Analytical Framework for Streaming over TCP

Jinyao Yan*†, Wolfgang Mühlbauer*, Bernhard Plattner*
*Computer Engineering and Networks Laboratory, ETH Zurich, CH-8092, Switzerland
†Computer and Network Center, Communication University of China, Beijing 100024, China

***Abstract*—Multimedia streaming applications are traditionally delivered over UDP. Recent measurements show that more and more multimedia streaming data are over TCP as web-based TV, P2P streaming, video sharing websites are getting increasingly popular. To improve the Quality of Experience (QoE) for users and to cope with variability in TCP throughput, streaming applications typically implement buffers. Yet, for improving the QoE and the streaming quality, e.g., playback continuity and timeliness, it is critical to dimension buffers and the initial buffering delay appropriately.**

In this paper, we first develop a model for TCP streaming systems and an analytical framework to assess the QoE. Our emphasis is on buffer occupancy, which depends on the TCP arriving rate and the playout rate (the coding rate). We observe that TCP window "bounds", namely congestion window sizes immediately before a triple duplicate or timeout event, allow to distinguish the minimum and maximum buffer occupancy for TCP streaming systems. As confirmed by experiments, the proposed analytical framework allows to estimate the frequency of buffer overflow or underflow events if buffer sizes and the initial buffering delays are known parameters, or conversely, to dimension the buffer and delay appropriately.

We further extend our model and analysis for P2P multicast streaming systems. Simulations and experiments in real networks validate our proposed analytical framework in terms of underflow/overflow probabilities and delay.

***Index Terms*—HTTP/TCP streaming, overflow, underflow, delay, buffer size, Quality of Experience (QoE)**

I. Introduction

Conventionally, multimedia streaming applications in the Internet run over UDP instead of TCP due to the following reasons. First, the additive increase multiplicative decrease (AIMD) congestion control algorithm of TCP leads to a varying throughput for real-time streams, which severely impacts the video quality perceived by the end user. Second, TCP applies packet retransmission to guarantee the delivery of packets. But ultimately, such packets may arrive too late for playback and therefore be useless. The real-time transport protocol (RTP) [1], therefore, was designed for streaming multimedia on top of UDP to meet a lot of functionality for multimedia applications including communicating the choice of coding scheme, the timing relationship and synchronization among the sent/received data, packet loss indication, etc. RTP has been used extensively in communication and entertainment systems such as VOIP and video teleconference applications.

For UDP-based multimedia applications, a main concern in the design of transmission control algorithms has been the coexistence of traffic with TCP traffic. Indeed, without congestion control, non-TCP traffic can cause starvation of TCP traffic if both types of traffic compete for resources at a congested FIFO queue [2]. Among the principal representatives of the TCP-friendly congestion control algorithms is the TCP-friendly rate control (TFRC) algorithm. TFRC is an equation-based congestion control mechanism over UDP that uses the TCP throughput function of the measured rate of loss events derived in [3] to calculate the actual available rate for a media stream. TFRC was adopted as a profile in the Datagram Congestion Control Protocol (DCCP) [4]. However, TFRC and other TCP friendly UDP algorithms are not yet widely used in today's Internet probably due to some limitations reported in [5] [6].

Recent measurements [7][8] show that streaming over TCP is becoming practical and popular as applications such as web-based TV, P2P streaming, video sharing websites and social media systems are rapidly adopted in the Internet. The trend of TCP streaming is mainly due to the fact that the deployment and the use of TCP/HTTP multimedia applications is easier than relying on UDP based multimedia applications given the wide use of Network Address Translation (NAT) and firewalls. For example, Adobe HTTP Dynamic Streaming[1] over flash platform is becoming one of the largest categories of video traffic according to Cisco's VNI report[9]. It enables the standard web servers to deliver on-demand and live streaming of multimedia over regular HTTP connections.

The throughput and performance of multimedia streaming (but also other) applications strongly depend on the AIMD congestion behavior of the underlying TCP [10] [11] when reacting the packet loss and delay jitter in the Internet. In general, today's Internet only provides a best-effort service and it does not provide quality of service (QoS) or guarantees for multimedia streaming applications. Network conditions such as available bandwidth, packet loss rate, delay and delay jitter vary over time. To improve the QoE and to cope with this variability in TCP data throughput, streaming applications typically use buffers. While QoE is a subjective measure from the users' perspective of the overall service quality, we assess QoE and the quality of IPTV service in terms of the buffer

Report published first in Dec. 2010, updated in Jan. 2012.

[1]http://www.adobe.com/products/httpdynamicstreaming/

delay, playback jitter (buffer underflow and overflow) and video quality in this paper. When TCP packets arrive, they are inserted into a buffer. To achieve a steady playout, packets are removed from the buffer at a constant rate. However, the capacity of devices used in the Internet ranges from high performance computers to low memory and performance mobile nodes. Ideally, both the initial buffer delay and the size of the buffers should be as small as possible, yet big enough to avoid buffer underflow and overflow events. Playback stops when an underflow occurs due to packets arriving late. When an overflow occurs, video is discontinuous as packets are skipped and lost. An empirical evaluation in [12] for commercial multimedia software such as Skype, Google Talk and MSN Messenger shows that these applications do not dimension and adjust their buffer sizes very well. Proper buffer and initial buffer delay dimensioning algorithms and analysis are needed to provide better QoE.

In this paper, we propose a framework to analyze the probability of discontinuing playback and delay for any stream[2] under the conditions of the dynamics of the Internet, the variability of TCP throughput and the streaming buffer size. Guided by our analytical framework for TCP streaming, a Skype video conference over TCP, for example, may trade video quality for playback continuity and low delay. Namely, Skype may scale down the video quality to a certain rate much lower than the TCP average available rate for the desired and less overflow/underflow events while better meeting the real time delay constraints[3].

Figure 2 illustrates the relationship between TCP window size (namely sending/arriving packets in a round-trip time), buffer occupancy, and the playout rate in our analytical framework. The evolution of the congestion window size of a TCP connection is shown in Figure 1. Interestingly, we observe that the concept of "bounds" on TCP congestion window sizes, i.e., the window sizes immediately before a triple duplicate event[4], allow to distinguish between three cases with respect to the times of minimum and maximum buffer occupancy within a triple duplicate period (TDP).

To summarize our contributions, we first develop a TCP streaming model and analytical framework with emphasis on relationship between TCP arriving rate (namely TCP window size number of packets per RTT), buffer occupancy, and the playout rate (the encoding rate). In our previous work [13], we developed a TCP window "bounds" model tailored to streaming applications. We find that TCP window "bounds" aligned with the classification of Figure 2 allow to describe the relationship among buffer sizes, the initial buffer delay and overflow/underflow probabilities in an elegant and simple manner.

Secondly, we show that, given desired underflow/overflow probabilities, we can determine appropriate buffer sizes and the initial buffer delay based on our framework by using the distribution of TCP window bounds. Or, conversely, it is possible to estimate underflow/overflow probabilities if buffer sizes and the initial buffer delay are given.

Finally, we extend our TCP streaming model and analytical framework for P2P multicast streaming systems.

We verify our models and analysis both in simulated environments and in the real Internet with our P2P multicast streaming system.

The rest of the paper is organized as follows. Section II reviews related work and highlights the novel aspects of this work compared to previous work. In Section III, we present our system model, define the research problems and QoE metrics for TCP-based streaming. We further characterize buffer occupancy within a TDP. We scrutinize the relationship between buffer sizes and underflow/overflow probabilities (Section IV) in various streaming cases. We extend our model and analysis from unicast to P2P multicast streaming systems in Section V. We evaluate the buffer performance in terms of underflow/overflow probabilities and delay by simulations and experiments in real networks in Section VI. Finally, we conclude the paper in Section VII. For completeness, we present the model and the solution for the distribution of bounds on TCP congestion window sizes in Appendix A.

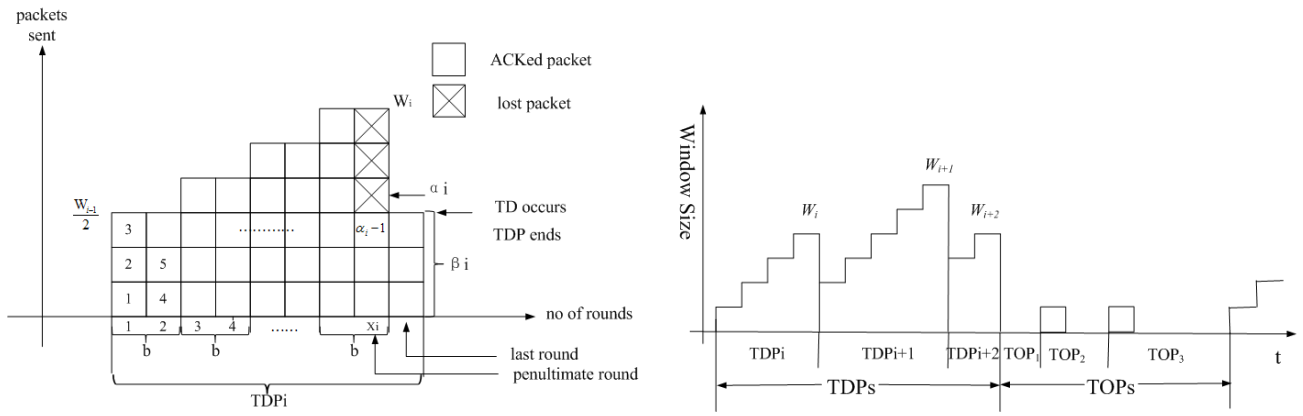
II. Related work

Researchers have studied the performance of TCP streaming with infinite buffers in [14] and with finite buffers in [15]. Work reported in [14] suggests that TCP generally provides good streaming performance when the achievable TCP throughput is roughly twice the media bit-rate with only a few seconds of startup delay. The performance is measured by late packets in [14]. In [16], buffer and delay performance of streaming over random VBR channels was studied. Random VBR channels in [16] are channels with statistical properties such as wireless access networks modeled by an extended Gilbert channel. Our proposed analytical framework is developed for streaming over TCP channels and it is more comprehensive to analyze the playback continuity, delay and video quality by taking the variability of TCP throughput, buffer delay, buffer size and streaming rate into account. Moreover, we extend our analytical framework for P2P multicast streaming. Evaluation of the performance and the quality of P2P streaming has been a hot topic in the recent past [17] [18] [19] [20] and [21]. However, these studies rely on packet traces that are collected from a limited number of measurement points to infer and estimate the P2P streaming quality. Without an analytical framework, it is difficult to systematically understand the properties and quality of large scale P2P streaming systems through limit measurements.

[2]Streaming rate and its variability

[3]The recommended one-way transmission time for interactive applications is 150 ms in ITU-T Rec. G.114 (05/2003).

[4]The window size during a TOP (timeout period) is fixed to 1. Hence, playout buffer occupancy during a TOP monotonously decreases.



