doi:10.3772/j.issn.1006-6748.2017.01.015

# A study on TCP performance of crowdsourced live streaming①

Zhou Jianer (周建二)②*** , Wu Qinghua* , Li Zhenyu* , Xu Chuan*** , Xie Gaogang*
(* Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, P. R. China)
(** University of Chinese Academy of Sciences, Beijing 100049, P. R. China)
(*** College of Communication, Chongqing University of Post and Telecommunication, Chongqing 400065, P. R. China)

**Abstract**

The prevalence of smart phone and improvement of wireless net promote the usage of crowdsourced live streaming, where individual users act as live streaming sources to broadcast themselves online. Characterizing the performance and identifying its bottleneck in such systems can shed light on the system design and performance optimization. TCP performance of a commercial crowdsourced live streaming system is examined by analyzing packet-level traces collected at streaming servers. TCP stalls that heavily hurt the QoE of user have been identified. In particular, the TCP stalls account for as much as 31.6% of the flow completion time for upload flows and result in abandonment of upload on the corresponding channels. Stalls caused by timeout retransmissions are further dissected and timeout retransmission characteristics are revealed to be dependent on the video encoding methods. These findings provide new insights in crowdsourced live streaming systems and can guide designers to improve the TCP efficiency.

**Key words**: crowdsourced live streaming, TCP performance, system design and measurement

## 0 Introduction

The flourish of mobile terminals and wearable devices has greatly stimulated the development of online video streaming, such as FaceTime, Twitch.tv, intelligent monitors[1]. Among these applications, crowdsourced live streaming emerges as a new way of video sharing, where individual users act as live streaming sources to broadcast themselves online. The multi-source nature of streaming leads to a significant difference between crowdsourced live streaming and traditional live streaming.

Traditional live streaming has stable providers and multiple viewers, such as TV and football broadcasters[2,3]. As such, uploading video content to live streaming servers is not a performance bottleneck at all. In addition, as the popularity of individual channels is predicable, content delivery network (CDN) can be leveraged for improved performance by replicating streaming content on CDN servers located in the regions where the channels are predicted to be popular[4,5].

Crowdsourced live streaming systems, on the other hand, host live streaming channels broadcasted by individual users. Uploading video content to live streaming servers thus might be a performance bottleneck that impacts not only on the uploading itself, but also the views that subscribe to the corresponding channels. Besides, the popularity of channels is less predictable, and thus it is challenging to deliver the streaming via CDN[6]. These unique features lead to a quest for the analysis of the crowdsourced live streaming system's architecture and the performance from a network's perspective.

Recently, much attention has been paid to the study of crowdsourced live streaming systems. Simoens, et al.[7] proposed a framework for streaming store and search, leaving the delivery problem untouched. Zhang, et al.[8] collected traces from clients to "infer" the system architecture and examine the user behavior. Jain, et al.[9] proposed a system to index the live streaming automatically with examining video frames. To accommodate the geo-distributed streaming, Chen, et al.[10] presented a framework that used the cost-effective cloud service, which facilitated deployment of crowdsourced live streaming system and the improvement of streaming quality. Stohr, et al.[11] examined the usage pattern of a crowdsourced live streaming system, and pointed out the difference between crowdsourced

① Supported by the National Basic Research Program of China (2012CB315801) and the National Natural Science Foundation of China (No. 6157060397).
② To whom correspondence should be addressed. E-mail: zhoujianer@ict.ac.cn
Received on Feb. 17, 2016

live streaming and traditional live streaming. These studies have not analyzed the system from TCP (transmission control protocol) performance perspective as done in this paper.

