avoid network congestion has become critical, since congestion results in self-inflicted delay and packet loss which deteriorate the quality of real-time video service. The existing congestion control is hard to simultaneously achieve low latency, high throughput, good adaptability and fair bandwidth allocation, mainly because of the hardwired control strategy and egocentric convergence objective.

To address these issues, we propose an end-to-end statistical learning based congestion control, named Iris. By exploring the underlying principles of self-inflicted delay, we reveal that congestion delay is determined by sending rate, receiving rate and network status, which inspires us to control video bit rate using a statistical-learning congestion control model. The key idea of Iris is to force all flows to converge to the same queue load, and adjust the bit rate by the model. All flows keep a small and fixed number of packets queuing in the network, thus the fair bandwidth allocation and low latency are both achieved. Besides, the adjustment step size of sending rate is updated by online learning, to better adapt to dynamically changing networks.

We carried out extensive experiments to evaluate the performance of Iris, with the implementations of transport layer (UDP) and application layer (QUIC) respectively. The testing environment includes emulated network, real-world Internet and commercial LTE networks. Compared against TCP flavors and state-of-the-art protocols, Iris is able to achieve high bandwidth utilization, low latency and good fairness concurrently. Especially over QUIC, Iris is able to increase the video bitrate up to 25%, and PSNR up to 1dB.

**Index Terms**—Congestion control, real-time video streaming, low latency, statistical learning, adaptive adjustment.

## I. INTRODUCTION

WITH the widespread deployments of LTE/WiFi wireless networks and the forthcoming 5G [1], interactive video applications are growing exponentially, from the mobile video chatting, such as Skype [2], FaceTime, to AR/VR streaming [3] and cloud gaming [4, 5]. These video applications require not only higher bandwidth but also lower transmission delay. However, real-world network capacity is constrained, especially in wireless networks with unpredictable dynamics (e.g. random packet loss, channel fading, etc) [6, 7]. It imposes great challenges on nowadays video bitrate adaptation, which adjusts video streaming bit rate to reduce self-inflicted delay and loss.

The existing work related to rate adaptation can be mainly divided into two categories. The first is the research of adaptive bit rate (ABR) algorithms over application layer, including [8–11]. They usually use HTTP as the transport protocol and adjust the video bit rate according to bandwidth estimation and buffer state [9]. However, the underlying TCP

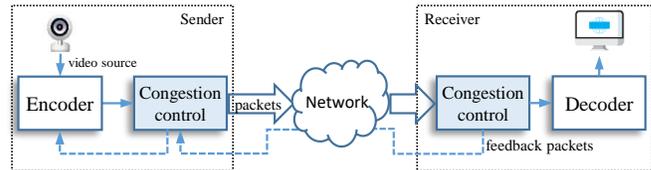


Fig. 1: The architecture of real-time video transmission.

of HTTP will essentially bring high latency [12], which is not suitable for real-time video transmission.

The second kind of rate adaptation related work is congestion control based on transport layer. As shown in Figure 1, congestion control plays an important role in the real-time video transmission, which adapts video bit rate to avoid self-inflicted delay and packet loss. Many rate adaptation schemes [13–18] have been proposed for video calling or video conferencing. Most of them focus on rate allocation upon given bandwidth [13, 14] or relay server selection [17, 18]. Seldom of them considers how to reduce end-to-end congestion delay. In addition to the conventional TCP-like algorithms [19, 20], there are also some end-to-end congestion control designed for real-time video streaming or low-latency transmission, which can limit the self-inflicted delay [15, 21, 22] or achieve TCP fairness [16]. Some methods also adjust video bit rate with online learning [23, 24]. But these existing algorithms still have some flaws, mainly including the following two aspects.

- **Non-coexistence of high throughput and low latency.** Considering only packet loss will lead to high queuing delay, but the algorithms overly concerned about delay also lead to low throughput [25].
- **Hardwired control strategy.** Most methods adjust sending rate with fixed step size or multiplier [26]. This manual mapping cannot always be optimal in changing networks, resulting in performance degradation.
- **Egocentric convergence objective.** Some algorithms based on objective function are self-centered and lack communication between concurrent data streams, thus they can not keep the same convergence goal for different clients, which leads to unfair bandwidth allocation.

Motivated by these issues, to obtain higher video transmission quality, we probe into the essence of congestion control and consider whether it is possible to design an algorithm that achieves the goals of low latency, high channel utilization, good adaptability to changing networks and fair bandwidth allocation. Some recently proposed algorithms [27, 28] have enlightened us: a congestion control with low-latency fairness

objective and a learned rate adjustment strategy.

In this paper, we start with an in-depth investigation into network congestion in LTE, WiFi and Internet, to explore the underlying principles of congestion delay. The data-driven analysis reveals that there is a strong correlation between transmission latency, sending rate and receiving rate. Inspired by the observations, we have designed Iris, an end-to-end learning-based congestion control algorithm for real-time video streaming. It mainly consists of two components: *low-latency fairness model* and *learning-based rate control strategy*. The fairness model forces all streams to keep a small and fixed number of packets queuing in network, achieving fairness and low congestion delay. The learning-based rate control builds a statistical function between round-trip time (RTT), sending rate and receiving rate, based on online linear regression learning. Then it is used to determine the proper sending rate, which avoids fixed adjustment step size and converges to the fairness objective more quickly.

The contributions of this paper can be summed up in the following three aspects:

- We explore the underlying reasons for congestion delay in video transmission and reveal the correlation between transmission latency, sending rate and receiving rate, which inspires us to design a low-latency algorithm with statistical learning.
- An adaptive bitrate adjustment scheme is introduced. According to the learning model, the rate adjustment step size can be periodically updated online, which avoids the hardwired control strategy to better adapt to dynamically changing networks.
- A low-latency fairness model which can be proved theoretically is introduced. With the estimation of network status, the fairness model indicates the direction and size of delay adjustment, so that a fair share of bandwidth and low latency are guaranteed.

The paper is organized as follows: Section II introduces the related work at first. Section III highlights the motivation of this paper. Section IV describes our proposed Iris algorithm and the details of implementation are shown in Section V. Section VI shows the experimental results and corresponding analysis. Section VII concludes the paper.

## II. RELATED WORK

Conventional congestion control algorithms can be mainly divided into two categories: loss-based and delay-based. The loss-based methods, starting from Reno [29] and extending to Cubic [19] and Compound [30], interpret packet loss as the fundamental congestion signal. They continually push packets into the buffer of bottleneck link until packet loss occurs, resulting in “bufferbloat” and high queuing delay [12]. Besides, in wireless networks, low bandwidth utilization also results from the stochastic loss unrelated to congestion [31]. To address these issues, the algorithms like Vegas [20], FAST [32] and LEDBAT [33] use delay, rather than packet loss, as the congestion signal. They perform well in constraining queuing delay, but overestimate delay because of ACK compression or network jitter, resulting in inadequate

utilization of bandwidth [25]. Moreover, they will be starved when sharing a bottleneck link with concurrent loss-based flows [34]. Therefore, these conventional methods are not suitable for real-time video streaming.

There are also many studies focusing on special cases of network environments, including the algorithm customized for datacenters [35–37], cellular networks [38, 39], Web applications [40] and so on. These solutions yield good performance under the specific network conditions, but can not improve the performance of video streaming transmission. The algorithms specially designed for real-time video transmission mainly include [15, 21, 41]. They use packet loss and delay as congestion metrics, and empirically set some fixed thresholds. When the congestion metrics are higher or lower than these thresholds, the sending rate is adjusted accordingly. As we mentioned earlier, this hardwired control strategy can not maintain effectiveness in changing networks, resulting in performance degradation.

Over the past decade, more and more researchers have abandoned the TCP-like hardwired mapping rate control which maps events to fixed reactions (e.g. using fixed thresholds or step sizes). They prefer to generate effective control strategies through algorithms rather than handicraft, such as PCC [42], Verus [43], Remy [27] and Vivace [23]. Especially, Remy replaces the human designed algorithm with an offline optimization scheme that searches for the best scheme within its assumed network scenario. But the performance drastically degrades when the actual network conditions deviate from its assumptions [44]. PCC and Vivace propose to empirically observe and adopt actions that result in higher performance, but they have to try many times before deciding on the best action. The lag selection will affect the delay performance of real-time video streaming. Google proposes BBR [22] to address the limitations of conventional TCP, which tries to make the number of inflight packets (i.e. data sent but not yet acknowledged) converge to bandwidth-delay product (BDP). However, it has poor performance in TCP-fairness, and its throughput will fluctuate periodically and violently due to its synchronization mechanism, which destroys the smoothness of video bitrate [45].