(a) Packets sent during a TDP (b) Evolution of window sizes

Fig. 1. TCP Reno model [3]. The congestion window size is adjusted by packet losses which can be detected by either triple-duplicate ACKs or timeouts. TDP (triple-duplicate period) is a period ending in triple-duplicate ACKs while TOP (timeout period) is a period ending in a timeout. During a TDP the congestion window size increases by $1/b$ packets per round. b is the number of packets that are acknowledged by an ACK and it is typically 1. We define the duration of a ***round*** as the round-trip time (RTT), namely the time between the transmission of packets and the reception of the first acknowledgment (ACK). The window size is immediately cut to half when triple-duplicate ACKs are detected. However, when less than three duplicated ACKs are received, a TO period (TOP) begins. During a TOP, the sender waits for a timeout interval, and then the congestion window size is set to one and the sender retransmits non-acknowledged packets. The timeout interval in a TOP increases exponentially for each unsuccessful retransmission until it reaches $64T_0$.

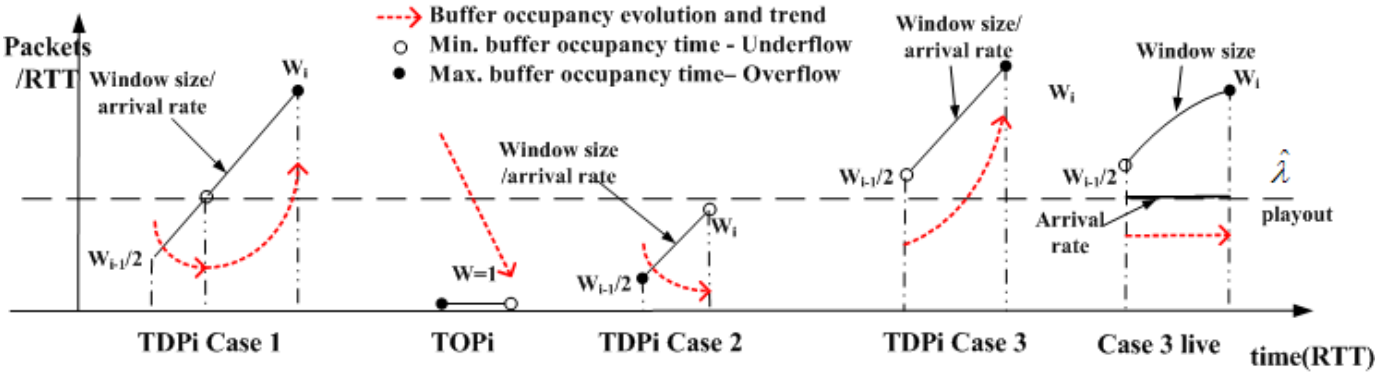


Fig. 2. Window bounds and buffer occupancy for TCP streaming – TDP Case 1: window sizes and arrival rates roughly match playout rate (The minimum buffer occupancy is when the playout rate equals the arriving rate); TDP Case 2: window sizes and arrival rates are smaller than stream playout rate (The maximum buffer occupancy is at the beginning of the period and the minimum buffer occupancy is in the end); TDP Case 3: window sizes and arrival rates are higher than the stream playout rate for stored streams while the arrival rate is constrained by the coding rate for live streams (The maximum buffer occupancy is at the end of the period and the minimum buffer occupancy is in the beginning); TOP: window sizes during TOP are set to be one and the buffer occupancy is monotonously decreasing.

III. TCP Streaming system

A. System model

The right side of Figure 3 shows the model of video streaming systems consisting of TCP connections and a receiver or a P2P node (for example, node h_1) in the P2P streaming system shown at the left side of the figure. We assume that the sender transmits CBR video packets over a unicast TCP connection, and that the receiver node is equipped with a buffer in front of a video decoder/player. We assume that one single playout buffer is used both for smoothing delay jitter and for delivering to the decoder as work in [22] showed that a single receiver buffer always performs at least as well as two separate buffers. The buffer is used to accommodate the fluctuations of the TCP receiving rate. The decoder waits to fill the buffer to some degree before beginning to display the video. For P2P streaming system, data in the buffer are sent to children nodes over TCP as soon as possible.

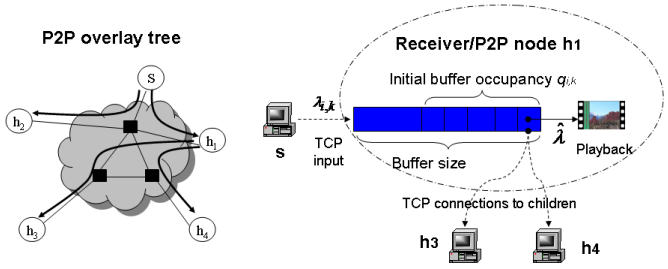


Fig. 3. TCP streaming model for P2P node (The left side shows the P2P overlay tree while the buffer model of a node is described at the right side.)

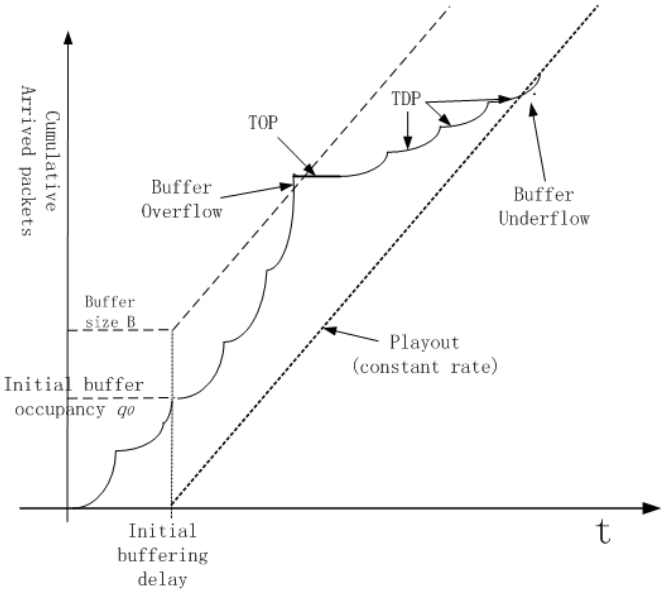


Fig. 4. Buffer occupancy illustration (Each arc is a buffer occupancy evolution of a TDP (as given in Eq.(4)). The straight line is the buffer occupancy evolution during a TOP (as given in Eq.(9)).

TDP_0	the TDP when packets begin to be played out
$q_{1,0}$	buffer occupancy at the beginning of of TDP_1
$q_{i,k}$	buffer occupancy right after (at) round k of TDP_i
q_{min}	minimum buffer occupancy of the TDP
q_{max}	maximum buffer occupancy of the TDP
B	buffer size
p	packet loss rate
R	round-trip time
$\lambda_{i,k}$	arrival rate of video packets at round k of TDP_i
$\hat{\lambda}$	video encoding/playout rate (packets per round)
$\bar{\lambda}$	average TCP sending rate (packets per round)
$r(R,p)$	average available TCP throughput (packets per round)
q_0	initial buffer occupancy when playback begins
w_0	the TCP window size when playback begins
D	initial buffer delay, defined by $q_0/\hat{\lambda}$
b	number of packets that are acknowledged by an ACK
W_i	window size at the end of TDP_i (window bounds)
X_i	the number of rounds in TDP_i where a TD loss is detected
Y_i	the number of packets sent in TDP_i
T_0	retransmission timeout
P_u	buffer underflow probability
P_o	buffer overflow probability

TABLE I
NOTATION FOR TCP STREAMING

To model the arrival rate and its variation of TCP connections, we apply the TCP model proposed in [3]. Let $\lambda_{i,k}$ be the arrival rate (namely, the window size per RTT round) of video packets at round k of TDP_i and $\hat{\lambda}$ be the video encoding/playback rate. We assume $\hat{\lambda}$ is constant. Since QoE degradations are a) buffer underflows that result in playout disruptions or freezings of images; b) buffer overflows that result in the loss of data and non-smooth video; c) the buffer delay include the initial buffer delay as well as re-buffer delay to fill the buffer for avoiding further underflows. Intuitively, the longer the buffer delay is, the less the buffer underflow events are. The larger the buffer size is, the less the buffer overflow events are. Fig. 4 illustrates the buffer size with the initial buffer occupancy and the buffer overflow/underflow events during streaming.

Table I summarizes the variables and notation for the TCP streaming system model and buffer occupancy.

B. Problem statement

Given desired delay, buffer overflow and underflow probabilities in the network condition of the packet loss rate and RTT, the research questions are a) How big should the buffer be? b) How much should the buffer be filled before starting to playback?

An analytical framework for TCP streaming is needed to analyze the trade off of the probability of discontinuing playback, the buffer size and delay for any specific stream, given the dynamics of the Internet and the variability of TCP throughput.

Guided by the analytical framework, we are interested in dimensioning the buffer size and the initial buffer delay to provide streaming quality with low buffer delay and limited underflow probability and overflow probability. Or, we may estimate the underflow/overflow probability for given buffer sizes and the initial buffer delay. Live streaming applications such as football matches often have a requirement of very little playout delay and continuing playout with low underflow probability. End systems such as cell phones or PDAs are often confronted with limited buffer space and not able to store the entire multimedia content, however still desire low overflow probability of streams.

C. QoE Metrics

We evaluate the QoE of a video streaming application by startup delay (namely, the initial buffer delay), the buffer underflow probability and the buffer overflow probability, and video quality.

–The *initial buffer delay* is the time difference between the first streaming packet arrived in the buffer and it is sent to decoder. For simplicity, we define the initial buffer delay D as the time to fill the buffer up to initial buffer occupancy q_0 at the coding rate $\hat{\lambda}$, namely $D = \frac{q_0}{\hat{\lambda}}$.

–The *buffer underflow/overflow probability* is defined by the probability that the buffer is underflowed/overflowed in a specific TDP or TOP. The total number of buffer underflow/overflow events can be estimated from the buffer underflow/overflow probability given the streaming length in terms of the average duration of TDPs and TOPs. We measure the buffer underflow/overflow events instead of the number of the packets involved in buffer underflow/overflow events. A buffer underflow/overflow event may consist of one or more packets underflowed/overflowed within a TDP/TOP. The difference between buffer underflow/overflow events and buffer underflow/overflow packets is small for small underflow/overflow probabilities. In our experiments, a long underflow/overflow event covering multiple TDP time is counted as multi-

ple events. Therefore, we only consider the number of underflow/overflow events.

–The *video quality* is calculated by the encoding rate for simplicity reason, although the most widely used objective video quality metric is the peak signal-to-noise ratio (PSNR). Moreover, the video quality is a non-linear, increasing and concave function of the encoding rate [23]. The higher the encoding rate is, the better the quality of video streaming and the higher the PSNR are.

The video encoding rate and playback rate are the same $\hat{\lambda}$. While the encoding rate is a metric for the video quality, we assume that it is constant during the QoE analysis. Therefore, for a specific stream we consider mainly the initial buffer delay, the buffer underflow and overflow probabilities as QoE metrics, and trade the encoding rate and buffer sizes for the playback continuity and timeliness in this paper.

We further define the *buffer size*, denoted by B, as the amount of buffer allocated by the system. The buffer size may vary from very small such as in nodes in the sensor network to very large such as in PCs and Servers in the Internet.