The dataset used in this paper consists of packet-level traces from front-end servers of a commercial crowdsourced live streaming system. The system adopts a tailored video encoding technique, where three types of video frames are used: intra frame (I frame), predictive frame (P frame), and voice frame (V frame)[12]. The TCP performance is first examined with particular focus on the TCP stalls. Then the impact of video encoding on TCP stalls is analyzed. To the best of our knowledge, the work is the first to study the TCP performance of crowdsourced live streaming systems at server side in wild. The main contributions are as follows:

- TCP stalls prolong the time required to upload content by over 30%. In addition, the upload flow throughput drops greatly as the packet reordering rate grows. The performance degradation caused by TCP stalls results in abandonment of uploads and viewing sessions.
- For specific channels, the performance of upload greatly affects the performance of views on the corresponding channels. In particular, about 40% of TCP stalls in upload leads to no data transferring to subscribers.
- The characteristics of timeout retransmission, an expensive packet loss recovery mechanism and one major cause for TCP stalls, are dependent on the types of frame.

Based on the findings above, corresponding implications are proposed. As the upload performance is vital to the overall performance, the front-end servers with better network should be selected for the video sharing users. To reduce the timeout retransmission influence, more aggressive timeout retransmission recovery methods should be deployed on front-end servers.

The rest of the paper is structured as follows. Section 1 describes the system architecture and the dataset used. In Section 2 the TCP performance for both upload and download phases are investigated. Section 3 examines the impact of streaming frames on performance. Section 4 summarizes our main findings and implications. Section 5 concludes the paper.

# 1 System overview and dataset

## 1.1 Background

Fig. 1 shows an examined architecture of the crowdsourced live streaming system. The system consists of five components: a centralized controller, front-end servers, distributed storage system, streaming sharing users (i. e., publishers) and streaming viewing users (i. e., subscribers). The centralized controller decides the front-end server to serve the streaming sharing or requests. For such live streaming system a controller can make it effective[13].

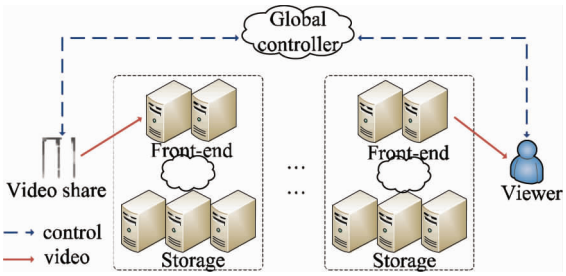


**Fig. 1** An overview of the crowdsourced live streaming system

Streaming frames generated by streaming sharing users are uploaded to storage system via the selected front-end server through TCP connections. Streaming viewers (i. e., subscribers) request the desired streaming (i. e., channels) from storage system via the selected front-end server, also using TCP. Note that the front-end servers are deployed in geo-distributed locations and are scheduled by controller. Hence, the front-end server that serves the subscribers may be different from one that receives the streaming content from publisher. In this work, performance of flows between front-end servers and users is analyzed.

H. 264 is used as the streaming codec in the system. There are 4 types of frames: I, P, bi-directional (B) and V[12,14]. Frame I is the key frame with the largest size (10 – 100 kbytes) and the slowest production rate (0. 25f/s). Frames P and B (both of which are called P frame for short in the follows) are predicted frames with size ranging between 2 and 10 kbytes. The average production speed of frame P is 25f/s and is adaptive according to the network quality. Frame V carries voice data with size less than 1 kbytes and production speed of about 25f/s.

## 1.2 Dataset

The packet-level dataset used for analysis was collected via tcpdump at one of the front-end servers for one week in July, 2015. Both the upload and download flows that go through the front-end servers were captured. Overall, 4.88 billion packets, corresponding to 1.24 million flows were collected. The average packet size is 1028 bytes and average flow size is 3.86MB.

Fig. 2 shows the variation of online users over one

week. Due to business considerations, the real number of users is not revealed. Sharing and viewing workloads follow similar diurnal pattern over the week: the number of online users continues to grow from 6AM and reaches peak at 11PM. This is not surprising because most of users take the crowdsourced live streaming as a kind of entertainment and thus do not use it during work time.