In this paper, we propose a learning-based congestion control algorithm. Compared against these existing works, our innovations (and differences) lie in the following aspects:

- **Convergence objective.** Although we need to obtain delay information in Iris, its explicit convergence objective is the load of the queue, as described in Section IV-A2. This is different from the traditional delay-based algorithms in which explicit delay values are used as congestion benchmarks. *It can avoid the latecomer issue [33] of delay-based algorithms and improve fairness performance, but does not need to adopt hard synchronization like BBR [22].*
- **Environmental adaptation.** Iris introduced a rate adjustment method based on statistical learning, as described in Section IV-A4, to balance the generality and specificity of the algorithm. Based on the collected historical data, Iris can adaptively change the rate adjustment step size without setting specific parameters for each network.

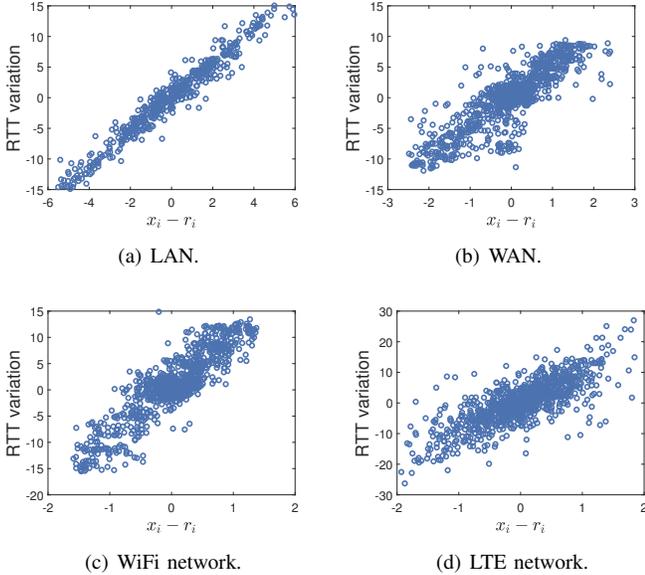


Fig. 2: The scatter plot of  $x - r$  (Mbps) and RTT variation (ms) under different networks.

- **Low overhead.** Different from the existing work that combines the concept of "learning", Iris uses a simple but effective linear regression learning model, which guarantees low overhead in complexity and time consumption. This approach avoids the drawbacks of Remy's offline learning due to its complexity [44], and is better at real-time than PCC and Vivace.

### III. DATA STUDY OF CONGESTED NETWORK

In order to understand the dynamics of the network, we generate packets to probe real-world networks, and explore the specific factors that directly affect the network state. To collect the network trace, we build a measurement testbed, consisting of an Ali Elastic Compute Service (ECS), a laptop and a Huawei P20 mobile phone. With UDP used in the transport layer, we tagged packets with sequence numbers and sender timestamp, and implemented ACK return for each packet in the application layer, to enable RTT and receiving rate calculation. The network trace is collected separately under LAN, WAN, WiFi and LTE networks, and the network operators are all China Mobile.

For collecting network trace, UDP packets are emitted with the fixed interval, a sending epoch of 100ms. Within  $i$ -th epoch, the sending rate  $x_i$  is constant. The corresponding RTT  $r_{tt_i}$  and receiving rate  $r_i$  is calculated from the ACKs of the all packets emitted in this epoch. The calculation details are shown in Section V-A.

#### A. Correlation Between Sending, Receiving Rate and RTT

We collected the trace under different networks, and extracted the sending rate, receiving rate and delay information for correlation analysis. Define  $\Delta r_{tt_i}$  as the RTT variation between two adjacent epochs:

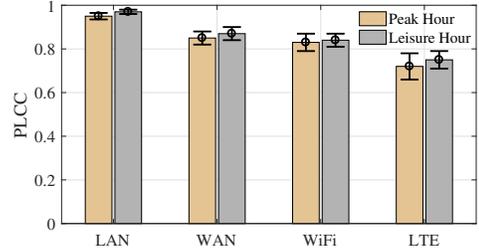


Fig. 3: Average PLCC of  $\Delta r_{tt}$  and  $x - r$  with 90% confidence interval in different real-world networks in leisure hours and peak hours.

$$\Delta r_{tt_i} = r_{tt_i} - r_{tt_{i-1}} \quad (1)$$

which represents the change of network congestion.

The correlation analysis is used to study the relationship among sending rate  $x$ , receiving rate  $r$  and RTT variation. Figure 2 displays the scatter plots of the traces under different networks, each of which contains more than 1500 data points. Intuitively, whether it is in WAN, LAN, WiFi or LTE networks, the difference between sending rate and receiving rate, i.e.  $x - r$ , always has a strong positive correlation with  $\Delta r_{tt}$ . Besides, the correlation is the strongest in LAN, while the correlation decreases slightly in LTE network because of more network jitter noise. This positive correlation indicates that, if the sending rate is larger than the receiving rate, the RTT in the network will also increase, which is not conducive to avoiding congestion.

#### B. Quantitative Correlation Study

To verify the correlation, we collected a large number of traces under different networks, respectively in leisure hours (i.e. 9:00 to 11:00) and peak hours (i.e. 19:00 to 21:00). For each trace, Pearson linear correlation coefficient (PLCC) is calculated for statistical analysis.

Figure 3 displays the PLCC of  $\Delta r_{tt}$  and  $x - r$  with 90% confidence interval under different networks in peak and leisure hours. We find that PLCC is the largest in LAN (up to 0.97), but the smallest in LTE networks (around 0.7), which is similar to the discovery in Section III-A. It is mainly because, compared to the simple network environment in the LAN, LTE networks have more noise and will weaken the correlation. Besides, PLCC in leisure hours is a little bigger than that during peak hours, resulting from the fact that the network load is heavier during the peak hour and will produce more noise.

In summary, PLCC is different under different networks and at different times. But even so, PLCC is still larger than 0.65 in the worst case, which is enough to prove the correlation. Therefore, the further verified conclusion is obtained that  $\Delta r_{tt}$  and  $x - r$  have strong linear correlation.

#### C. Linear Regression Learning

Due to the consistent high PLCC as shown in Figure 3, we believe the rate difference  $x - r$  and RTT variation are

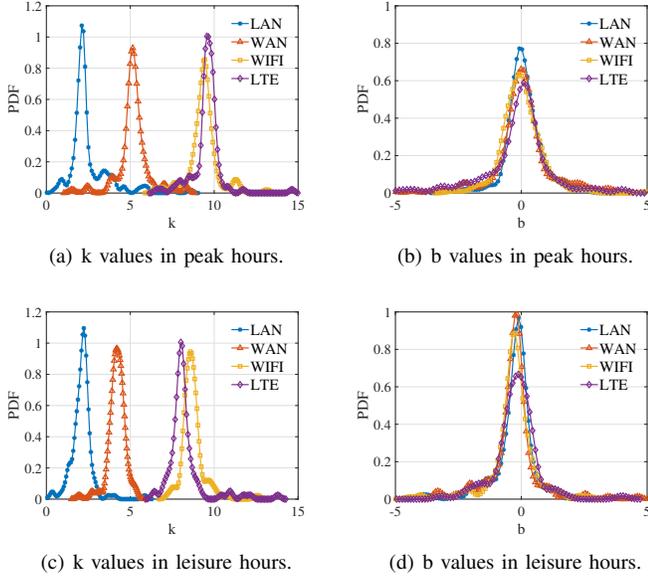


Fig. 4: The probability density function (PDF) of  $k$  and  $b$  values under different networks in leisure and peak hours.

always strongly linearly correlated. Therefore, the discovery motivates us to build the model expressing  $\Delta rtt_i$  as follows:

$$\Delta rtt_i = k \times (x_i - r_i) + b \quad (2)$$

where  $k$  indicates the extent to which  $\Delta rtt$  is affected by  $x - r$  and  $b$  is a bias. Since  $b$  can be regarded as a Gauss white noise, we use *maximum likelihood* to solve Equation (2) based on historical data [46].

This expression is logical from the view of congestion control. When sending rate exceeds receiving rate, the number of packets in the queue of bottleneck link will increase, resulting in the rise of queuing delay and RTT, and vice versa.