D. Buffer occupancy

Now, let us take a look at the buffer occupancy at time t. We assumed that the arriving rate is the window size number of packets per round, $\lambda_{i,k}$ is the number of packets received during round k of TDP_i, and $\hat{\lambda}$ packets are drained at round k of TDP_i. We can get $\lambda_{i,k}$,

$$\lambda_{i,k} = W_{i-1}/2 + k - 1 \tag{1}$$

where $k = 1, 2, \cdots X_i$ and X_i is the round number when the ith triple duplicate ACKs event is detected $X_i = W_i - \frac{W_{i-1}}{2} + 1$. We denote round 0 or $k = 0$ as the beginning of TDP i before round 1 starts. For simplicity, both the playout rate and the encoding rate are $\hat{\lambda}$ and constant. The playout buffer occupancy at round of k of TDP_i is $q_{i,k}$ given by

$$\begin{aligned} q_{i,k} &= q_{i,k-1} + \lambda_{i,k} - \hat{\lambda} \\ &= q_{i,0} + \sum_{n=1}^{k} (\lambda_{i,n} - \hat{\lambda}) \\ &= q_{i,0} + \frac{k^2}{2} + \frac{(W_{i-1} - 2\hat{\lambda} - 1)k}{2} \end{aligned} \tag{2}$$

where $q_{i,0}$ is the playout buffer occupancy at the beginning of TDP i (when $k = 0$ or before round 1 begins). The playout buffer occupancy at round 0 of TDP i is given as follows,

$$\begin{aligned} q_{i,0} &= q_{1,0} + \sum_{m=0}^{i-1} \sum_{k=1}^{X_m} [\lambda_{m,k} - \hat{\lambda}] \\ &\approx q_{1,0} + i * \bar{k}(\bar{\lambda} - \hat{\lambda}) \end{aligned} \tag{3}$$

where $i \to \infty$, $\bar{\lambda}$ is the average throughput of TCP estimated by the throughput from TDP_0 to TDP_{i-1}: $\bar{\lambda} \approx \frac{\sum_{m=0}^{i-1} \sum_{k=1}^{X_m} [\lambda_{m,k}]}{\sum_{m=0}^{i-1} X_m}$, and $\bar{k}$ is the average number of rounds of a TDP estimated by the number of rounds from TDP_0 to TDP_{i-1}: $\bar{k} \approx \frac{\sum_{m=0}^{i-1} X_m}{i}$.

Substituting Eq.(3) into Eq.(2), the playback buffer occupancy at round k of TDP_i is

$$q_{i,k} = q_{1,0} + i * \bar{k}(\bar{\lambda} - \hat{\lambda}) + \frac{k^2}{2} + \frac{(W_{i-1} - 2\hat{\lambda} - 1)k}{2} \tag{4}$$

The maximum buffer occupancy q_{max} and the minimum buffer occupancy q_{min} of TDP_i could be at the time of

$$\begin{aligned} \frac{\partial q_{i,k}}{\partial k} &= 0, or \\ k &= 0 \ (namely \ at \ q_{i,0}), or \\ k &= X_i \end{aligned} \tag{5}$$

The buffer is overflowed during TDP_i when the maximum buffer occupancy of TDP_i reaches the buffer size B, namely $q_{max} \geq B$. On the other hand, buffer is underflowed during TDP_i when the minimum buffer occupancy of TDP_i reaches empty, namely $q_{min} \leq 0$. We have

$$P_u = P\{q_{min} \leq 0\} \tag{6}$$

$$P_o = P\{q_{max} \geq B\} \tag{7}$$

Therefore, we find in our analysis Eq. (4) - Eq. (7) that the overflow and underflow probability can be derived from TCP window bounds W, initial buffer delay q_0, streaming playback rate $\hat{\lambda}$ in terms of RTT and buffer size B.

Moreover, the window size and thus its bounds during TOPs is 1 and the successfully arriving data is zero. Since all packets are lost in a TOP, no packet is successfully delivered to the receiver buffer. The playout buffer occupancy during TOP is monotonously decreasing and given by

$$\begin{aligned} q_{i,k} &= q_{i,k-1} - \hat{\lambda} \\ &= q_{i,0} - k * \hat{\lambda} \end{aligned} \tag{8}$$

Substitute Eq.(3) into Eq.(8)

$$q_{i,k} \approx q_{1,0} + i * \bar{k}(\bar{\lambda} - \hat{\lambda}) - k * \hat{\lambda} \tag{9}$$

As a receiving node is either in a TDP or a TOP, the buffer underflow probability at time t is defined by the sum of conditional probabilities, such that,

$$\begin{aligned} P\{q_{min} \leq 0\} &= P\{q_{min} \leq 0 | t \in TDP\} P\{t \in TDP\} \\ &\quad + P\{q_{min} \leq 0 | t \in TOP\} P\{t \in TOP\} \end{aligned} \tag{10}$$

Similarly, the overflow probability for a given buffer size B is,

$$\begin{aligned} P\{q_{max} \geq B\} &= P\{q_{max} \geq B | t \in TDP\} P\{t \in TDP\} \\ &\quad + P\{q_{max} \geq B | t \in TOP\} P\{t \in TOP\} \end{aligned} \tag{11}$$

We will derive the solutions of buffer underflow/overflow probabilities for a given buffer size and the initial buffer delay in Section IV by using the distribution of TCP

window bounds in Eq. (39) and Eq. (42) derived in Appendix A and also in our previous work [13]. Or, knowing desired underflow/overflow probabilities we can determine the appropriate buffer size and the initial buffer delay by using the distribution of TCP window bounds.

IV. Performance analysis: from the case study to the general case

In this section, we study the buffer underflow probability and the buffer overflow probability given the buffer size and the initial buffer delay in TCP streaming. We show that a closed form of TCP congestion window bounds in Eq.(39) is useful in TCP streaming applications for finding the right buffer sizes and the initial buffer delay. For a stream with encoding/playout rate at $\hat{\lambda}$ packets per round, a TCP window bound can be a)as low as 1 during TOP, b)higher than 1 and up to $\hat{\lambda}$, c) higher than $\hat{\lambda}$ and up to $2\hat{\lambda}$, or d)higher than $2\hat{\lambda}$. This also corresponds to the four cases of Figure 2.

In what follows, we will show the underflow probability and the overflow probability for 3 detailed scenarios of TCP streaming applications. Then, we will finally present the underflow probability and the overflow probability for the general TCP streaming.

A. Scenario 1: The coding rate matches TCP available throughput

In this subsection, we show that the variance of TCP window bounds determine the probability of buffer overflow and underflow in this scenario where $\hat{\lambda} = \bar{\lambda} = r(R,p)$. Intuitively, the higher the variance of TCP window bounds is, the higher the underflow and overflow probabilities for the streaming application are. Moreover, the higher the initial buffer delay is, the lower the underflow probability is. The higher the buffer size is, the lower the overflow probability is. Let's consider a TCP streaming application where the playout rate matches the TCP average sending rate (the average window size), namely Case 1 in Figure 2 where $\frac{W_i}{2} < \hat{\lambda} < W_i$. For more details of the following derivation, we refer to Appendix B.

For Case 1 in Figure 2, the minimum buffer occupancy q_{min} of a TDP_i is at round k where $\frac{\partial q_{i,k}}{\partial k} = 0$. We get $k = \hat{\lambda} - \frac{W_{i-1}}{2} + 1$ and substitute it into Eq.(4).

The **underflow probability** of the buffer of TDP_i is the underflow probability of q_{min} at $k = \hat{\lambda} - \frac{W_{i-1}}{2} + 1$,

$$\begin{aligned} P_u &= P\{q_{min} \le 0\} \\ &= P\{q_{i,0} - \frac{W_{i-1}^2}{8} + \frac{2\hat{\lambda}+1}{4}W_{i-1} - \frac{\hat{\lambda}^2+\hat{\lambda}}{2} \le 0\} \\ &= P\{W_{i-1} \le (2\hat{\lambda}+1) - \sqrt{8q_{i,0}+1}\} \end{aligned}$$

When the playout rate matches the TCP average sending rate, $q_{i,0}$ approximately equals $q_{1,0}$ in Eq.(3). Substituting the CDF of TCP window bounds in Eq.(39), we obtain

$$P_u = P\{q_{min} \le 0\} = F(2\hat{\lambda} + 1 - \sqrt{8q_{1,0}+1}) \qquad (12)$$

where $q_{1,0} \approx q_0 + \frac{1}{18p} + \frac{0.2}{\sqrt{p}}$ (See in Eq. (53)).

The **overflow probability** of TDP_i is the overflow probability of q_{max} at round $k = X_i$ (equivalently, $k = 0$),

$$\begin{aligned} P_o &= P\{q_{max} > B\} \\ &= P\{q_{min} + \frac{W_i^2}{2} + W_i(1/2 - \hat{\lambda}) + \hat{\lambda}^2 - \hat{\lambda} > B\} \\ &= P\{W_i > \hat{\lambda} - 1/2 + \sqrt{2(B - q_{min}) + 1/4}\} \\ &= 1 - F(\hat{\lambda} - 1/2 + \sqrt{2(B - q_{1,min}) + 1/4}) \end{aligned} \qquad (13)$$

where $q_{i,min}$ approximately equals $q_{1,min}$ in Eq.(3) when the playout rate matches the TCP average sending rate, and $q_{1,min} \approx q_0 - \frac{1}{36p} + \frac{0.2}{\sqrt{p}}$ (See in Eq. (56)).

Knowing the desired underflow/overflow probabilities we can now determine appropriate buffer sizes and the initial buffer delay by using the distribution of TCP window bounds.

Given an underflow probability for a TCP streaming application with rate $\hat{\lambda}$, we derive **the initial buffer occupancy** q_0 from Eq.(45)

$$q_0 = \frac{(2\hat{\lambda} + 1 - F^{-1}(P_u))^2 - 1}{8} - \frac{1}{18p} - \frac{0.2}{\sqrt{p}} \qquad (14)$$

And the **buffer delay** D is

$$D = \frac{q_0}{\hat{\lambda}} \qquad (15)$$

Given an overflow probability for a TCP streaming application with rate $\hat{\lambda}$ and given the initial buffer occupancy, the needed **buffer size** B is derived from Eq.(49)

$$B = \frac{(1/2 - \hat{\lambda} - F^{-1}(1 - P_o))^2 - 0.25}{2} + q_0 - \frac{1}{36p} + \frac{0.2}{\sqrt{p}} \qquad (16)$$

B. Scenario 2: The coding rate is much higher than TCP available throughput

Now let us consider an application where the streaming rate is much higher than TCP available bandwidth, namely window size and playout rate are mostly in Case 2 and $W_i < \hat{\lambda}$ ($\bar{\lambda} = r(R,p) < \hat{\lambda}$) in Figure 2. In this application, available bandwidth is not enough for live streaming. Intuitively, a large buffer delay is needed for smooth playing back, otherwise underflows are unavoidable.