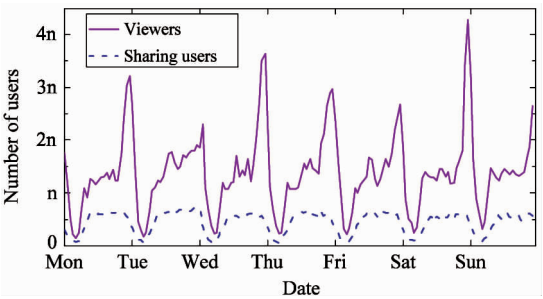


**Fig. 2** Number of viewers (subscribers) and sharing users (publishers) over one week

Fig. 3 shows the distribution of flow completion time (FCT) for sharing users (i. e., upload flows) and viewers (i. e., download flows). For upload flows, FCT measures the duration between the first synchronous (SYN) packet received by front-end servers and the last byte of data from the client. While for download flows, FCT is the duration between the first SYN packet and the acknowledgment of the last byte received by front-end server. Servers send keep-alive packets when clients do not upload data and terminate the flows if there is no data from client for 20s, which explains the observation from Fig. 3 that nearly all of upload flow completion time is more than 20s. It can be also observed that for both upload and download flows, about 60% of flows are less than 30s, indicating that most of users view or share short streaming. Since users might abort viewing before publisher terminates the video sharing process, the FCT of upload flows is in general longer than that of download flows.

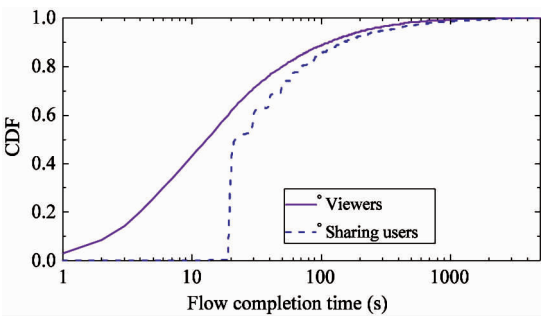


**Fig. 3** Upload and download flow completion time

Fig. 4 shows the distribution of throughput for upload and download. It can be observed that the throughput is comparable for the two types of flows and is relatively low (the median throughput is less than 100kB/s), because the throughput is heavily dependent on the frame production rate of publisher, which tends to be lower than the available bandwidth.

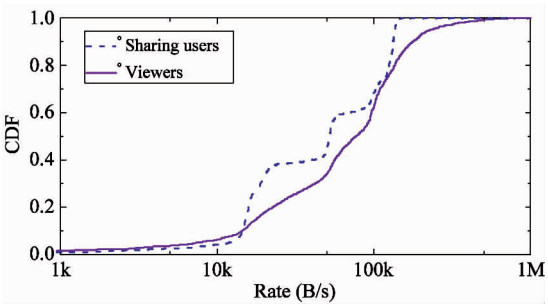


**Fig. 4** The throughput of upload and download flows

## 2 System measurement

In this section, the TCP traces are used to examine the TCP performance of crowdsourced live streaming system. The upload TCP flows are analyzed, which transfer data from streaming publishers to front-end servers, and then detail the download TCP flows, which move video data from front-end servers to subscribers. Finally, the impact of upload performance of a channel on the download flows is studied.

### 2.1 Analysis of TCP upload flows

Streaming publishers upload streaming data to front-end servers once new frames are generated in their devices. Two factors may influence the upload process: one is the interval of video frame producing process and the other is the delay of the network.

TCP stall is defined as an event where the duration between two consecutive packets received/sent by the server is larger than $\min(\tau \cdot \mathrm{SRTT}, \mathrm{RTO})$. Parameter $\tau$ is set to 2 in this work as the TCP endpoint is able to at least receive or send one packet during 2 × RTT in the ideal condition. As the front-end server (where we collected the data) is the receiver for upload flows, only the 3-way handshake (3WHS) round trip time (RTT) can be measured during which servers send a SYN packet and receive the corresponding acknowledgment (ACK). Thus, RTT is used in 3WHS as the smooth RTT (SRTT) for this flow.