#### D. Distribution of $k$ and $b$

Furthermore, we study whether this function expression is constant in the real complex networks. Based on the collected traces, above maximum likelihood estimation is used to obtain the quantized values of  $k$  and  $b$ . Figure 4 displays the probability density function (PDF) of obtained  $k$  and  $b$  under different networks, respectively in leisure hours (i.e. 9:00 to 11:00) and peak hours (i.e. 19:00 to 21:00). It shows that the distributions of  $k$  and  $b$  are always approximate Gauss distribution regardless of the network type, which validates the applicability of the maximum likelihood method.

For further analysis, under different networks, the mean of  $k$  is obviously different, which is mainly related to the available bandwidth at that time. But as for  $b$ , the mean value is always near 0, no matter what type of network it is in. Although its variance is slightly different, we think of  $b$  as a Gauss noise, and it is usually small enough to be overlooked.

As shown in Figure 4(a) and Figure 4(c), although the distribution of  $k$  values is always Gaussian, its mean value has shifted markedly. That is to say, the distribution of  $k$  values

will change over time in the same network. Therefore, the way to generate a fixed function to adjust the sending rate is not optimal. Because when the network conditions change, the functional relationship between  $x - r$  and  $\Delta rtt$  will also change. It inspires us to design the step size adaptation mechanism as described in Section V-B.

#### E. Key Insights

Through the above correlation analysis and verification, we further emphasize the motivation of this paper into three aspects:

- An important discovery is observed, i.e. the difference between sending rate  $x$  and receiving rate  $r$  always has a strong correlation with RTT variation  $\Delta rtt$ .
- Through the analysis of correlation coefficient, the linear model as Equation (2) is established, which enlightens us on congestion control model.
- The values of  $k$  and  $b$  that represents the model parameters are various at different networks and different times, so a step size adaptation mechanism is necessary.

### IV. LEARNING BASED CONGESTION CONTROL

In this section, our Iris is introduced, which is a statistical learning based congestion control deeply inspired by our research on the correlation analysis in real-world networks.

#### A. Iris Algorithm

##### 1) Key Ideas:

Iris is designed as an efficient real-time congestion control, which mainly consists of two modules, a **low-latency fairness model** and a **learning-based rate adjustment**. The fairness model forces all flows to keep a small and constant number of packets queuing in network, i.e. *queue load*, so as to achieve fairness and low latency concurrently when there are multiple flows competing on a bottleneck link. The learning-based rate adjustment builds a statistical function between RTT and rate through online linear regression learning, then uses the model to determine the proper sending rate. It avoids fixed adjustment step size and converges to the fairness objective more quickly.

##### 2) Low-latency fairness Model:

As the core of Iris, the objective function with the ability to achieve low latency and fairness is designed, and its main component is the estimation of *queue load*.

As mentioned above,  $rtt_i$  represents the average RTT within the  $i$ -th epoch. We define  $T$  as target delay, which is set to the minimum RTT in a time window by default. So  $rtt_i - T$  represents the extra queuing delay and the number of extra packets in bottleneck queue within  $i$ -th epoch is calculated as  $x_i \cdot (rtt_i - T)$ . It represents the extra queue load estimation which indicates the extent of congestion. Therefore, the congestion can be avoided by maintaining it at a fixed range, i.e.  $x_i \cdot (rtt_i - T) = B$ , with  $B$  defined as the queue load target.

Therefore, our objective function can be expressed as:

$$U(x_i, rtt_i) = x_i \cdot (rtt_i - T) - B \quad (3)$$

where  $B$  represents the convergence status of Iris. Fewer redundant packets in the bottleneck queue helps Iris achieve high channel utilization and low latency at the same time. Thus it also tells the sender how to adjust its sending rate. When  $x_i \cdot (rtt_i - T)$  is higher than  $B$ , the network is considered to be congested, so the sending rate is supposed to be decreased, and vice versa. In addition, the fairness is guaranteed if each flow maintains the same number of packets filled into the bottleneck link.

### 3) Expected RTT Variation Adjustment:

The objective function in our *Low-latency Fairness Model* implies the extent of congestion. Combining the results of our previous correlation analysis, we designed the following strategies to adjust the expected next RTT Variation  $d_{i+1}$ : when  $U(x_i, rtt_i) < 0$ , the queue load does not reach our target. To improve bandwidth utilization, next RTT Variation  $d_{i+1}$  is expected to be greater than 0, because it usually indicates a higher sending rate. In the opposite case,  $d_{i+1}$  is expected to be less than 0. By adjusting the expected RTT step by step, Equation (3) will tend to be established. Therefore, we first simply define the expected RTT variation for the next epoch  $d_{i+1}$  as follows:

$$d_{i+1} = -U(x_i, rtt_i) \quad (4)$$

However, the real-world network has jitter noise so that the measurement of RTT is not completely accurate, which also leads to the existence of abnormal value of  $U(x_i, rtt_i)$ . To prevent over-adjustment, the activation function  $\tanh$  is introduced to limit the range of RTT variation, which is expressed as follows.

$$\tanh(U) = \frac{e^U - e^{-U}}{e^U + e^{-U}} \quad (5)$$

The main reason for using  $\tanh$  function is that it has an effective suppression on abnormal values. However, in the original  $\tanh$  function, the effective range of independent variables is only about  $[-1, 1]$ , while our objective function  $U$  can be as high as dozens. Therefore, in the process of implementation, we also need to stretch the effective scope of  $\tanh$  function through dividing  $U(x_i, rtt_i)$  by an expansion coefficient  $M$ . Finally,  $d_{i+1}$  can be expressed as follows.

$$d_{i+1} = -\delta \cdot \tanh\left(\frac{U(x_i, rtt_i)}{M}\right) \quad (6)$$

where  $\delta > 0$  represents the threshold of RTT variation.

By this method, the farther away from convergence target, the larger adjustment step size will be adopted. Besides, it helps to solve the problem of delay overestimation [25].

### 4) Learning-based Rate Adjustment:

According to the result of correlation analysis in Section III, the strong linear correlation between  $\Delta rtt$  and  $x - r$  is captured. Therefore, if the expected next RTT Variation  $d_{i+1}$  has been determined, the next sending rate  $x_{i+1}$  can be expressed as:

$$x_{i+1} = r_{i+1} + \frac{d_{i+1}}{k} \quad (7)$$

but when Iris makes the decision on  $x_{i+1}$  for the next epoch,  $r_{i+1}$  is unknown to the sender, so it is estimated by recently seen receiving rate in implementation, i.e.  $r_i$ , and  $k$  is learned from linear regression of historical data, determining the rate adjustment step size.

The design of Equation (7) is reasonable. When  $d_{i+1}$  is greater than 0, the channel is considered underutilized and we tend to fill more packets into the bottleneck queue, so the sending rate is supposed to be increased. On the contrary, if  $d_{i+1}$  is less than 0, it means that the number of packets in the queue exceeds the budget and is easy to cause congestion. In this case, sending rate is supposed to be adjusted to the decreasing direction.

## B. Fairness Analysis

Iris uses the low-latency fairness model, coupling with the learning-based rate adjustment, to guarantee high performance from the individual sender's perspective and ensure the convergence to a global fair rate allocation. Specifically, we consider a network model in which  $N$  flows compete on a bottleneck link with the bandwidth of  $C$  and the buffer is a FIFO queue.

**Theorem.** *When  $N$  Iris-senders share a congested bottleneck link and each sender uses the rate control mechanism as Equation (7), the senders' sending rates converge to a global fair configuration.*

*Proof.* We define that for a pair of senders  $a$  and  $b$ , within epoch  $i$ , their sending rates are respectively  $x_{i,a}$  and  $x_{i,b}$ . And  $x_{i,b}$  is larger than  $x_{i,a}$ . So, if the rate difference between any two flows decreases with time, the theorem is valid. It is equivalent to proving that

$$|x_{i+1,b} - x_{i+1,a}| < |x_{i,b} - x_{i,a}| \\ |r_{i,b} - r_{i,a} + \frac{d_{i+1,b}}{k} - \frac{d_{i+1,a}}{k}| < |x_{i,b} - x_{i,a}| \quad (8)$$

The bottleneck link is congested, i.e.  $\sum_{j \in N} x_{i,j} > C$ . Combined with the network link model, the difference in receiving rate can be expressed as  $r_{i,b} - r_{i,a} = \frac{C(x_{i,b} - x_{i,a})}{\sum_{j \in N} x_{i,j}}$ . Therefore, the following formula can be obtained.

$$0 < r_{i,b} - r_{i,a} < x_{i,b} - x_{i,a} \quad (9)$$

In order for the condition of Equation (8) to be satisfied, the following equation must be established.