The buffer occupancy is as follows and minimum $q_{i,k}$ is at $k = X_i = W_i - \frac{W_{i-1}}{2} + 1$ as $\frac{W_{i-1}}{2} + X_i < \hat{\lambda}$.

$$\begin{aligned} q_{min} &= q_{i,X_i} \\ &= q_{i,0} + \frac{k^2}{2} + \frac{(W_{i-1} - 2\hat{\lambda} - 1)k}{2} \\ &= q_{i,0} + \frac{(W_i - \frac{W_{i-1}}{2} + 1)(W_i + \frac{W_{i-1}}{2} - 2\hat{\lambda})}{2} \end{aligned} \qquad (17)$$

The **underflow Probability** P_u is

$$\begin{aligned} P_u &= P\{q_{min} \le 0 | t \in case2\} \\ &= P\{q_{i,0} + \frac{(W_i - \frac{W_{i-1}}{2} + 1)(W_i + \frac{W_{i-1}}{2} - 2\hat{\lambda})}{2} \le 0\} \end{aligned} \qquad (18)$$

The underflow probability changes over the TDPs as $q_{i,0} = q_{1,0} + i * \bar{k}(\bar{\lambda} - \hat{\lambda})$ and $\bar{\lambda} - \hat{\lambda} < 0$. Therefore, when $i > \frac{q_{1,0}}{\bar{k}(\hat{\lambda}-\bar{\lambda})}$ we have $q_{i,0} = q_{1,0} + i * \bar{k}(\bar{\lambda} - \hat{\lambda}) = 0$ and

$$P_u \to 1 \tag{19}$$

For $i < \frac{q_{1,0}}{\bar{k}(\hat{\lambda}-\bar{\lambda})}$, we approximate by $W_i \approx W_{i-1}$

$$\begin{aligned} P_u &\approx P\{W_i < \sqrt{\frac{4\hat{\lambda}^2}{9} + \frac{4\hat{\lambda}}{3} + 1 - \frac{8q_{i,0}}{3}} + \frac{2\hat{\lambda}}{3} \\ &= F(2\hat{\lambda} + 1 - \sqrt{6p}(q_{i,0} + \frac{1}{3p})) \end{aligned} \tag{20}$$

The **overflow probability** P_o in TDP_i is zero, as the TCP throughput is always lower than the playout rate.

$$P_o = P\{q_{max} > B | t \in case2\} = 0 \tag{21}$$

C. Scenario 3: The coding rate is much lower than TCP available throughput

In this scenario, the coding rate is much lower than the available TCP throughput. For live streams, the TCP sending rate equals the coding rate which is lower than the available TCP throughput ($\hat{\lambda} = \bar{\lambda} < r(R,p)$). For stored streams, TCP available bandwidth equals the average sending rate in this scenario ($\hat{\lambda} < \bar{\lambda} = r(R,p)$). In this scenario ($\hat{\lambda} < r(R,p)$ in general), TCP window size and playout rate are mostly in Case 3, namely $P\{t \in case3\} = 1 - F(2\hat{\lambda}) \to 1$. Intuitively, live streaming applications will generally provide good performance (both in terms of underflows and overflows) as TCP available bandwidth is over provisioned in the scenario. For stored streaming applications, a large buffer space is needed for fast downloaded data.

Figure 5 illustrates concave increasing window sizes measured for a live streaming application in this scenario. The window bounds distribution in Eq. (42) still holds in this scenario. However, it takes longer than one RTT to fill a window size number of packets as the sender is application-limited and offer less data to the network than the congestion window allowing. If the congestion window is increased too large to be fully utilized for a period of one RTO, congestion window will stop increasing as stated in RFC 2861 (TCP congestion window validation) [24]. For stored streams, the window sizes are linear additive increasing as normal.

The minimum $q_{i,k}$ is at $k = 0$ when $\frac{W_{i-1}}{2} \geq \hat{\lambda}$, Then the **underflow Probability** P_u is

$$P\{q_{min} \leq 0 | t \in case3\} = 0 \tag{22}$$

Then maximum $q_{i,k}$ at $k = W_i - \frac{W_{i-1}}{2} + 1$ $\frac{W_{i-1}}{2} \geq \hat{\lambda}$.

$$\begin{aligned} q_{i,k} &= q_{i,0} + \frac{k^2}{2b} + \frac{(W_{i-1} - 2\hat{\lambda} - 1)k}{2} \\ &= q_{i,0} + \frac{(W_i - \frac{W_{i-1}}{2} + 1)(W_i + \frac{W_{i-1}}{2} - 2\hat{\lambda})}{2} \end{aligned} \tag{23}$$

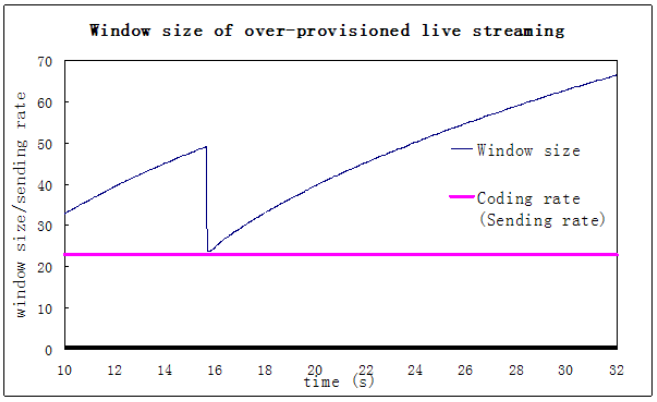


Fig. 5. Concave increasing window sizes for over-provisioned live streaming (It takes longer than one RTT to send a window size number of packets as sender is application-limited. If the congestion window is increased too large to be fully utilized for a period of one RTO, congestion window will stop increasing as stated in RFC 2861.)

The **overflow probability** P_o **for stored streams** is

$$\begin{aligned} P_o &= P\{q_{max} > B | t \in case3\} \\ &= P\{q_{i,0} + \frac{(W_i - \frac{W_{i-1}}{2} + 1)(W_i + \frac{W_{i-1}}{2} - 2\hat{\lambda})}{2} > B\} \\ &\approx P\{W_i > \sqrt{\frac{8}{3}(B - q_{i,0}) + (\frac{1+2\hat{\lambda}}{3})^2} + \frac{1+2\hat{\lambda}}{3}\} \\ &= 1 - F(\sqrt{\frac{8}{3}(B - q_{i,0}) + (\frac{1+2\hat{\lambda}}{3})^2} + \frac{1+2\hat{\lambda}}{3}) \end{aligned} \tag{24}$$

where $W_i \approx W_{i-1}$. The overflow probability changes over the TDPs as $q_{i,0} = q_{1,0} + i * \bar{k}(\bar{\lambda} - \hat{\lambda})$ and $\bar{\lambda} - \hat{\lambda} > 0$. Therefore, when $i > \frac{B - q_{1,0}}{\bar{k}(\bar{\lambda} - \hat{\lambda})}$, $q_{i,0} = q_{1,0} + i * \bar{k}(\bar{\lambda} - \hat{\lambda}) = B$ and $q_{i,0} >> \frac{(W_i - \frac{W_{i-1}}{2})(W_i + \frac{W_{i-1}}{2} - 2\hat{\lambda} - 1)}{2}$

$$P_o \to 1 \tag{25}$$

In this scenario, TCP available bandwidth is higher than streaming coding/playout rate, namely $\bar{\lambda} - \hat{\lambda} > 0$. For live streaming, the actual send rate is not higher than coding rate, namely $\lambda_{i,k} \leq \hat{\lambda}$. Therefore, we have $\lambda_{i,k} = \hat{\lambda}$ and the **overflow probability** P_o **for live streams** is

$$P\{q_{max} > Q | t \in case3\} = 0 \tag{26}$$

D. The general TCP streaming

Finally, we compute the underflow probability of $TDP_i (i >> 1)$ for TCP streaming in general. In general TCP streaming, TOP and all TDP cases in Figure 2 may occur.

$$\begin{aligned} P_u &= P\{q_{min} \le 0\} \\ &= P\{q_{min} \le 0 | t \in case1\} \cdot P\{t \in case1\} \\ &\quad + P\{q_{min} \le 0 | t \in case2\} \cdot P\{t \in case2\} \\ &\quad + P\{q_{min} \le 0 | t \in case3\} \cdot P\{t \in case3\} \\ &\quad + P\{q_{min} \le 0 | t \in TOP\} \cdot P\{t \in TOP\} \\ &= F(2\hat{\lambda} + 1 - \sqrt{8q_{i,0} + 1}) \\ &\quad \cdot (F(2\hat{\lambda}) - F(\hat{\lambda})) \cdot (1 - min(1, \frac{3}{E[W]})) \\ &\quad + F(2\hat{\lambda} + 1 - \sqrt{6p}(q_{i,0} + \frac{1}{3p})) \\ &\quad \cdot F(\hat{\lambda}) \cdot (1 - min(1, \frac{3}{E[W]})) \\ &\quad + P\{q_{i,0} - T_0 \cdot \hat{\lambda} \le 0\} \cdot min(1, \frac{3}{E[W]}) \quad (27) \end{aligned}$$

where $P\{t \in TOP\} \approx min(1, \frac{3}{E[W]})$, $q_{i,0} = q_{1,0} + i * \bar{k}(\bar{\lambda} - \hat{\lambda})$.

Remarks: From Eq. (27), we learn that,

1) The initial buffer delay must be greater than TCP timeout for a desired low underflow probability; for very low delay applications such as interactive video/audio conferences, TCP timeouts, typically on the order of hundreds miniseconds, post a problem to the delay requirement of 150 ms in ITU-T Rec. G.114 (05/2003). Therefore, the underflow probability of these very low delay applications is higher than the probability of TCP timeouts;
2) The ratio of TCP average sending rate $\bar{\lambda}$ and playout rate $\hat{\lambda}$, thus $q_{i,0}$, determines mainly the underflow probability: $P_u \to 0$ if $\bar{\lambda} - \hat{\lambda} > 0$ and $P_u \to 1$ if $\bar{\lambda} - \hat{\lambda} < 0$;
3) For the case of $\bar{\lambda} = \hat{\lambda}$, either it is over provisioned live streaming where the larger the difference between available bandwidth and the playout rate, the lower the underflow probability (namely, $\hat{\lambda} \downarrow$ and $p \downarrow \Longrightarrow P_u \downarrow$). In particular, $P_u \to 0$ when $\bar{\lambda} << r(R, p)$. Or, it is the case that available TCP bandwidth matching playout rate where P_u is determined by the initial buffer delay, timeouts and the gamma distribution with parameter p.

The overflow probability for the general TCP streaming is

$$\begin{aligned} P_o &= P\{q_{max} > B\} \\ &= P\{q_{max} > B | t \in case1\} \cdot P\{t \in case1\} \\ &\quad + P\{q_{max} > B | t \in case2\} \cdot P\{t \in case2\} \\ &\quad + P\{q_{max} > B | t \in case3\} \cdot P\{t \in case3\} \\ &\quad + P\{q_{max} > B | t \in TOP\} \cdot P\{t \in TOP\} \\ &= P\{q_{max} > B | t \in case1\} \cdot P\{t \in case1\} \\ &\quad + P\{q_{max} > B | t \in case3\} \cdot P\{t \in case3\} \quad (28) \end{aligned}$$

The buffer occupancy is monotonously decreasing during TOP and TDP where $W_i < \hat{\lambda}$. Therefore, the probability of overflow in TOP and Case 2 is zero, namely $P\{q_{max} > B | t \in TOP\} = 0$ and $P\{q_{max} > B | t \in case2\} \cdot P\{t \in case2\} = 0$ in the derivation of Eq. (28).