The front-end server may receive a series of disordered packets that may be reordered by the network, or retransmitted because the previously transmitted segment is dropped. The front-end server could not distin-

guish whether the disordered packet is a packet reordering or packet retransmission. When a stall happens, receivers may have received a series of disordered packets, or sequential packets. The stalls of the former is further distinguished as reordering stalls and the latter as normal stalls. Keep-alive stalls are further defined as those caused by keep-alive packets sent by servers every 5s.

It is found that the stall time accounts for 31.6% of the completion time in upload flows. Among these stalls, normal stall contributes 93.5% of stall time, reordering stall contributes 1.3% of stall time, and keep-alive stall accounts for 5.2% of stall time. As the reordering stall just contributes 1.3% of stall time, the packet loss or reordering is not the main cause of stalls in upload flows. Instead, the interval of video frames is the main factor that influences the performance of upload flows. This can also be confirmed by the fact that the median throughput of upload flows is 60kB/s, much lower than the available upload bandwidth.

We define reordering packets as the packets which do not arrive in order. And we further define the reordering rate as the ratio of reordering packets to all data packets. It is found that the overall reordering rate is about 1% and among those disordered packets, 6% of them suffer from timeout reordering, in which cases the interval between two disordered packets is larger than 0.2s (i.e., the minimum retransmission timeout (RTO) set in Linux kernel).

Although packet loss or reordering happens rarely, once it happens, it will significantly impact the performance of upload flows. Packet reordering strongly correlates with the upload flow size, especially when the reordering rate becomes large. Table 1 shows the impact of reordering rate on upload flow size. We can see that a higher reordering rate leads to lower average throughput. The streaming publishers might abort the sharing in the case of lower throughput, which further results in smaller flow size and shorter flow completion time. For example, when the reordering rate is larger than 20%, the mean size of upload flows is only 0.73MB, much smaller than the flow size (5.1MB) when the reordering rate is less than 10%.

Table 1 Impact of reordering rate on upload flows

| Reordering rate | Average throughput | Flow size | FCT |
|---|---|---|---|
| <10% | 64kB/s | 5.1MB | 87.7s |
| 10~20% | 48kB/s | 4.7MB | 72.8s |
| >20% | 17kB/s | 0.73MB | 60.7s |

## 2.2 Analysis of TCP download flows

The stall in download flows is defined the same as that in upload flow (see Section 2.1). As the front-end server is the TCP sender in the video content distributing phase, RTT is computed dynamically each time it receives acknowledgment of the transmitted data. The stall time occupies 35.2% of the completion time of download flows.

Based on the position where each stall happens, TCP stalls are classified into the five types as shown in Table 2. Timeout stalls are those caused by the timeout retransmissions; resource constraint stall happens when there is no data in the buffer to transmit and it is often caused by delaying upload; packet delay stalls happen when ACKs are delayed, either by network or receiver; zero receive windows (rwnd) stalls happen when viewers do not have enough buffer to store the data; keep alive stalls caused by the fact that when there is no data to transmit, the front-end server will send a keep-alive packet every 5s to prevent the connection from being terminated.

Table 2 shows the ratio of different types of stalls in terms of time. It can be seen that the resource constraint accounts for 39.4% of download stalls, the largest part amongst all, which is partially caused by the stalls in upload flows. The publishers do not upload any data during stall time, which in turn leads to scarcity of video data to the corresponding subscribers.

Table 2 Ratio of different stalls in download flow

| Type | Rsrc cons. | Pkt delay | Zero rwnd | Keep alive | Time out | Others |
|---|---|---|---|---|---|---|
| % | 39.4 | 8.2 | 16.3 | 10.1 | 22.8 | 1.5 |

About 16.3% stalls happened due to zero rwnd, which implies that the viewers can not receive data, the performance of viewing video will be impacted greatly. Keep alive stalls on the other hand account for 10.1%, which happens because users might wait for some time after the completion of streaming share by publishers. As many as 22.8% of stalls belong to timeout retransmission due to network congestion. In Section 3, further analysis will be done to the timeout retransmission stalls.