$$0 < \frac{d_{i+1,a}}{k} - \frac{d_{i+1,b}}{k} < x_{i,b} - x_{i,a} \quad (10)$$

As the buffer is shared across all flows on the bottleneck link, they have the same queuing delay, i.e.  $q_i = rtt_{i,a} - T_{i,a} = rtt_{i,b} - T_{i,b}$ . And the  $\tanh$  function in Equation (6) is monotonically increasing, hence  $\frac{d_{i+1,a}}{k} > \frac{d_{i+1,b}}{k}$ . Defining  $U_i$  as  $\frac{x_i(rtt_i - T) - B}{M}$ , we can prove that

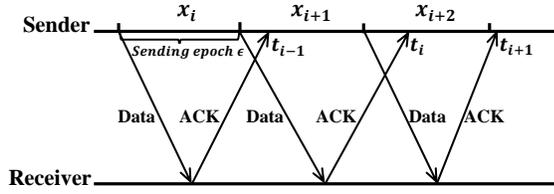


Fig. 5: A diagram for calculating the receiving rate.

$$\begin{aligned}
 \frac{1}{k}(d_{i+1,a} - d_{i+1,b}) &< x_{i,b} - x_{i,a} \\
 \frac{\delta}{k}(\tanh(U_{i,b}) - \tanh(U_{i,a})) &< x_{i,b} - x_{i,a} \\
 \frac{\delta}{k} \frac{\tanh(U_{i,b}) - \tanh(U_{i,a})}{U_{i,b} - U_{i,a}} \frac{U_{i,b} - U_{i,a}}{x_{i,b} - x_{i,a}} &< 1 \\
 \frac{\delta}{k} \frac{r_{tt_i} - T}{M} \frac{\tanh(U_{i,b}) - \tanh(U_{i,a})}{U_{i,b} - U_{i,a}} &< 1 \quad (11)
 \end{aligned}$$

where  $\tanh$  function has the largest slope at zero and its value is 1, so  $0 < \frac{\tanh(U_{i,b}) - \tanh(U_{i,a})}{U_{i,b} - U_{i,a}} < 1$ . And the value of  $M * k$  is usually much larger than  $\delta(r_{tt_i} - T)$ , as described in Section V-D. Thus the Equation (8) is established and the fair configuration is achieved.

## V. IMPLEMENT

### A. Receiving Rate Estimation

Considering that the channel unpredictably changes over time, especially in wireless networks, Iris adjusts sending rate in a small epoch of  $\epsilon$  ms to quickly adapt to the changing link. In order to obtain the receiving rate estimation, we first tagged packets with sequence numbers and sender's timestamp, and implemented ACK return for each data packet in the application layer, with UDP used in the transport layer.

In particular, our receiving rate is not obtained by the common calculation method. As shown in Figure 5, within  $i$ -th epoch, the total number of packets sent is  $x_i \cdot \epsilon$ .  $t_i$  represents the ACK return time of the last packet of this epoch, thus  $t_i - t_{i-1}$  represents the total time for the successful delivery of all packets in the  $i$ -th epoch. So the receiving rate estimate  $r_i$  of the  $i$ -th epoch is calculated at  $t_i$ , as follows.

$$r_i = \frac{x_i \cdot \epsilon}{t_i - t_{i-1}} \quad (12)$$

### B. Update $k$

As described in Section III-D, the  $k$  value reflects the relationship between  $\Delta r_{tt}$  and  $x - r$ , which directly determines the rate adjustment step size according to Equation (7). But it will change with the dynamics of the network. That is to say, a fixed  $k$  value can not be effective under various network conditions. Therefore, a periodic update mechanism for the  $k$  values is designed in Iris. To keep up with the network dynamics, the update cycle is set to 5 seconds empirically. Then, linear regression learning is used to fit  $k$  values periodically, based on the historical data.

TABLE I: Default parameter settings.

Parameter	Value
Epoch time $\epsilon$ ms	50
Scaling multiplier $M$ of $\tanh$	100
Extra queue load target $B$	10
RTT variance boundary $\delta$	3

### C. Cold Start

At the start-up of Iris, there is no data to support it to obtain the effective  $k$  value, and any handcrafted initial value is difficult to be robust to all kinds of networks with different capacities. Therefore, we adopt the similar control strategy as the slow-start stage of TCP to quickly perceive the link capacity and collect the training data in this cold start process.

After Iris starts, the initial sending rate of 100Kbps is first adopted, then it is doubled every epoch for updating the next sending rate. Once the packet loss rate is increased, Iris will exit from the startup phase and use the collected data to learn an initial  $k$  value. After that,  $k$  is periodically updated as described in Section V-B.

### D. Parameters Settings

Different parameters will directly affect the performance of iris. Unless stated otherwise, we implement Iris using the parameter default values in Table I.  $\epsilon$  represents the sending rate adjustment interval. The smaller  $\epsilon$  is more conducive to improving the adaptability of the algorithm, but will affect the accuracy of receiving rate estimation in a single epoch. Considering these two aspects, we empirically set  $\epsilon$  as 50ms based on a large number of experimental tests under different networks.  $M$  determines the effective limits of  $\tanh$  function and  $\delta$  limits the range of RTT variance, empirically set to 100 and 3 respectively, taking into account the applicability and generality for various networks.  $B$  is the extra queue load target and too large  $B$  will result in high self-inflicted delay, so we set the value of  $B$  to 10.

## VI. PERFORMANCE EVALUATION

In this section, we conducted extensive experiments to evaluate the performance of Iris, considering both the transport layer and the application layer. The experimental environment consists of a variety of networks, including emulated network, real-world Internet and commercial LTE networks.

### A. Testbed

In order to evaluate the performance of Iris in transport layer and network layer respectively, we implement it in UDT [47, 48] and QUIC [49]. The aim of QUIC implementation is to build an HTTP live streaming server, to evaluate the bitrate and PSNR gain of Iris in the application layer.

For transport layer testing, we implement a user-space prototype based on UDT. In this scenario, the comparison objects mainly include Sprout [38, 50], PCC (with latency utility function by default) [51], BBR and TCP variants (e.g. Cubic and Vegas). Two main metrics of the performance

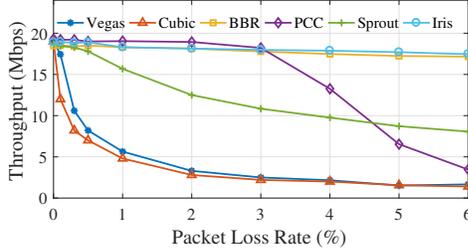


Fig. 6: Iris is robust to random packet loss.

are considered, namely throughput and delay characteristics. The testbeds in emulation and real-world environment are introduced below.

In the emulation environment, we design a dumbbell topology consisting of two nodes in the LAN, where one plays as sender and the other is receiver. The nodes are connected through Gigabit fiber and run a configurable number of sources with *Linux core 4.9*. For PCC and Sprout, the developers' implementations are used. We employ the *NetEm* linux module along with the traffic shaper *TC* to configure the link parameters, such as bottleneck bandwidth, propagation delay, packet loss and maximum queue size. Besides, *tcpdump* is used to capture packets to measure the metrics.

In the real-world environment, we mainly consider inter-continental Ethernet and LTE mobile network. In the case of inter-continental Ethernet, four Ali ECSs are employed to establish a node in Beijing, HongKong, Singapore and America respectively. Our host is used as sender while these nodes are regarded as receivers. As for LTE network, we employ a laptop connected with a 4G mobile phone to run these algorithms on China Mobile, a commercial LTE network. In order to evaluate the performance of these algorithms, a lot of tests have been carried out considering different time, different network access modes and other factors. The total time of collected traces is over 50 hours.

For the evaluation in application layer, in addition to the HTTP live streaming server over QUIC, we implement a video player based on the open source MPEG-DASH *dash.js* [52]. The QUIC server provides the video with five different bitrate versions (350Kbps, 620Kbps, 1.57Mbps, 2.54Mbps and 3.60Mbps), which are divided into segments with the same length (1 second), and the buffer length is set as 15 seconds. The traffic shaper *TC* is also used to configure the link parameters from server to client. When switching different algorithms in the congestion control module of QUIC, the performance of Iris can be evaluated based on the information output from the browser console.

## B. Evaluation in Emulated Networks

### 1) Robustness to Random Loss:

Lossy links in today's networks are not uncommon: wireless links are often unreliable and very long wired network paths can also have random loss caused by unreliable infrastructure. To further quantify the effect of random loss, we use *tc* to configure a bottleneck link with 20Mbps bandwidth, 50ms

round-trip propagation delay and varying loss rate, to evaluate the algorithms' robustness to random loss.