For live streaming either in an over-provisioning case (Case 3) or in the rate-matching case (Case 1), the actual send rate is not higher than coding/playout rate. In Case 3, the overflow probability is close to zero namely $P\{q_{max} > B | t \in case3\} = 0$. For over-provisioning live streaming, the larger difference between available bandwidth and playout rate, the lower overflow probability (namely, $\hat{\lambda} \downarrow$ and $p \downarrow \Longrightarrow P_o \downarrow$). Thus, plugging the overflow probability in Case 1 (Eq. (49)) into Eq. (28), we get

$$\begin{aligned} P\{q_{max} > B\} = (1 - F(\hat{\lambda} - 1/2 + \sqrt{2(B - q_{1,min}) + 1/4})) \\ \cdot (F(2\hat{\lambda}) - F(\hat{\lambda}))(1 - min(1, 3\sqrt{\frac{3p}{8}})) \quad (29) \end{aligned}$$

For stored streaming, plugging the overflow probability in Case 1 (Eq. (49)) and the overflow probability in Case 3 (Eq. (24)) into Eq. (28), we get

$$\begin{aligned} P\{q_{max} > B\} &= (1 - F(\hat{\lambda} - 1/2 + \sqrt{2(B - q_{1,min}) + 1/4})) \\ &\quad \cdot (F(2\hat{\lambda}) - F(\hat{\lambda})) \cdot (1 - min(1, 3\sqrt{\frac{3p}{8}})) \\ &\quad + (1 - F(\sqrt{\frac{8}{3}(B - q_{i,0}) + (\frac{1 + 2\hat{\lambda}}{3})^2} + \frac{1 + 2\hat{\lambda}}{3})) \\ &\quad \cdot (1 - F(2\hat{\lambda})) \cdot (1 - min(1, 3\sqrt{\frac{3p}{8}})) \end{aligned} \quad (30)$$

V. Extending from unicast streaming to P2P multicast streaming

In the recent past, more and more multimedia streams have been distributed over application-layer multicast or P2P multicast. A very large number of P2P multicast proposals and systems have emerged, see in [25] [26] and their references. The majority of these systems can be classified as tree-based systems such as End System Multicast (ESM) [27] and *PeerCast*[5]. In tree-based P2P multicast systems, the participating nodes are formed into a tree structure with "parent-child" relationships before data can be transmitted. Whenever a parent node receives a packet, it creates a copy and forwards it to each of its children. A tree-based system is a natural approach and it is efficient for streaming applications when the tree and nodes are stable. In most current P2P multicast streaming systems, TCP is used for underlying congestion control.

We extend our analytical framework for tree-based P2P multicast streaming systems. As shown in Fig.3, a TCP connection to a child will send packets as soon as possible in the P2P streaming. Let us assume a parent node with the initial buffer delay $q_0/\hat{\lambda}$ will play a specific packet out at time t. Consequently, the packet will be played out at a child node at $t + R/2$ when its initial buffer delay is the same as $q_0/\hat{\lambda}$. R is the round-trip time between

[5]http://www.peercast.org/

the child node and the parent node. The buffer at the parent node is underflowing when the packet due at time t arrives late than t. Consequently, the packet will arrive later than $t + R/2$ at the child node and the child's buffer with the initial delay $q_0/\hat{\lambda}$ will underflow. Therefore, the ***underflow probability***[6] of a peer c with initial buffer delay $q_{c,0}/\hat{\lambda}$ is

$$\widetilde{P}_{c,u} = 1 - \prod_i (1 - P_{i,u}) \tag{31}$$

where $P_{i,u}$ is the underflow probability of connection i when having the same initial buffer delay as $q_{c,0}/\hat{\lambda}$. Connection i is any unicast upstream along the path from peer c to root. We assume that the underflow probability of a unicast connection is independent from other connections. The underflow probability of each upstream connection is approximately given in Eq. (27).

A new arriving packet will be dropped due to overflow when the buffer at the receiver is full of unplayed packets. Otherwise, it will replace the earliest played out packets. Similar to the underflow probability for a peer, a packet is not dropped/overflowed only when it is not dropped at each of its upstream nodes. The ***overflow probability***[7]of a peer c with initial buffer delay $q_{c,0}/\hat{\lambda}$ and buffer size B_c is

$$\widetilde{P}_{c,o} = 1 - \prod_i (1 - P_{i,o}) \tag{32}$$

where $P_{i,o}$ is the overflow probability of connection i when having the initial buffer delay $q_{i,0}/\hat{\lambda}$ and buffer size B_i. Connection i is any unicast upstream along the path from Peer c to root. Again, we assume that the overflow probability of a unicast connection is independent from other connections. The overflow probability of each upstream connection is approximated in Eq. (29) when the available bandwidth of its child connections are no less than the streaming playout rate. When the available bandwidth of a child connection is lower than the playout rate, new packets will replace the played but unforwarded packets in the buffer (packets are not overflowed but dropped[8]). The performance of downstream nodes will decrease dramatically. A QoS-based tree should avoid these cases of performance degradation.We assume a bandwidth/QoS-based tree construction mechanism for real time P2P streaming: trying to find a good quality node with enough download/upload bandwidth as a parent. Intermediate nodes forward the received data as soon as possible without any delay.

VI. Evaluation

In this section, we validate our analytical framework by evaluating the underflow and overflow probability for various buffer sizes and buffer delay of TCP streaming in both simulated and testbed environments.

A. Evaluations on TCP streaming buffer

1) Simulation Setup: We use network simulator NS-2.30 [28] for our simulations. In our simulation setting, TCP streaming connections are generated over a dumbbell topology with one bottleneck link. Cross traffic is generated by some FTP flows. We find that the packet loss rate p is close to constant, and the packet losses are independent as the correlation between subsequent losses is highly reduced in the high multiplexing on the bottleneck link. We simulate in the following 3 scenarios with a duration of 1000 seconds of a precoded video stream. Moreover we run each scenario 20 times using random seeds. The bottleneck link has 100 ms delay and has different capacities for each scenario. All access links have 100 Mbps capacity and 1 ms delay. In all scenarios, the TCP stream coding/playout rate is 1 Mbps. The TCP initial window size is 20 packets, and the packet size is set to 1200 bytes. In all scenarios, the buffer is filled up to half buffer size before packets start to be sent for decoding, namely $q_0 = \frac{B}{2}$. Thus, we adjust both the initial buffer delay and the buffer size at the same time. We examine the streaming overflow/underflow probabilities/events in buffers with various the buffer sizes and the initial buffer delay($q_0/\hat{\lambda}$).

Scenario 1: The TCP available bandwidth equals to the stream code rate. In this scenario, 19 FTP flows are competing for the bottleneck link with 20 Mbps capacity. The measured packet loss rate is 0.002533 and the measured RTT is 0.229367 ms.

Scenario 2: The stream code rate is higher than the TCP available rate where 19 FTP flows are competing for the bottleneck link with 10 Mbps capacity. The measured packet loss rate is 0.010773 and the measured RTT is 0.2319 ms.

Scenario 3: The stream code rate is lower than the TCP available rate where 19 FTP flows compete for the bottleneck link with 60 Mbps. The measured network packet loss rate is 0.000275 and the measured RTT is 0.223241 ms.

2) Results of 3 Scenarios: We first compare the measured overflow and underflow probabilities with the analytical results in the previous sections we derived for Scenario 1. The overflow probability is calculated as the number of overflowed TDP/TOPs derived by the total number of TDP/TOPs, so is underflow probability.

Indeed, Fig. 6 and Fig. 7 show that the measured results are close to our model and analytical results. Small values for the standard error of the estimate σ_{est} confirm this finding, underlining the expressiveness of our analytical framework. From Fig. 6 and Fig. 7 we also learn that more overflow events are more likely to happen than underflow events with the initial buffer delay q_0 set to half of buffer size when TCP available bandwidth exactly matches the coding rate. This is somewhat counter-intuitive. We may

[6]More general speaking, packets are arrived later than its playback time

[7]More general speaking, packets are dropped or skipped due to the full buffer

[8]This is a shortcoming of the single buffer design for both playback and sending buffer of children connections

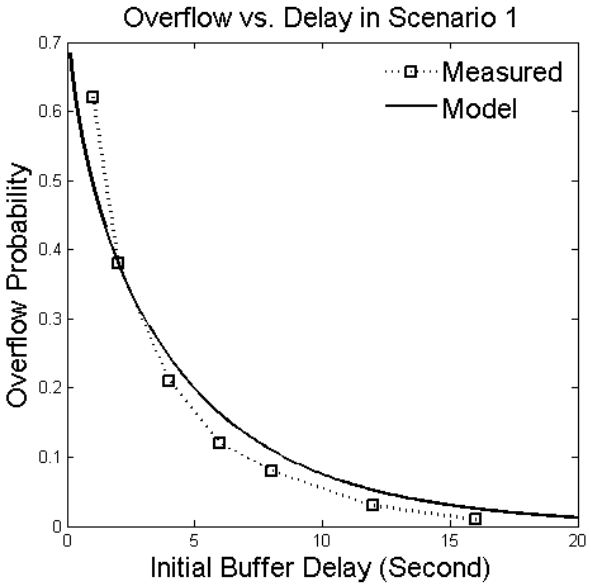


Fig. 6. Overflow probabilities for various initial buffer delays in Scenario 1 (Initial buffer occupancy and buffer size are both linearly to initial buffer delays. Initial buffer occupancy: $q_0 = D\hat{\lambda}$, buffer size: $B = 2q_0$, the standard error of the estimate: $\sigma_{est} = 0.053$)

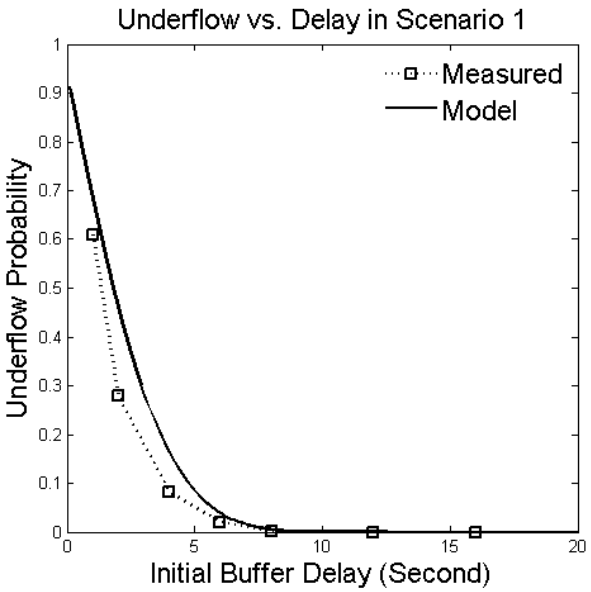


Fig. 7. Underflow probabilities for various initial buffer delays in Scenario 1 (The initial buffer occupancy is linearly to the initial buffer delay. Initial buffer occupancy: $q_0 = D\hat{\lambda}$, the standard error of the estimate: $\sigma_{est} = 0.080$)

expect the overflow and underflow probability are roughly same with the initial buffer delay set to half of the buffer size. Our explanation is that the window size at the time of starting to playback is more likely closer to the minimum buffer occupancy than the maximum buffer occupancy of a TDP as the buffer occupancy is non-linear with the round k in the TDP. In this scenario, TCP can provide streaming for a low underflow/overflow probability with less than 30 seconds buffer delay when streaming at the average TCP available rate in a steady network.

In the simulation Scenario 2, the overflow probability measured is very close to zero, obviously because the TCP throughput is always lower than playout rate. This is expected from our model. Figure 8 shows the underflow events (probability) in the simulation of Scenario 2 with varying initial buffer delay. For a specific TDP_i, its initial buffer occupancy $q_{i,0}$ namely $q_{1,0} + i * \bar{k}(\bar{\lambda} - \hat{\lambda})$, decides the underflow probability of this TDP_i. Given the streaming time length and $\bar{\lambda} - \hat{\lambda}$, the underflow probability tends to be 0 when $q_{1,0} > i * \bar{k}(\bar{\lambda} - \hat{\lambda})$. When $q_{1,0} < i * \bar{k}(\bar{\lambda} - \hat{\lambda})$, the underflow probability tends to be 1. Figure 8 shows how much the underflow probability is affected with different initial buffer delay $q_{1,0}$ given the streaming duration of 1000 seconds and $\bar{\lambda} - \hat{\lambda}$.