The impact of RTT on TCP throughput is examined further. The flows with smaller RTTs are more likely to get high throughput, and the variations of RTTs will result in packet delay stalls[15]. RTT is determined by two factors. One is the number of hops between two nodes, which is controlled by the route algorithm. The other is the packets' buffering time at routers, which is controlled by both the router buffer ca-

pacity and the volume of packets in the network.

In Fig. 5 variation of RTTs is analyzed when the *in _flight* size is varied, which represents the amount of data bytes that have been sent already but not yet received by the viewers. Specially:

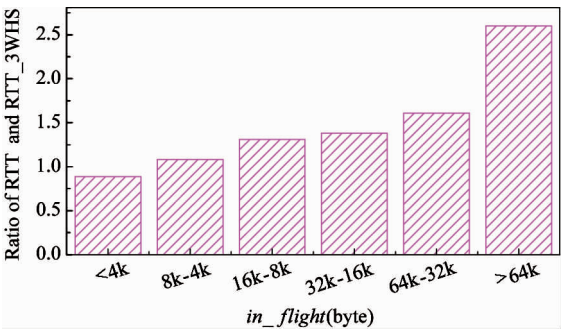


**Fig. 5** Ratio of RTT over RTT _3WHS when varying *in _flight*

$$in\_flight = packets\_ous + retrans\_out - (sacket\_out + lost\_out) \quad (1)$$

where *packets _out* represents the data bytes between *snd _una* (the packet with highest sequence number the receiver has acknowledged) and *snd _nxt* (the next new byte the sender would transmit), *retrans _out* is the data bytes which have been retransmitted but not yet acknowledged, *sacked _out* refers to the selective acknowledgment (SACK) data bytes, and *lost _out* measures the estimated dropped data bytes. the variation of RTT is examined by comparing it with RTT calculated in the 3WHS phase (referred to as RTT _3WHS). From Fig. 5, it can be seen that as the *in _flight* becomes large, RTT also becomes large. When the *in _flight* is 64kB, the RTT is even 2.6 times larger than the RTT which is in the 3WHS phase. This is because when there are a large number of *in _flight* packets, routers in the path need more time to process them and thus result in larger RTT.

### 2.3 Impact of upload performance on download performance

So far, the TCP upload and download flows have been investigated in the system separately. However, an upload flow may have an impact on the performance of the download flows that subscribe to the corresponding upload flow, i. e., which belong to the same channel. For example, reordering in upload flows may cause no data transferring to download flows. In this section, the impact of upload performance on the download performance will be detailed.

The system assigns each live streaming a unique ID for sharing and viewing, the upload and download flows can be matched by the video ID. Note that since the data was collected from one front-end server, not all flows could be matched, since upload and download flows of the same channel might be served by different front-end servers.

To measure the impact of stalls in upload flows on the download flows, each pair of upload and download flows will be matched first by the same video ID. In time $T$, if upload flows have a stall, then if download flows also have a stall in the interval of $[T+1\text{s}, T-1\text{s}]$ will be searched. If download flows do have, such stalls will be defined as stalls caused by upload flows. It is found that 37.2% of stalls in upload flows cause stalls in download flows. Table 3 shows different download flow stalls caused by upload stalls. From the table, we can see that 87.6% stalls caused by upload flows are resource constraint, as a stall in upload usually results in no data to send in download flows. 7.1% stalls are caused by timeout retransmissions, because that lack of data will result in not enough dupacks to trigger fast retransmit, a less expensive packet loss recover mechanism than timeout retransmission.

Table 3 Impact of upload stalls to download flows

| Type | Rsrc cons. | Time out | Pkt delay | Zero rwnd | Keep alive | Other |
|---|---|---|---|---|---|---|
| % | 87.6 | 7.1 | 4.5 | 0 | 0 | 0.7 |

## 3 Analysis of streaming frames

As mentioned in Section 1, frame size and production speed are dependent on frame types, which may result in different network performance. In this section, the performance of frames I, P and V in TCP download flows will be analyzed in detail.