As Figure 6 shown, Cubic and Vegas perform badly when there is random packet loss. Even 0.5% of the random packet loss rate will reduce their channel utilization to less than 50%. And the performance of PCC will also drop sharply when the packet loss rate is greater than 3%. In contrast, our Iris and BBR are robust to random packet loss, which is because the packet loss is not considered as a congestion signal in the rate adjustment mechanism.

### 2) Tolerance to Long RTT:

Satellite Internet is widely used for critical missions such as emergency and military communication, which is challenging for congestion control because it has high propagation delay (RTprop), large bandwidth-delay product and random loss. Referring to the real-world measurement for the WINDs satellite Internet system [53], we evaluate Iris on an emulated link with 20 Mbps capacity, 1 BDP of buffer, 0.74% stochastic loss rate and changing RTprop.

Table II shows the average throughput and queuing delay of the protocols v.s. Iris. Compared against Iris, the throughput of Cubic and Vegas is less than 10%, due to the random packet loss in the link. PCC obtained the highest throughput, but at the cost of excessive latency. BBR is able to ignore random loss, but still underutilized the link as its rate oscillated wildly. In contrast, Iris achieves over 70% of optimal throughput at a time delay cost of only around 20 milliseconds. Although it is also affected by the high BDP, a good trade-off is achieved between throughput and delay.

TABLE II: Average throughput (Mbps) and queuing delay (ms) vs. Iris in emulated satellite link with high RTprop.

RTprop	600ms		800ms		1000ms	
	Rate	Delay	Rate	Delay	Rate	Delay
Country	16.7	16	15.6	19	13.9	21
Iris	0.03×	0.06×	0.02×	0.05×	0.03×	0.09×
Vegas	0.06×	0.12×	0.03×	0.15×	0.03×	0.19×
Cubic	0.71×	1.02×	0.65×	0.88×	0.51×	0.93×
BBR	1.12×	31.3×	1.24×	38.6×	1.34×	42.3×
PCC						

### 3) Responsiveness to Network Variation:

We next demonstrate how quickly Iris can adapt to dynamically changing network conditions. We start with a network employing *tc* where the bottleneck bandwidth changes every 40 seconds, with 50ms round-trip propagation delay and 10KB buffer. For each protocol, we repeat the test with 160 seconds duration.

Figure 7 illustrates the behavior of several of the protocols across time. Our Iris and PCC have almost the same throughput in the stable state, but when the bottleneck bandwidth changes, PCC's response is much slower. From startup to rate stability, PCC even takes ten seconds. This is mainly because PCC determines the direction of adjustment by several attempts before the sending rate is adjusted. As for Cubic, the higher capacity results in the larger rate jitter, due to the small buffer. BBR will periodically reduce its sending rate because of its time delay synchronization mechanism.

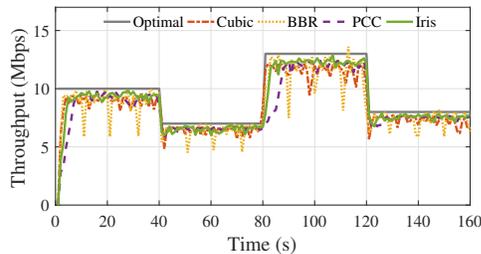


Fig. 7: Reaction to Changing Network.

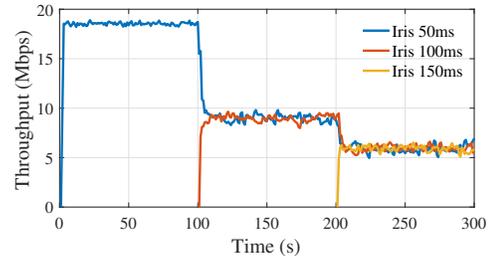


Fig. 9: The bandwidth allocation of three concurrent flows with different round-trip propagation delays.

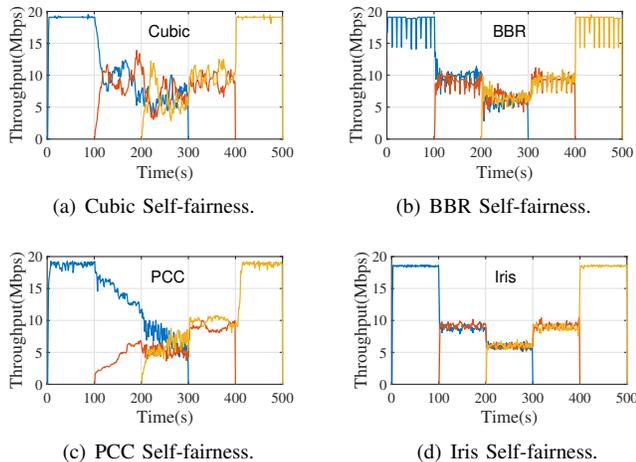


Fig. 8: Performance of intra-protocol fairness.

#### 4) Fairness Evaluation:

##### Intra-protocol fairness

We first evaluate the intra-protocol fairness of Cubic, BBR, PCC and our Iris separately. For this purpose, a dumbbell topology in the LAN is built to demonstrate their dynamic behavior with three flows sharing a bottleneck link with 20 Mbps bandwidth, 50 ms RTT and 0.5 BDP buffer.

Figure 8 shows the bandwidth allocation. Basically, Cubic, BBR, PCC and Iris all achieve a fair share of bandwidth. However, at the equilibrium point of fair bandwidth allocation, the throughput of each Cubic flow is severely jitter, which is not a stable convergence. As for BBR, the intra-protocol fairness is much better than Cubic, but there is a periodical sharp drop in throughput due to its synchronization mechanism for aligning the state of each flow. Although PCC can also share the bottleneck bandwidth fairly, its convergence time is too long, even tens of seconds, which is also confirmed in [22]. In contrast, our algorithm performs better either in terms of convergence speed or stability of convergence points.

##### RTT fairness

Typically, the flows sharing the same bottleneck link have different propagation delay. Ideally, they should get identical bandwidth allocation, but many algorithms exhibit significant RTT unfairness, disadvantaging flows with larger RTTs. To evaluate the performance of Iris in this area, we further consider an experiment where three competing Iris flows with three different propagation delays of 50 ms, 100 ms and 150 ms share a 20 Mbps bottleneck link.

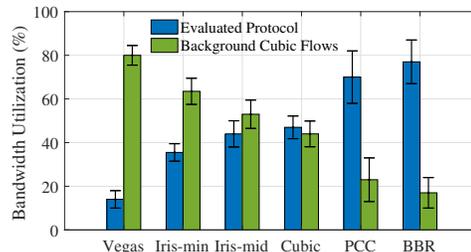


Fig. 10: Throughput of different algorithms versus Cubic, shown by plotting the mean and standard deviation of the bandwidth utilization.

Figure 9 shows the temporal variation in the throughput of the three different Iris flows. Intuitively, the throughput of Iris flows is independent of the propagation delays of the flows, so Figure 9 doesn't look very different from Figure 8(d). It is due to the effect of our proposed *Low-latency Fairness Model*, where the difference between  $d_i$  and  $T$  in Equation (3) can eliminate the effect of different propagation delays. It is queuing delay instead of RTT that can actually affect Iris algorithm to adjust the sending rate.

##### Fairness with TCP

Since most traffic in the Internet is still TCP-based (such as Cubic and Compound), if a congestion control algorithm is to be applied in real network environment, it is necessary to achieve a reasonable bandwidth allocation in the competition with TCP traffic. For this purpose, we use the same network as Section VI-B4 to carry out some experiments. We run a Cubic sender as background traffic and a sender of the congestion control algorithm being evaluated. These flows are run concurrently for 20 seconds.