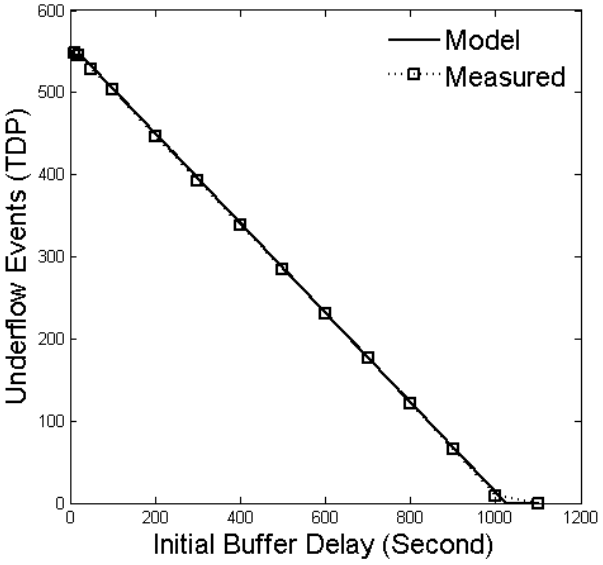


Fig. 8. Underflow probability in Scenario 2

In the simulation of Scenario 3, the overflow events measured is shown in Figure 9 with varying initial buffer delay for stored streams. For a specific TDP_i, the buffer size B and its initial buffer occupancy $q_{i,0}$ namely $q_{1,0} + i * \bar{k}(\bar{\lambda} - \hat{\lambda})$, decides the overflow probability of this TDP_i. Given the streaming time length and $\bar{\lambda} - \hat{\lambda}$, the overflow probability tends to be 0 when $B - q_{1,0} > i * \bar{k}(\bar{\lambda} - \hat{\lambda})$. When $B - q_{1,0} < i * \bar{k}(\bar{\lambda} - \hat{\lambda})$, the overflow probability tends to be 1. The underflow probability for stored streams in this Scenario is close to zero which is expected from the model.

We simulate also this over-provisional scenario for the live stream where CBR and video traces over TCP are used. We find that both the overflow probability and the underflow probability for the live stream are close to zero, which is intuitive and confirmed from our analysis. This means for over provisional live streaming only a small buffer size and initial buffer delay are enough for very low underflow/overflow probability.

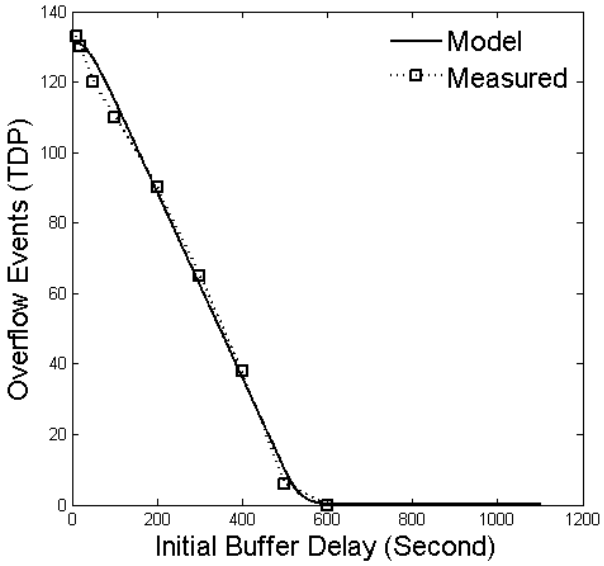


Fig. 9. Overflow probability in Scenario 3 for stored streams

B. P2P multicast streaming experiment results on PlanetLab

Since the network environment in the Internet is dynamic and unknown, we examine our model for overlay streaming qualitatively on PlanetLab[9].

Network conditions and measurements settings: From April to July in 2010, our experimental video server, located at Communication University of China, multicasted repeatedly the same video (Alizee - La Isla Bonita) with a streaming rate of 300 Kbps simultaneously into three P2P streaming channels. Our P2P system is an improved version of *PeerCast*. We set buffer size B to $B = 12$, 64, and 128 packets respectively and all $q_0 = B/2$ for three streaming channels. Each packet is of size 16 KBytes. We used Iperf[10] to measure the TCP available bandwidth in parallel with three streaming channels. We assume that three channels experience the same network conditions and the same streaming application, namely the same round-trip time R, packet loss rate p and playout rate $\hat{\lambda}$. We examine the different underflow/overflow probabilities of three channels with different buffer sizes. While we have carried out experiments on a large number of P2P topologies, we present here some representative results. Four P2P multicast nodes on PlanetLab were selected as shown in the topology in Fig.10. There were no long-term bottleneck connection in the topology as the average available TCP rates are 1-3Mbps measured by Iperf. Fig. 11 shows the underflow events (late packets) of P2P multicast streaming nodes for each channel while Fig.12 shows the overflow events detected.

The following observations in the experiments reflected our analysis in Section IV and Section V.

1. A larger buffer gets a smaller number of overflow and underflow events, however a longer delay at each node. The longer the buffer delay is, the less the buffer underflow events are. The larger the buffer size is, the less the buffer overflow events are.

2. Due to TCP traffic co-existing with our inserted *Iperf* traffic and a longer path to the multicasting server, child nodes experience a somewhat higher number of overflow and underflow events than their parent nodes for over-provisioned live streaming. The larger the difference between the available bandwidth and the playout rate is, the lower the underflow probability is.

3. During our experiments, we also find other possible factors influencing the overflow/underflow packets, namely dynamics of packet losses, timeouts and dynamics of streaming rate.

[9]www.planet-lab.org

[10]http://iperf.sourceforge.net/

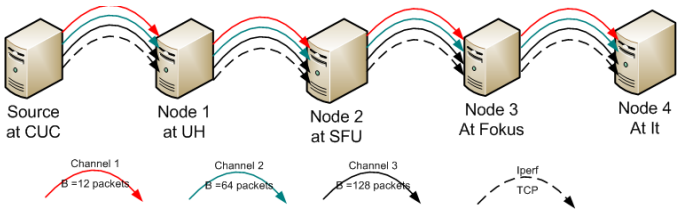


Fig. 10. Experiment example topology on PlanetLab

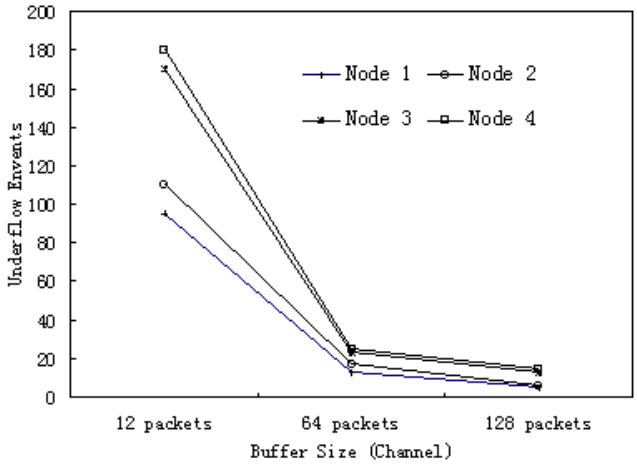


Fig. 11. Underflow events of peers and channels

VII. Conclusion

In this paper, we proposed an analytical framework for QoE in terms of the playback continuity (underflow and overflow probabilities), timeliness (delay) and the video quality in TCP-based streaming systems. Using the variance of TCP window bounds, we derived the underflow and overflow probabilities for the given buffer sizes and buffer delay. Or, given the required underflow/overflow probabilities we allocate appropriate buffer sizes and the initial buffer delay. From our analytical framework we conclude the following general results. a) AIMD plays a role only in the rate match scenarios. However, timeouts and retransmission always plays a role in very low delay

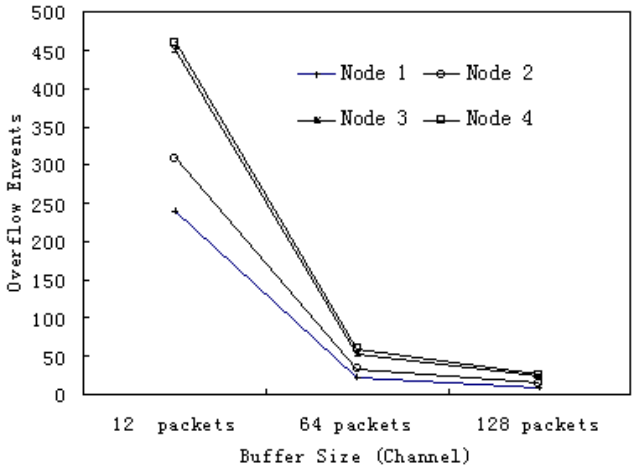


Fig. 12. Overflow events of peers and channels

applications and large RTT networks. b) The difference between the available bandwidth and the streaming rate mainly decide the buffer size needed for a low discontinuity of playout. c) In P2P multicast streaming, due to the long path to the broadcasting server, child nodes having the same buffer size and initial delay will experience a somewhat higher number of overflow and underflow events than parent nodes.

References

[1] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," 2003. http://www.ietf.org/rfc/rfc3550.txt.

[2] S. Floyd and K. Fall, "Promoting the use of end-to-end congestion control in the internet," *IEEE/ACM Trans. Netw.*, vol. 7, pp. 458–472, August 1999.

[3] J. Padhye, V. Firoiu, D. F. Towsley, and J. F. Kurose, "Modeling TCP Reno performance: a simple model and its empirical validation," *IEEE/ACM Trans. Netw.*, vol. 8, pp. 133–145, April 2000.

[4] E. Kohler, M. Handley, and S. Floyd, "Designing DCCP: congestion control without reliability," in *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '06, (New York, NY, USA), pp. 27–38, ACM, 2006.

[5] I. Rhee and L. Xu, "Limitations of equation-based congestion control," *Networking, IEEE/ACM Transactions on*, vol. 15, pp. 852 –865, aug. 2007.

[6] M. Vojnović and J.-Y. Le Boudec, "On the long-run behavior of equation-based rate control," *IEEE/ACM Trans. Netw.*, vol. 13, pp. 568–581, June 2005.

[7] L. Guo, E. Tan, S. Chen, Z. Xiao, O. Spatscheck, and X. Zhang, "Delving into internet streaming media delivery: a quality and resource utilization perspective," in *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, IMC '06, (New York, NY, USA), pp. 217–230, ACM, 2006.

[8] M. Saxena, U. Sharan, and S. Fahmy, "Analyzing video services in web 2.0: a global perspective," in *Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, NOSSDAV '08, (New York, NY, USA), pp. 39–44, ACM, 2008.

[9] C. Systems, "Cisco visual networking index: Forecast and methodology, 2010 - 2015," 2011.

[10] V. Misra, W.-B. Gong, and D. Towsley, "Stochastic Differential Equation Modeling and Analysis of TCP Windowsize Behavior," in *Proceedings of IFIP WG 7.3 Performance*, November 1999.