### 3.1 Base video frame information

Table 4 shows the statistics on different types of frames. Frames P take up 82.8% and constitute the largest part of all frames. Frames V have the smallest percentage, just 5.8%. Frames I on the other hand account for 11.5%. In the system, the production intervals of I and V are set to 4s and 0.04s respectively. For download flows, the intervals begin as the server sends out a frame and end until it sends out next frame of the same type. We found the actual intervals of frames I and V are 4.85s and 0.069s respectively, because RTT jitter and packet loss may delay the frames. Frame P on the other hand has production interval 0.081s.

Table 4 Statistics of three types of frames

| Type | Per (%) | Interval (s) | Size | Loss rate | Timeout/loss (%) |
|---|---|---|---|---|---|
| I | 11.5 | 4.85 | 30kB | 1.1 | 11.8 |
| P | 82.8 | 0.081 | 3.4kB | 0.6 | 11.3 |
| V | 5.8 | 0.069 | 189bytes | 0.61 | 14 |

Frame I is the largest frame whose average size is 30kB, and its average size is 3.4kB while frame V is just 189 bytes, much smaller than the other two. The difference in frame size results in the distinct behaviour of burst for the three types of frames as shown in Fig. 6. Frame V experiences no burst packet as its size is less than one packet size (most of the packet size is around 1460 bytes, the same with maximum transmission unit (MTU)). For frame P, 15% of the bursts contain over 5 packets. About 40% of frames I lead to more than 5 burst packets. Routers are likely to drop burst packets[16], so the packet loss rate for frame I is 1.1%, which is higher than other two frames. In fact, P and V frame's packet loss rates are 0.6% and 0.61% respectively.

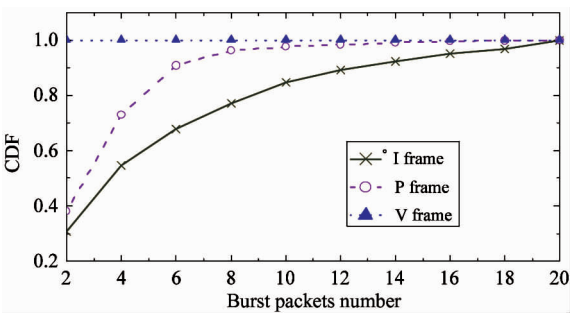


**Fig. 6** Burst packets of different frames

Among the lost packets, some result in timeout retransmissions, as they cannot be recovered by fast retransmit. For frame I, 11.8% of lost packets result in timeout retransmissions while for P and V the ratios are 11.3% and 14%, respectively. In what follows, the timeout retransmissions of different frames will be detailed.

### 3.2 Timeout retransmission of streaming frames

If fast retransmit fails to be triggered to recover packet loss, sender has to rely on costly timeout retransmission[17]. According to the situation when the timeout retransmission happens, the retransmission can be divided into several types. If the retransmitted packet by fast retransmit is dropped by the network, timeout retransmission has to be triggered and this timeout retransmission is referred to as double retransmission. If the lost packet is in the last three packets of the flow, there are not enough duplicate acknowledgments (dupacks) to trigger fast retransmit, it is referred to as tail retransmission. Insufficient number of dupacks to trigger fast retransmit may be also due to that the server sends less than 4 packets out (limited by either congestion window (cwnd) or rwnd), and referred to as less packet (pkt) retransmission. The ACK delay/loss or all packets sent out being lost will also result in timeout retransmission, as in both situations, no dupacks can be produced. These two types of timeout retransmissions are referred as ACK delay/loss and continuous loss, respectively.