Figure 10 shows the bandwidth utilization achieved by the evaluated algorithms versus the background Cubic traffic. Intuitively, Vegas is suffering from low throughput when competing against Cubic, which is just the weakness of most delay-based algorithms. On the contrary, PCC and BBR are too aggressive, which will seriously affect other flows on the Internet. As for Iris, even if we set target delay  $T$  to the minimum of RTTs (i.e. Iris-min), it can still obtain over 35% bandwidth allocation. This is because when Cubic accumulates packets to increase RTT, the minimum value Iris sees in the time window will also increase, which will improve its convergence target. If we use the median of RTTs

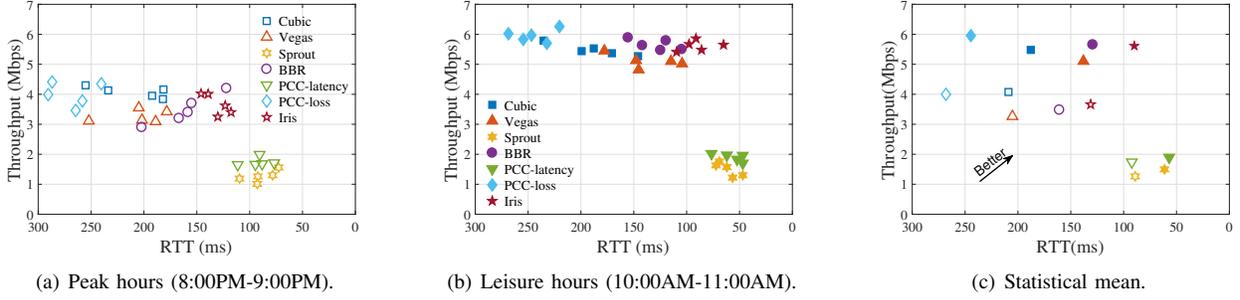


Fig. 11: Throughput and RTT in LTE networks of stationary scenario, five tests of 60s for each algorithm.

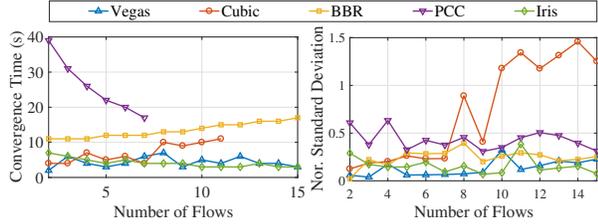


Fig. 12: Convergence speed and throughput stability in different concurrent numbers.

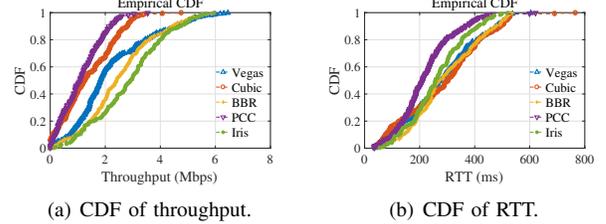


Fig. 13: Performance in LTE networks of moving scenario.

as the target delay  $T$  (i.e. Iris-mid), the bandwidth allocation will be closer to the ground truth (i.e. Cubic v.s. Cubic) [26].

### 5) Convergence Speed and Stability:

When the network environment changes, the convergence speed and stability of the protocol determine the performance of throughput and delay. In order to measure these two aspects quantitatively, the following experiments are designed in this paper. On a link of 50 Mbps bandwidth and 50 ms RTT, for each protocol, we start 15 flows one by one at intervals of 50 seconds, which is sufficient for most of the protocols to achieve fair convergence. The convergence speed is measured by the time required for convergence, which is calculated as the time from the newer flows entry to the earliest time after which Jain’s fairness index is maintained over 0.9 for at least 5 seconds. And the stability is estimated as the average standard deviation of throughput of all flows after the convergence point is reached.

Figure 12 displays the convergence time and stability as the number of flows increases. From the view of convergence speed and the stability after convergence, our Iris and Vegas have relatively close performance, which is much better than other algorithms. It is worth noting that when the number of flows exceeds 11, Cubic can no longer achieve a fair convergence, whose bandwidth utilization will also decrease dramatically. It is so-called “TCP Incast” phenomenon [54]. As for PCC, this threshold is even 7. Although BBR is able to achieve fair convergence, its convergence time will continue to rise with the increase of the number of flows.

## C. Evaluation in Real-World LTE Networks

### 1) Stationary Scenario:

In the real-world LTE network, an ideal congestion control algorithm can achieve high throughput and low latency at the same time. We first test the protocols in peak hours and leisure hours to evaluate the performance in stationary scenes. In addition, we evaluated PCC using loss utility function (PCC-loss) and delay utility function (PCC-latency) respectively. This evaluation was carried out in Peking University and connected to the commercial network operator, China Mobile.

Figure 11 displays the test results of average throughput and RTT. Intuitively, for all algorithms, the throughput in peak hours is smaller and RTT is larger. For PCC, the loss utility function makes it fill the buffer as much as possible, resulting in extremely high delay. But the utility function that tends to low delay will make it unable to get high bandwidth allocation, because its utility function can not correctly determine the direction of speed adjustment in highly variable networks. For Iris, although the throughput is not the highest, it can effectively balance throughput and delay, which is more important for real-time congestion control.

### 2) Motion Scenario:

Furthermore, we evaluated the performance of the protocols on LTE networks in motion scenarios. Note that the subway was chosen as the test scene, mainly because there is no influence from the multi base station switching in the tunnel, so the environment is more simple. On line four of Beijing Subway, from East Gate of Peking University to Yuanmingyuan Park Station, each protocol has been tested ten times.

Figure 13 shows the throughput CDF of these protocols. Different with the stationary scenario, we find that the bottleneck bandwidth of the network is closely related to the

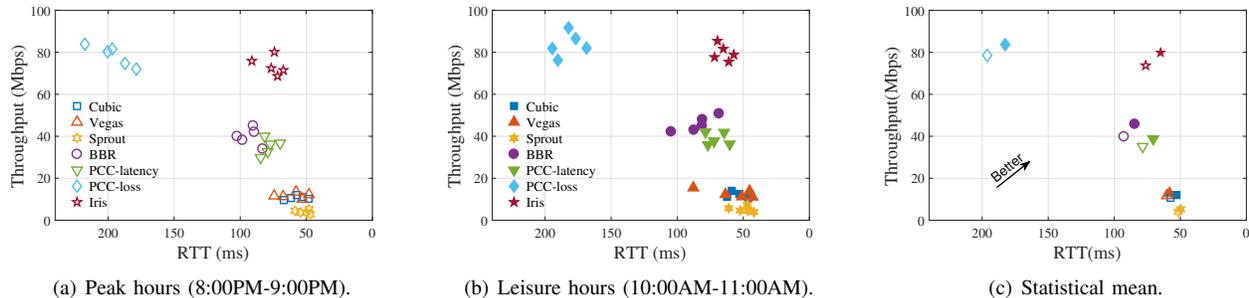


Fig. 14: Throughput vs. RTT from our host in Beijing to Ali ECS in HongKong, five tests of 60s for each algorithm.

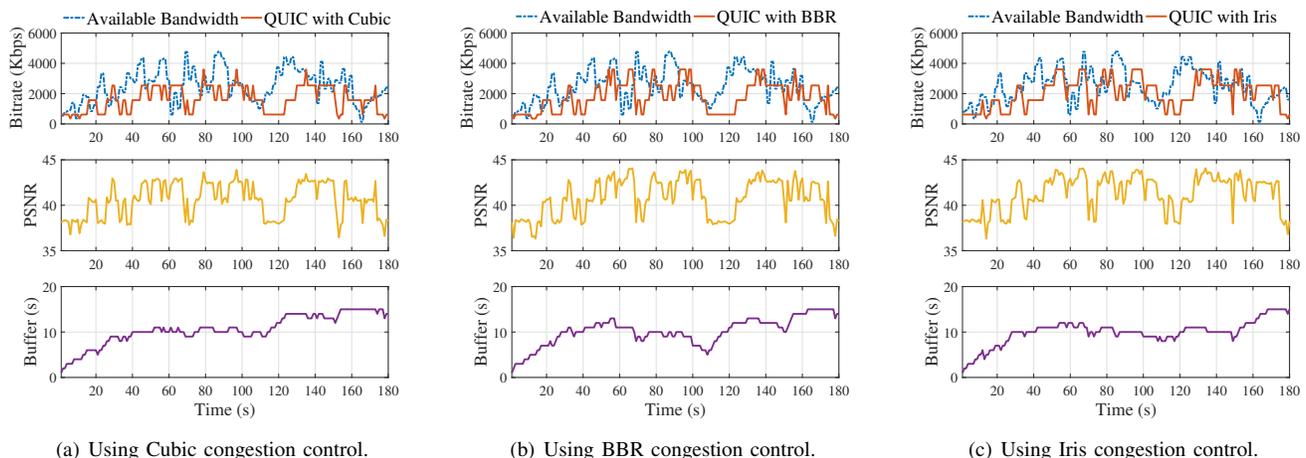


Fig. 15: The bitrate, PSNR and buffer occupancy during video playback when using different congestion control algorithms.

TABLE III: Average throughput (Mbps) and RTT (ms) vs. Iris on the link from Beijing to the Ali ECS Beijing, HongKong, Singapore and America.