[11] S. Bohacek and K. Shah, "TCP throughput and timeout - steady state and time-varying dynamics," vol. 3, pp. 1334 – 1340 Vol.3, nov. 2004.

[12] C.-C. Wu, K.-T. Chen, C.-Y. Huang, and C.-L. Lei, "An empirical evaluation of VoIP playout buffer dimensioning in Skype, Google talk, and MSN Messenger," in *Proceedings of the 18th international workshop on Network and operating systems support for digital audio and video*, NOSSDAV '09, (New York, NY, USA), pp. 97–102, ACM, 2009.

[13] J. Yan, W. Mühlbauer, and B. Plattner, "An analytical model for streaming over TCP," in *NEW2AN*, vol. 6869 of *Lecture Notes in Computer Science*, pp. 370–381, Springer, 2011.

[14] B. Wang, J. Kurose, P. Shenoy, and D. Towsley, "Multimedia streaming via TCP: An analytic performance study," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 4, pp. 16:1–16:22, May 2008.

[15] T. Kim and M. H. Ammar, "Receiver Buffer Requirement for Video Streaming over TCP," in *Proceedings of Visual Communications and Image Processing 2006, vol. 6077*, (San Jose, CA, USA), 2006.

[16] G. Liang and B. Liang, "Effect of Delay and Buffering on Jitter-Free Streaming Over Random VBR Channels," *Multimedia, IEEE Transactions on*, vol. 10, no. 6, pp. 1128 –1141, 2008.

[17] X. Hei, Y. Liu, and K. Ross, "Inferring Network-Wide Quality in P2P Live Streaming Systems," *Selected Areas in Communications, IEEE Journal on*, vol. 25, no. 9, pp. 1640 –1654, 2007.

[18] X. Hei, C. Liang, J. Liang, Y. Liu, and K. Ross, "A Measurement Study of a Large-Scale P2P IPTV System," *Multimedia, IEEE Transactions on*, vol. 9, no. 8, pp. 1672 –1687, 2007.

[19] H. Chang, S. Jamin, and W. Wang, "Live streaming performance of the zattoo network," in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, IMC '09, (New York, NY, USA), pp. 417–429, ACM, 2009.

[20] D. Ciullo, M. Garcia, A. Horvath, E. Leonardi, M. Mellia, D. Rossi, M. Telek, and P. Veglia, "Network Awareness of P2P Live Streaming Applications: A Measurement Study," *Multimedia, IEEE Transactions on*, vol. 12, no. 1, pp. 54 –63, 2010.

[21] D. Ciullo, M.-A. Garcia, A. Horvath, E. Leonardi, M. Mellia, D. Rossi, M. Telek, and P. Veglia, "Network awareness of P2P live streaming applications," in *Parallel Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, pp. 1 –7, May 2009.

[22] T. Stockhammer, H. Jenkac, and G. Kuhn, "Streaming video over variable bit-rate wireless channels," *Multimedia, IEEE Transactions on*, vol. 6, pp. 268 – 277, april 2004.

[23] J. Yan, K. Katrinis, M. May, and B. Plattner, "Media- and tcp-friendly congestion control for scalable video streams," *Multimedia, IEEE Transactions on*, vol. 8, no. 2, pp. 196 – 206, 2006.

[24] M. Handley, J. Padhye, and S. Floyd, "TCP congestion window validation," 2000.

[25] M. Hosseini, D.T.Ahmed, S. Shirmohammadi, and N. Georganas, "A Survey of Application-Layer Multicast Protocols," *IEEE in Communications Surveys and Tutorials*, vol. 9, no. 3, pp. 58–74, 2007.

[26] J. Liu, S. G. Rao, B. Li, and H. Zhang, "Opportunities and Challenges of Peer-to-Peer Internet Video Broadcast," in *In (invited) Proceedings of the IEEE, Special Issue on Recent Advances in Distributed Multimedia Communications*, 2007.

[27] Y.-h. Chu, S. G. Rao, and H. Zhang, "A Case for End System Multicast (keynote address)," in *SIGMETRICS '00: Proceedings of the 2000 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, (New York, NY, USA), pp. 1–12, ACM, 2000.

[28] "ns-2 simulator." www.isi.edu/nsnam/ns/.

[29] E. Altman, K. Avrachenkov, and C. Barakat, "A stochastic model of TCP/IP with stationary random losses," *IEEE/ACM Trans. Netw.*, vol. 13, pp. 356–369, April 2005.

[30] S. Ha, I. Rhee, and L. Xu, "CUBIC: a new TCP-friendly high-speed TCP variant," *SIGOPS Oper. Syst. Rev.*, vol. 42, pp. 64–74, July 2008.

[31] J.-C. Bolot, "End-to-end packet delay and loss behavior in the internet," in *Conference proceedings on Communications architectures, protocols and applications*, SIGCOMM '93, (New York, NY, USA), pp. 289–298, ACM, 1993.

[32] R. V. Hogg and A. T. Craig, *Introduction to Mathematical Statistics, 4th edition.* New York: Macmillan, 1978.

Appendix A
TCP congestion windows Model

In this appendix, we propose a simple model based on Padhye's model in [3], however, our model is accurately tractable by a Gamma distribution. Then we derive the distribution of the distribution of TCP window bounds which is useful for the performance analysis of TCP streaming.

Many TCP models such as [3] [29] [10] [11] have been proposed and used for network research investigating problems such as active queue management, TCP-friendliness, performance analysis, etc. Our TCP model differs from the previous work and we study congestion window variance with a focus on computing bounds for TCP congestion window sizes.

Our analytical model is based on the TCP Reno/NewReno/SACK model [3] in Figure 1 because TCP Reno is still the most widely used TCP implementations in the Internet[11]. p denotes the packet loss rate. We assume p is stationary for a certain time scale as the correlation between subsequent losses is highly reduced by the high statistical multiplexing on high speed links [31].

To derive the solution, we first assume that the probability of timeouts is small. This is the case when the packet loss rate is low and the congestion of the network is also low. In such cases we can restrict our analysis mostly on the TDP of TCP streaming. We further extend our model by taking TOP into account.

A. Triple duplicate ACKs

The duration of a *round* within a TDP period is defined by a duration between the transmission of packets and the reception of the first acknowledgment (ACK) in a congestion window. The duration of a round is equal to the round-trip time (RTT) and it is independent of the window size when the network is highly statistically multiplexed with TCP connections. According to the additive increase/multiplicative decrease TCP algorithm the window size increases by one packet per round and is reduced to half of its size immediately after receiving a triple duplicate ACK. Let W_i be the window size at the end of the ith TDP (TDP_i). Hence, $W_{i-1}/2$ is the window size at the beginning of TDP_i, and W_i and $W_{i-1}/2$ are the upper and lower bounds for the TCP window sizes during TDP_i. According to Padhye et al. [3], the number of rounds during TDP_i is $W_i - W_{i-1}/2 + 1$. We refer to Y_i as the number of packets successfully sent in TDP_i illustrated in Fig.13, and compute it as follows:

[11]Windows 7 is using TCP Reno by default. BSD systems and BSD-based systems such as Mac OS by default are using SACK, a modifaction of Reno. Reno is also available and used in Linux distributions although is being replaced by new TCP variants such as Cubic [30].

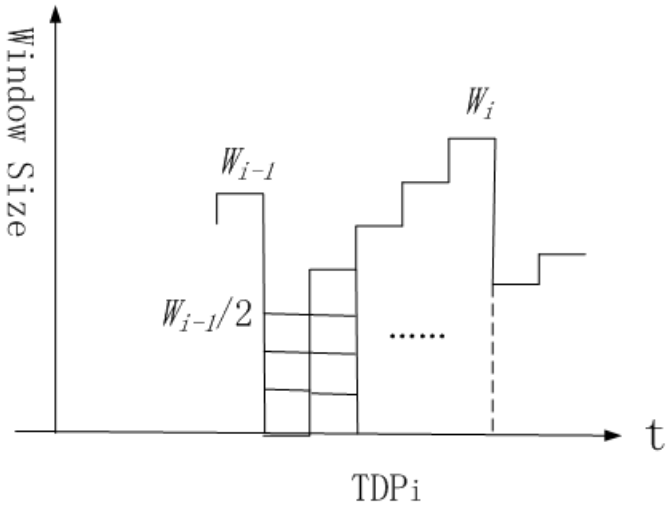


Fig. 13. The number of packets successfully sent in TDP_i

$$\begin{aligned} Y_i &= \sum_{k=1}^{X_i}(\frac{W_{i-1}}{2}+k-1) \\ &= \frac{(W_i^2-\frac{W_{i-1}^2}{4})}{2}+\frac{W_i+\frac{W_{i-1}}{2}}{2} \\ &\approx \frac{(W_i+1)^2-\frac{(W_{i-1}+1)^2}{4}}{2}, (W_i >> 1) \end{aligned} \quad (33)$$

where $X_i = W_i - \frac{W_{i-1}}{2} + 1$. We can model the distribution of W_i as a Markov process and obtain the stationary distribution of window size W_i, and thus also of $W_{i-1}/2$ at the beginning of the same TDP. Transforming Eq. (33) results in the state transition from W_{i-1} to W_i:

$$(W_i+1)^2 - \frac{(W_{i-1}+1)^2}{4} \approx 2 \cdot Y_i, \qquad i = 1, 2, ... \quad (34)$$

We assume each packet has the loss probability p and packet losses are independent in low-congestion networks. Thus, $P\{Y_i = n\} = (1-p)^n \cdot p$, $(n = 1, 2, \ldots)$, and it is a Geometric distribution for packet loss rates $p \ll 1$. While Y_i and W_i are discrete variables in the end, we treat them as continuous as they could be divided into non-integers in the following derivation. Each Y_i of ith TDP $(i = 1, 2, \ldots)$ follows the exponential distribution:

$$P\{Y_i = x\} = p \cdot e^{x \cdot ln(1-p)} = p \cdot e^{-p \cdot x}, (x \geq 0, p \ll 1) \quad (35)$$

Note that $Y_1, Y_2, \ldots, Y_i$ are independent, and all follow the exponential distribution. Then we define a new variable ψ as the sum of the weighted Y_i variables for the following derivation. In particular,

$$\psi = 2 \cdot \sum_{k=1}^{i} \frac{Y_k}{4^{i-k}} \quad (36)$$

According to statistical theory [32], the distribution for sequentially independent exponentially distributed variables with the same scale parameter $\theta = 1/p$ is a Gamma distribution with the sum of the shape parameters

s, and the scale parameter $\theta = 1/p$.

$$s = 2(\tfrac{1}{4^{i-1}} + \tfrac{1}{4^{i-2}} + \tfrac{1}{4^{i-3}} + ... + 1)) = 8/3 \ (i \to \infty)$$

Thus, the gamma distribution of ψ is $\Gamma(8/3, 1/p)$. Substituting Eq.(34) into Eq.(36), we get

$$\begin{aligned}\psi &= 2 \cdot \sum_{k=1}^{i} \frac{Y_k}{4^{i-k}} \\ &= (W_i+1)^2 - (W_{i-1}+1)^2/4 + (W_{i-1}+1)^2/4 \\ &\quad -(W_{i-2}+1)^2/4^2 + \dots \\ &= (W_i+1)^2 \end{aligned} \quad (37)$$

Let F_ψ is the cumulative distribution function (CDF) of ψ namely $F_\psi(W) \equiv Pr(\psi \le W)$. Then, the CDF of W_i is

$$F(W) \equiv Pr(W_i \le W) = Pr(\psi \le (W+1)^2) = F_\psi((W+1)^2) \quad (38)$$

Therefore the cumulative distribution function (CDF) of W_i is,

$$\begin{aligned} F(W) &= \frac{\gamma(8/3, p(W+1)^2)}{\Gamma(8/3)} \\ &= 0.6646\gamma(8/3, p(W+1)^2), W = 1, 2, 3 \dots \end{aligned} \quad (39)$$

Remark: Compared with the expected value of W_i in Eq.(14) in [3], our solution is the first to provide the distribution of W_i (the bounds of TCP window sizes). Note that the distribution of window sizes and bounds still holds for live streams where it may take more than one RTT round for data packets to fill the window sizes.