Table 5 shows the ratio of each type of timeout retransmission for frames I, P and V. Those that cannot be classified into any of the above types are referred to as others. For all types of frames, double transmission contributes the largest part, 46.8% for frame I, 44.7% for frame P and 47.2% for frame V. That is because when the network drops a packet, the network has a high probability to become congested so the retransmitted packet via fast retransmit is likely to be dropped again. For frame V, the ratio of less pkt retransmission is 30.8%, much higher than that for I and P (11.4% and 20.2% respectively). As frame V is less than one MTU, when sending frame V, there are often fewer packets in the network compared with that when sending frame I. This explains the higher ratio of less pkt retransmission for frame V, and higher ratio of timeout retransmission for frame V despite the lower packet loss rate (shown in Table 4). Frames I and P contain a larger number of burst packets, so for I and P the ratios of ACK delay/loss timeout retransmission are much higher than that for V frame. Also because different number of burst packets, the ratio of continuous loss for frame V is 0, while those for I and P are 4.2% and 1.5%, respectively.

Table 5 Timeout retransmissions for different frames

| Timeout retrans. type | I(%) | P(%) | V(%) |
|---|---|---|---|
| Tail retrans. | 3.8 | 4.0 | 5.0 |
| Less pkt retrans. | 11.4 | 20.2 | 30.8 |
| Double retrans. | 46.8 | 44.7 | 47.2 |
| ACK delay/loss | 19.1 | 15.1 | 8.5 |
| Continuous loss | 4.2 | 1.5 | 0 |
| Others | 14.7 | 14.4 | 8.5 |

## 4 Summary of findings and implications

Table 6 summarizes the main findings and implications of our measurement results. Overall, our major findings could be grouped into two categories, one is

the significant impact of upload performance on the engagement of streaming sharing and on the quality of experience (QoE) of viewing, the other is the distinct characteristics of timeout retransmission stalls in the three types of frame.

Table 6 Summary of findings and their implications

| Findings | Implications |
|---|---|
| 37% of stalls in upload flows cause the download flows to have no data to transmit and thus result in a stall in download. | Optimizing of upload network and improving the performance of upload flows may greatly improve the performance of download flows. |
| Increasing the number of packets that send out may enlarge RTT, and further influence the latency of flows. | Trade off the number of sent out packets and the RTT can prevent TCP flow from increasing its latency. |
| When reordering rate in upload flow grows, the completion time of upload flows decreases greatly, which means the users tend to terminate streaming sharing if the network quality becomes bad. | Service provider could deploy front-end servers closer to streaming sharing users to optimize the upload performance, which may greatly improve the QoE for users. |
| Timeout retransmission contributes 29.8% of stalls in download process. | Costly timeout retransmissions in short flows could be eliminated through slightly aggressively retransmission strategies, e.g., Refs[17,18]. |
| The three types of frame experience different timeout retransmission characteristics due to their distinct frame production speeds and frame sizes., e.g., frame I experiences more continuous loss stalls while frame V experiences more less pkt retrans stalls. | Transmit each type of frame in separate connections, and optimize each connection by eliminating the timeout retransmissions according to the characteristics of frames and connections. |

As the upload performance is vital to the overall performance of the system, more front-end servers should be deployed in different locations and the nearest one with better network quality (i.e., lower delay and lower packet loss rate) should be selected for the video sharing users, for gaining a better upload performance.

Even though previous work[17,18] found the great degradation of performance of short flows due to timeout retransmission, it is further found that flows may exhibit distinct characteristics of timeout retransmission due to different frame production intervals and frame sizes. To eliminate these costly timeout retransmissions, one could adopt existing solutions like TLP[17], S-RTO[18]. Service provider could also optimize the performance by transmitting different frames in separate connections. Since the drop of a P or V frame can only hurt the watching experience slightly, P and V frames could be delivered in UDP connections and frame I is transmitted in TCP connections. Divide-and-conquer optimization of each type of connections may bring performance improvement of the overall system.

## 5 Conclusion

A commercial crowdsourced live streaming system is examined from the perspective of TCP performance of flows and the impact of video codec using packet-level TCP traces. The upload process is the key as it significantly impacts the throughput of streaming download flows, which is exemplified by the fact that 37% of stalls in upload flows trigger stalls in download flows. Moreover, the three types of frames result in different characteristics of timeout retransmission, a costly packet loss recover mechanism. The findings shed light on the design of high-performance crowdsourced live streaming systems.