Link	BJ → BJ		BJ → HK		BJ → SG		BJ → USA	
	Rate	RTT	Rate	RTT	Rate	RTT	Rate	RTT
Iris	83.6	28	76.2	66	62.2	144	38.4	201
Vegas	0.52×	0.78×	0.19×	0.79×	0.22×	0.93×	0.21×	0.93×
Cubic	0.48×	0.80×	0.20×	0.74×	0.27×	0.93×	0.19×	0.93×
BBR	0.73×	1.07×	0.56×	1.11×	0.39×	0.99×	0.37×	0.97×
Sprout	0.05×	0.36×	0.06×	0.68×	0.06×	0.90×	0.06×	0.91×
PCC-Loss	1.08×	5.85×	1.07×	2.85×	0.90×	1.31×	1.06×	1.17×
PCC-Latency	0.69×	0.96×	0.49×	1.08×	0.41×	1.01×	0.32×	0.94×

location of the train when it is tested along the subway. Therefore, the measured throughput and RTT are approximately linear. In a comprehensive view, our Iris outperforms other protocols, achieving similar RTT but higher throughput. Facing more complex networks and more physical link loss in the high-speed motion scenarios, Cubic and Vegas can no longer achieve high throughput.

#### D. Evaluation in Intercontinental Networks

To evaluate the protocols in the intercontinental networks, we employ Ali ECS to establish four nodes in Beijing, HongKong, Singapore and America respectively. Our host in Beijing is used as a sender while these nodes are regarded as receivers, forming four links.

Taking the link from the host in Beijing to Ali ECS in HongKong as an example, Figure 14 shows throughput and RTT of the protocols, in peak hours and leisure hours respectively. Intuitively, our Iris algorithm outperforms others. Different from the results in Figure 11, the bandwidth utilization of Cubic and Vegas has decreased significantly, while the throughput of PCC has increased a lot, mainly because the complex background traffic often brings packet loss, but delay jitter is more gentle than LTE network. Our Iris is more competitive in the competition scenario, so it can still maintain high throughput, no matter at peak or leisure hours. In addition, we use the same method to evaluate the algorithms on the other three links. Table III displays the average throughput and RTT of these protocols vs. Iris, which further proves the excellence of Iris.

#### E. Performance in HTTP over QUIC

To evaluate the gain of Iris congestion control algorithm for Internet video transmission, we built an HTTP live streaming system with QUIC server [55] and MPEG-DASH *dash.js* [52]. Notice that the adaptive bitrate (ABR) algorithm we used in *dash.js* is a simple method based on bandwidth estimation.

As depicted in Figure 15, we compare our Iris with Cubic and BBR under a real Internet bandwidth trace (about 180 seconds), where both the long-term shifts and short-term

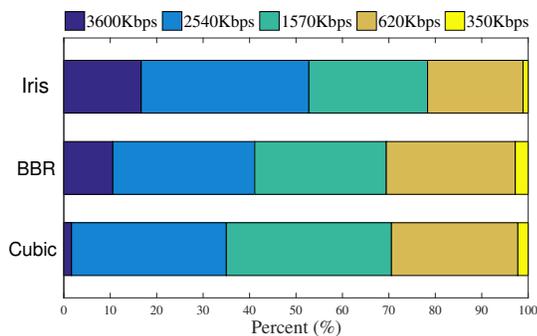


Fig. 16: The proportion of video bitrate level.

TABLE IV: Performance under the real Internet trace

Protocol	Average PSNR (dB)	Average bitrate (Kbps)
Cubic	40.536	1624.4
BBR	40.712	1763.5
Iris	41.27	2033.8

fluctuations of bandwidth can be observed. Besides, the round-trip propagation delay is set to 50ms and random loss is 1%. Due to the conservative bandwidth-based adaption logic, there is no stalling phenomenon in the process of video playback. However, because of the low bandwidth utilization of Cubic congestion control in this case, the client rarely requests the highest bitrate video segments, as shown in Figure 15. And the client with BBR algorithm has higher instability in bitrate, compared with Iris.

In order to show the difference more clearly, we summarize the bitrate proportion of the requested video segments, as depicted in Figure 16. It demonstrates that more than 50% of the video segments requested by Iris are 3600Kbps or 2540Kbps, while this percentage is only about 35% for Cubic. Quantitatively, as shown in Table IV, Iris achieves a 25% bitrate improvement over Cubic and 15% over BBR. And the average PSNR of Iris is also the highest, compared against Cubic and BBR. These results indicate that replacing the congestion control module in QUIC with our proposed Iris is able to achieve higher user experience quality.

## VII. CONCLUSIONS

In this paper, we have designed Iris, an end-to-end statistical learning based congestion control algorithm. The key ideas of Iris are keeping a small and fixed *queue load* in networks, and adaptively adjusting sending rate based on a linear-regression learning model. By forcing all flows to maintain the same queue load, a fair share of bandwidth and low latency are both achieved. The rate can be periodically updated by online regression learning, which avoids hardwired rate adjustment to better adapt to dynamically changing networks. Extensive experiments are carried out to evaluate the performance of Iris. It shows that, in various network environments, Iris is able to achieve high bandwidth utilization and low latency at the same time, and outperforms most existing algorithms. It also improves Internet video services in the application layer,

increasing the bitrate by 25% compared against Cubic, under the real network trace.

## ACKNOWLEDGMENT

The authors would like to thank all reviewers for providing many constructive suggestions which will significantly improve the presentation of this paper.

## REFERENCES

- [1] M. Shafi, A. F. Molisch, P. J. Smith, T. Haustein, P. Zhu, P. Silva, F. Tufvesson, A. Benjebbour, and G. Wunder, "5G: A tutorial overview of standards, trials, challenges, deployment, and practice," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 6, pp. 1201–1221, 2017.
- [2] X. Zhang, Y. Xu, H. Hu, Y. Liu, Z. Guo, and Y. Wang, "Modeling and analysis of skype video calls: Rate control and video quality," *IEEE Transactions on Multimedia*, vol. 15, no. 6, pp. 1446–1457, Oct 2013.
- [3] M. Ghobadi, Y. Cheng, A. Jain, and M. Mathis, "Trickle: Rate limiting YouTube video streaming," *Proc Usenix Atc*, pp. 17–17, 2012.
- [4] R. Shea, J. Liu, C. H. Ngai, and Y. Cui, "Cloud gaming: architecture and performance," *IEEE Network*, vol. 27, no. 4, pp. 16–21, 2013.
- [5] K. Chen, Y. Chang, H. Hsu, D. Chen, C. Huang, and C. Hsu, "On the quality of service of cloud gaming systems," *IEEE Transactions on Multimedia*, vol. 16, no. 2, pp. 480–495, Feb 2014.
- [6] J. Huang, F. Qian, Y. Guo, Y. Zhou, Q. Xu, Z. M. Mao, S. Sen, and O. Spatscheck, "An in-depth study of LTE: effect of network protocol and application behavior on performance," in *ACM SIGCOMM 2013 Conference on SIGCOMM*, 2013, pp. 363–374.
- [7] A. Gurtov and S. Floyd, "Modeling wireless links for transport protocols," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 85–96, 2004.
- [8] C. Zhou, C. Lin, and Z. Guo, "mdash: A markov decision-based rate adaptation approach for dynamic http streaming," *IEEE Transactions on Multimedia*, vol. 18, no. 4, pp. 738–751, April 2016.
- [9] S. Kim and C. Kim, "Xmas: An efficient mobile adaptive streaming scheme based on traffic shaping," *IEEE Transactions on Multimedia*, vol. 21, no. 2, pp. 442–456, Feb 2019.
- [10] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '17. New York, NY, USA: ACM, 2017, pp. 197–210. [Online]. Available: <http://doi.acm.org/10.1145/3098822.3098843>
- [11] M. Aguayo, L. Bellido, C. M. Lentisco, and E. Pastor, "Dash adaptation algorithm based on adaptive forgetting factor estimation," *IEEE Transactions on Multimedia*, vol. 20, no. 5, pp. 1224–1232, May 2018.
- [12] J. Gettys and K. Nichols, "Bufferbloat: dark buffers in the internet," *Communications of The ACM*, vol. 55, no. 1, pp. 57–65, 2012.