B. Timeouts

Now, we extend our analysis to include the case where packet losses are detected by timeouts. Let Q be the probability that a packet loss is recognized via a timeout. During a TOP, the window size is reduced to one. Then, the closed form for the distribution of TCP window (upper) bounds is:

$$\hat{F}(W) = (1-Q)F(W) + Q, W \ge 1 \quad (40)$$

According to Padhye et al. [3] the probability Q can be approximated as follows:

$$Q \approx min(1, 3\sqrt{\frac{3p}{8}}) \quad (41)$$

By substituting Eq. (41) into Eq. (40), we finally obtain:

$$\begin{aligned}\hat{F}(W) \approx (1 - min(1, 3\sqrt{\frac{3p}{8}}))F(W) \\ +min(1, 3\sqrt{\frac{3p}{8}}), W \ge 1 \end{aligned} \quad (42)$$

We point out that the distribution of Eq. (39) is an approximation of Eq. (42) if the probability of a timeout is very low. In Section IV we will leverage our insights to describe the relationship between buffer sizes, the initial buffering delay and overflow/underflow probabilities in TCP streaming.

In our previous work [13], we verified the TCP window bound model, and results showed that the bounds of TCP window can be closely modeled by the Gamma distribution. In addition, we have got almost the same results for different TCP variants including TCP Reno, TCP NewReno and TCP SACK. Hence, our TCP model, which is a simple model however accurately tractable by Gamma distribution, results accurate and closed analytical solutions. Namely, Eq. (39) is a simple yet accurate closed form solution.

Appendix B
Underflow and overflow probability in Scenario 1

Scenario 1: Coding rate matches TCP available throughput

This appendix shows more details of the derivation of the probability of buffer overflow and underflow in Scenario 1. Let's consider a TCP streaming application where the playout rate matchesthe TCP average sending rate (average window size), namely Case 1 in Figure 2 where $\frac{W_i}{2} < \hat{\lambda} < W_i$. In this application, we assume $P\{t \in case1\} = F(2\hat{\lambda}) - F(\hat{\lambda}) \to 1$, thus $P\{t \in TDP\} \to 1$ and $P\{t \in TOP\} \to 0$.

The minimum buffer occupancy q_{min} of TDP_i is at $q_{i,k}$ when $\frac{\partial q_{i,k}}{\partial k} = 0$, thus $k = \hat{\lambda} - \frac{W_{i-1}}{2} + 1$,

$$\begin{aligned} q_{min} &= q_{i,0} + \frac{k^2}{2} + \frac{(W_{i-1} - 2\hat{\lambda} - 1)k}{2} \\ &= q_{i,0} - \frac{W_{i-1}^2}{8} + \frac{2\hat{\lambda}+1}{4}W_{i-1} - \frac{\hat{\lambda}^2 + \hat{\lambda}}{2} \end{aligned} \quad (43)$$

The **underflow Probability** of the buffer of TDP_i is the underflow probability of q_{min},

$$\begin{aligned} P_u &= P\{q_{min} \le 0\} \\ &= P\{q_{i,0} - \frac{W_{i-1}^2}{8} + \frac{2\hat{\lambda}+1}{4}W_{i-1} - \frac{\hat{\lambda}^2+\hat{\lambda}}{2} \le 0\} \\ &= P\{W_{i-1} \le (2\hat{\lambda}+1) - \sqrt{8q_{i,0}+1}\} \end{aligned} \quad (44)$$

When the playout rate matches the TCP average sending rate, $q_{i,0}$ approximately equals $q_{1,0}$ in Eq. (3). Substitute Eq. 44 into the CDF of TCP window bounds and we get finally

$$\begin{aligned} P_u &= P\{q_{min} \le 0\} \\ &= F(2\hat{\lambda} + 1 - \sqrt{8q_{1,0}+1}) \end{aligned} \quad (45)$$

Since the window size w_0 at initial buffer occupancy q_0 may range from $W_0/2$ to W_0. The worst case is the time to start playing back is the beginning/end of the first TDP. Then $q_{1,0} = q_0$, $w_0 = W_0/2$ and

$$P_{u,worst} = F(2\hat{\lambda} + 1 - \sqrt{8q_0+1}) \ge P_u \quad (46)$$

The maximum buffer occupancy q_{max} is at $q_{i,k}$ when $k = W_i - \frac{W_{i-1}}{2} + 1$ (or, $k = 0$ equivalently),

$$\begin{aligned} q_{max} &= q_{i,0} + \frac{k^2}{2} + \frac{(W_{i-1} - 2\hat{\lambda} - 1)k}{2} \\ &= q_{i,0} + \frac{(W_i - \frac{W_{i-1}}{2} + 1)(W_i + \frac{W_{i-1}}{2} - 2\hat{\lambda})}{2} \\ &= q_{min} + \sum_{k=\hat{\lambda}-\frac{W_{i-1}}{2}+2}^{W_i - \frac{W_{i-1}}{2}+1} (W_{i-1}/2 + k - 1 - \hat{\lambda}) \\ &= q_{min} + \frac{W_i^2}{2} + W_i(1/2 - \hat{\lambda}) + (\hat{\lambda}^2 - \hat{\lambda})/2 \quad (47) \end{aligned}$$

The **overflow probability** of TDP_i, namely the overflow probability of q_{max},

$$\begin{aligned} P_o &= P\{q_{max} > B\} \\ &= P\{q_{min} + \frac{W_i^2}{2} + W_i(1/2 - \hat{\lambda}) + \hat{\lambda}^2 - \hat{\lambda} > B\} \\ &= P\{W_i > \hat{\lambda} - 1/2 + \sqrt{2(B - q_{min}) + 1/4}\} \quad (48) \end{aligned}$$

When the playout rate matches the TCP average sending rate, $q_{i,min}$ approximately equals $q_{1,min}$ in Eq. (3). Substitute into the CDF of TCP window bounds and we get,

$$\begin{aligned} P_o &= P\{q_{max} > B\} \\ &= 1 - F(\hat{\lambda} - 1/2 + \sqrt{2(B - q_{1,min}) + 1/4}) \quad (49) \end{aligned}$$

The worst case is the time to start playing out is the minimum buffer occupancy of TDP_1. Then $q_{0,min} = q_0$, $w_0 = \hat{\lambda}$ and

$$P_{o,worst} = 1 - F(\hat{\lambda} - 1/2 + \sqrt{2(B - q_0) + 1/4}) \geq P_o \quad (50)$$

To find out the expected value of P_u, we need the expected value of $q_{1,0}$. Let us assume the playout started at k_0 round in TDP_0. The initial buffer occupancy is,

$$\begin{aligned} q_0 &= q_{1,0} - \sum_{n=k_0}^{X_0} (\lambda_{i,n} - \hat{\lambda}) \\ &= q_{1,0} + \frac{(X_0 - k_0)^2}{2} - (X_0 - k_0) \cdot (W_0 - \hat{\lambda} - \frac{1}{2}) - (W_0 - \hat{\lambda}) \end{aligned} \quad (51)$$

where $\lambda_{i,n} = W_0 - X_0 + n$. The expected value of $q_{1,0}$ is,

$$\begin{aligned} E[q_{1,0}] &= q_0 - E[\frac{(X_0 - k_0)^2}{2} - (X_0 - k_0) \cdot (W_0 - \hat{\lambda} - \frac{1}{2}) \\ &\quad -(W_0 - \hat{\lambda})] \\ &= q_0 - \frac{1}{X_0}[\sum_{k_0=1}^{X_0} \frac{(X_0 - k_0)^2}{2} \\ &\quad -(X_0 - k_0) \cdot (W_0 - \hat{\lambda} - \frac{1}{2})] + (W_0 - \hat{\lambda}) \\ &= q_0 - \frac{(X_0 - 1)(2X_0 - 1)}{12} + \frac{(X_0 - 1)(W_0 - \hat{\lambda} - 1/2)}{2} \\ &\quad +(W_0 - \hat{\lambda}) \\ &\approx q_0 - \frac{(\sqrt{\frac{2}{3p}} - 1)(2\sqrt{\frac{2}{3p}} - 1)}{12} \\ &\quad + \frac{(\sqrt{\frac{2}{3p}} - 1)(\sqrt{\frac{8}{3p}} - \hat{\lambda} - 1/2)}{2} + \sqrt{\frac{8}{3p}} - \hat{\lambda} \\ &= q_0 + \frac{1}{9p} + 1/6 + \sqrt{\frac{2}{3p}} - \frac{\sqrt{\frac{2}{3p}} + 1}{2}\hat{\lambda} \end{aligned} \quad (52)$$

where $W_0 \approx E[W_0] = \sqrt{\frac{8}{3p}}$, $X_0 \approx E[X_0] = \sqrt{\frac{2}{3p}}$. For scenario 1 $\hat{\lambda} \approx \sqrt{\frac{3}{2p}}$, then

$$E[q_{1,0}] \approx q_0 + \frac{1}{18p} + \frac{0.2}{\sqrt{p}} \quad (53)$$

Given desired underflow probability for a TCP streaming application with rate at $\hat{\lambda}$, **the initial buffer occupancy** q_0 and **buffer delay** D needed is

$$\begin{aligned} q_0 &= q_{1,0} - \frac{1}{18p} - \frac{0.2}{\sqrt{p}} \\ &= \frac{(2\hat{\lambda} + 1 - F^{-1}(P_u))^2 - 1}{8} - \frac{1}{18p} - \frac{0.2}{\sqrt{p}} \quad (54) \end{aligned}$$

$$D = \frac{q_0}{\hat{\lambda}} \quad (55)$$

To find out th expected value of P_o, we need the expected value of q_{min}. Similar to the method to find the expected value of $q_{1,0}$, we get

$$E[q_{min}] = E[q_{i,0}] - \frac{1}{12p} \approx q_0 - \frac{1}{36p} + \frac{0.2}{\sqrt{p}} \quad (56)$$

Given desired overflow probability for a TCP streaming application with rate at $\hat{\lambda}$ and initial buffer occupancy, the **buffer size** B needed is derived by substituting Eq.(56) into Eq.(49):

$$\begin{aligned} B &= \frac{(1/2 - \hat{\lambda} - F^{-1}(1 - P_o))^2 - 0.25}{2} + q_{min} \\ &= \frac{(1/2 - \hat{\lambda} - F^{-1}(1 - P_o))^2 - 0.25}{2} \\ &\quad +q_0 - \frac{1}{36p} + \frac{0.2}{\sqrt{p}} \end{aligned} \quad (57)$$