**References**

[1] Ishimaru S, Kunze K, Kise K, et al. In the blink of an eye: combining head motion and eye blink frequency for activity recognition with google glass. In: Proceedings of the 5th Augmented Human International Conference, Kobe, Japan, 2014. 15-18

[2] Mukerjee M K, Hong J A, Jiang J, et al. Enabling near real-time central control for live video delivery in CDNs. *ACM SIGCOMM Computer Communication Review*, 2015, 44(4): 343-344

[3] Yin H, Liu X, Zhan T, et al. Design and deployment of a hybrid CDN-P2P system for live video streaming: experiences with LiveSky. In: Proceedings of the 17th ACM International Conference on Multimedia, Beijing, China, 2009. 25-34

[4] Xu Y, Yu C, Li J, et al. Video telephony for end-consumers: measurement study of Google +, iChat, and Skype. In: Proceedings of the 2012 ACM Conference on

Internet Measurement Conference (IMC), Boston, USA, 2012. 371-384
[5] Mukerjee M K, Naylor D, Jiang J, et al. Practical, real-time centralized control for CDN-based live video delivery. In: Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication (SIGCOMM), London, UK, 2015. 311-324
[6] Yin X, Jindal A, Sekar V, et al. A control-theoretic approach for dynamic adaptive video streaming over HTT. In: Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication (SIGCOMM), London, UK, 2015. 325-338
[7] Simoens P, Xiao Y, Pillai P, et al. Scalable crowd-sourcing of video from mobile devices. In: Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys), Taipei, China, 2013. 139-152
[8] Zhang C, Liu J. On crowdsourced interactive live streaming: a Twitch. tv-based measurement study. In: Proceedings of the 25th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV), Porland, USA, 2015. 55-60
[9] Jain P, Manweiler J, Acharya A, et al. FOCUS: clustering crowdsourced videos by line-of-sight. In: Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems (SenSys), Roma, Italy, 2013. 90-104
[10] Chen F, Zhang C, Wang F, et al. Crowdsourced live streaming over the cloud. In: Proceedings of the IEEE Conference on Computer Communications (INFOCOM), Hong Kong, China, 2015. 2524-2532
[11] Stohr D, Li T, Wilk S, et al. An analysis of the YouNow live streaming platform. In: Proceedings of the IEEE Conference on Local Computer Networks (LCN), Dubai, UAE, 2015. 673-679
[12] Schwarz H, Marpe D, Wiegand T. Overview of the scalable video coding extension of the H. 264/AVC standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 2007, 17(9): 1103-1120
[13] Ganjam A, Siddiqui F, Zhan J, et al. C3: Internet-scale control plane for video quality optimization. In: Poceedings of the 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI), Oakland, USA, 2015. 131-144
[14] Seeling P, Reisslein M. Video transport evaluation with H. 264 video traces. *Communications Surveys & Tutorials*, *IEEE*, 2012, 14(4): 1142-1165
[15] Mittal R, Sherry J, Ratnasamy S, et al. Recursively cautious congestion control. In: Proceedings of the 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI), Seattle, USA, 2014. 373-385
[16] Vamanan B, Hasan J, Vijaykumar T N. Deadline-aware datacenter tcp (d2tcp). *ACM SIGCOMM Computer Communication Review*, 2012, 42(4): 115-126
[17] Flach T, Dukkipati N, Terzis A, et al. Reducing web latency: the virtue of gentle aggression. In: Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM), New York, USA, 2013. 159-170
[18] Zhou J, Wu Q, Li Z, et al. Demystifying and mitigating TCP stalls at the server side. In: Proceedings of the 11th International Conference on Emerging Network Experiments and Technologies (CoNEXT), Herdelberg, Germany, 2015. 99-112

**Zhou Jianer**, born in 1986. He is now a forth year Ph. D student from Institute of Computing Technology, Chinese Academy of Sciences. He received his M. S and B. S degrees both from Chongqing University of Post and Telecommunication. His research interest is Internet architecture, network measurement.