- [13] G. Bakar, R. A. Kirmiziloglu, and A. M. Tekalp, "Motion-based rate adaptation in webrtc videoconferencing using scalable video coding," *IEEE Transactions on Multimedia*, vol. 21, no. 2, pp. 429–441, Feb 2019.
- [14] J. Wu, R. Tan, and M. Wang, "Energy-efficient multipath tcp for quality-guaranteed video over heterogeneous wireless networks," *IEEE Transactions on Multimedia*, pp. 1–1, 2018.
- [15] C. Greco, M. Cagnazzo, and B. Pesquet-Popescu, "Low-latency video streaming with congestion control in mobile ad-hoc networks," *IEEE Transactions on Multimedia*, vol. 14, no. 4, pp. 1337–1350, Aug 2012.
- [16] H. Shiang and M. van der Schaar, "A quality-centric tcp-friendly congestion control for multimedia transmission," *IEEE Transactions on Multimedia*, vol. 14, no. 3, pp. 896–909, June 2012.
- [17] M. H. Hajiesmaili, L. T. Mak, Z. Wang, C. Wu, M. Chen, and A. Khonsari, "Cost-effective low-delay design for multiparty cloud video conferencing," *IEEE Transactions on Multimedia*, vol. 19, no. 12, pp. 2760–2774, Dec 2017.
- [18] Y. Hu, D. Niu, and Z. Li, "A geometric approach to server selection for interactive video streaming," *IEEE Transactions on Multimedia*, vol. 18, no. 5, pp. 840–851, May 2016.
- [19] S. Ha, I. Rhee, and L. Xu, "Cubic: a new TCP-friendly high-speed TCP variant," *Operating Systems Review*, vol. 42, no. 5, pp. 64–74, 2008.
- [20] L. SBrakmo and L. LPeterson, "TCP Vegas: end to end congestion avoidance on a global internet," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 8, pp. 1465–1480, 1995.
- [21] G. Carlucci, L. De Cicco, S. Holmer, and S. Mascolo, "Analysis and design of the google congestion control for web real-time communication (webrtc)," in *Proceedings of the 7th International Conference on Multimedia Systems*. ACM, 2016, p. 13.
- [22] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: Congestion-based congestion control," *ACM Queue*, vol. 14, no. 5, p. 50, 2016.
- [23] D. M. M. T. and e. a. Zarchy D, "PCC vivace: Online-learning congestion control," in *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*. USENIX Association, 2018.
- [24] O. Habachi, H. Shiang, M. van der Schaar, and Y. Hayel, "Online learning based congestion control for adaptive multimedia transmission," *IEEE Transactions on Signal Processing*, vol. 61, no. 6, pp. 1460–1469, March 2013.
- [25] K. N. Srijith, L. Jacob, and A. L. Ananda, "TCP vegas-a: Improving the performance of TCP vegas," *Computer Communications*, vol. 28, no. 4, pp. 429–440, 2005.
- [26] L. Cai, , and J. W. Mark, "Performance analysis of tcp-friendly aimd algorithms for multimedia applications," *IEEE Transactions on Multimedia*, vol. 7, no. 2, pp. 339–355, April 2005.
- [27] K. Winstein and H. Balakrishnan, "Tcp ex machina: Computer-generated congestion control," in *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4. ACM, 2013, pp. 123–134.
- [28] A. Sivaraman, K. Winstein, P. Thaker, and H. Balakrishnan, "An experimental study of the learnability of congestion control," in *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4. ACM, 2014, pp. 479–490.
- [29] J. Padhye, V. Firoiu, D. F. Towsley, and J. F. Kurose, "Modeling TCP reno performance: a simple model and its empirical validation," *IEEE/ACM Transactions on Networking*, vol. 8, no. 2, pp. 133–145, 2000.
- [30] K. Tan, "Compound TCP: A scalable and tcp-friendly congestion control for high-speed networks," *Pfldnet Feb*, 2006.
- [31] G. Xylomenos, G. C. Polyzos, P. Mahonen, and M. Saaranen, "TCP performance issues over wireless links," *IEEE Communications Magazine*, vol. 39, no. 4, pp. 52–58, 2001.
- [32] D. X. Wei, C. Jin, S. H. Low, and S. Hegde, "FAST TCP: motivation, architecture, algorithms, performance," *IEEE/ACM Transactions on Networking*, vol. 14, no. 6, pp. 1246–1259, 2006.
- [33] D. Rossi, C. Testa, S. Valenti, and L. Muscariello, "LED-BAT: The new bittorrent congestion control protocol," in *International Conference on Computer Communications and Networks*, 2010.
- [34] L. A. Grieco and S. Mascolo, *Performance evaluation and comparison of Westwood+, New Reno, and Vegas TCP congestion control*, 2004.
- [35] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data center TCP (DCTCP)," *ACM Sigcomm Computer Communication Review*, vol. 40, no. 4, pp. 63–74, 2010.
- [36] M. Alizadeh, A. Kabbani, T. Edsall, B. Prabhakar, A. Vahdat, and M. Yasuda, "Less is more: trading a little bandwidth for ultra-low latency in the data center," in *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*. USENIX Association, 2012, pp. 19–19.
- [37] M. Alizadeh, S. Yang, M. Sharif, S. Katti, N. McKeown, B. Prabhakar, and S. Shenker, "pfabric: Minimal near-optimal datacenter transport," in *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4. ACM, 2013, pp. 435–446.
- [38] K. Winstein, A. Sivaraman, H. Balakrishnan *et al.*, "Stochastic forecasts achieve high throughput and low delay over cellular networks." in *NSDI*, 2013, pp. 459–471.
- [39] Y. Zaki, T. Pötsch, J. Chen, L. Subramanian, and C. Görg, "Adaptive congestion control for unpredictable cellular networks," in *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4. ACM, 2015, pp. 509–522.
- [40] T. Flach, N. Dukkipati, A. Terzis, B. Raghavan, N. Cardwell, Y. Cheng, A. Jain, S. Hao, E. Katz-Bassett, and R. Govindan, "Reducing web latency: the virtue of gentle aggression," in *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4. ACM, 2013, pp. 159–170.

- [41] X. Zhu, R. Pan, M. Ramalho, S. de la Cruz, C. Ganzhorn, P. Jones, and S. DAronco, “Nada: A unified congestion control scheme for real-time media,” *Draft IETF*, Mar, 2013.
- [42] M. Dong, Q. Li, D. Zarchy, P. B. Godfrey, and M. Schapira, “PCC: Re-architecting congestion control for consistent high performance.” in *NSDI*, 2015, pp. 395–408.
- [43] Y. Zaki, T. Pötsch, J. Chen, L. Subramanian, and C. Görg, “Adaptive congestion control for unpredictable cellular networks,” *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 5, pp. 509–522, 2015.
- [44] A. Sivaraman, K. Winstein, P. Thaker, and H. Balakrishnan, “An experimental study of the learnability of congestion control,” in *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4. ACM, 2014, pp. 479–490.
- [45] T. Dai, X. Zhang, and Z. Guo, “Analysis and experimental investigation of BBR,” in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2018, pp. 1–2.
- [46] L. Fahrmeir and H. Kaufmann, “Consistency and asymptotic normality of the maximum likelihood estimator in generalized linear models,” *The Annals of Statistics*, pp. 342–368, 1985.
- [47] Y. Gu and R. L. Grossman, “UDT: UDP-based data transfer for high-speed wide area networks,” *Computer Networks*, vol. 51, no. 7, pp. 1777–1799, 2007.
- [48] “UDT: Breaking the data transfer bottleneck,” available online: <http://udt.sourceforge.net/>.
- [49] A. Langley, J. Iyengar, J. Bailey, J. Dorfman, and I. Swett, “The QUIC transport protocol: Design and internet-scale deployment,” in *the Conference of the ACM Special Interest Group*, 2017.
- [50] “Sprout user-space implementation,” available online: <http://alfalfa.mit.edu/>.
- [51] “PCC user-space implementation,” available online: <https://github.com/modong/pcc>.
- [52] D. I. Forum, “dash.js,” available online: <http://github.com/DASH-Industry-Forum/dash.js>.
- [53] H. Obata, K. Tamehiro, and K. Ishida, “Experimental evaluation of TCP-STAR for satellite internet over WINDS,” in *2011 Tenth International Symposium on Autonomous Decentralized Systems (ISADS)*. IEEE, 2011, pp. 605–610.
- [54] Y. Chen, R. Griffith, J. Liu, R. H. Katz, and A. D. Joseph, “Understanding TCP incast throughput collapse in datacenter networks,” in *Proceedings of the 1st ACM workshop on Research on enterprise networking*. ACM, 2009, pp. 73–82.
- [55] “QUIC, a multiplexed stream transport over udp,” available online: <https://www.chromium.org/quic>.