Non bandwidth-intrusive video streaming over TCP

Hiroyuki Hisamatsu
Osaka Electro-Communication University

Background

- Current video streaming services utilize TCP
  - i.e.: YouTube, nicovideo, Dailymotion
- TCP is not suitable for video streaming
  - Greedy congestion control
    - Tries to exhaust the entire bandwidth

    ⇒ Increases its transfer rate regardless of the video playback rate

    ⇒ Takes the bandwidth from other competing traffic unnecessarily

Objective

- Investigate the characteristics of the data transfer of current Video streaming over TCP
  - They transfer video at a much higher rate than the video playback rate
- Propose a new data transfer mechanism for Video streaming
  - It controls data transfer at an application layer
- Show propose mechanism works effectively by simulation experiments
  - It avoids excessively taking bandwidth from competing traffic

Investigation of current video streaming over TCP

- Examined YouTube and nicovideo
  - nicovideo has the same kind of problem as that of YouTube
- Observed data transfer at a packet level using *tcpdump* at a receiver
- Video Sequences
  - Playback time was 10 [m]
  - Quality was 1080p
    - Playback rate was about 3.6 [Mbit/s]

Summary of Investigation Results

- Found two mechanisms
  - Mechanism(i) has two phase
    - First phase: beginning in the data transfer, average transfer rate 43.4 [Mbit/s] >> 3.6 [Mbit/s]
    - Second phase: after the first phase to the end average transfer rate 6.13 [Mbit/s] > 3.6 [Mbit/s]
  - Mechanism(ii) has no special control
    - Server sends data video data at an high rate from the beginning to the end
    - Average transfer rate 45.1 [Mbit/s] >> 3.6 [Mbit/s]

Video Playback Rate

Transmits video data at a rate far beyond what is necessary

Outline of Proposed Mechanism

- Assumptions
  - An application program is installed at the sender and receiver
  - An application program can acquire TCP state variables
    - Easily possible to acquire them by using web100 kernel
- Operations
  - The receiver notifies the sender of the amount of buffered video data, b_{dst}
  - The sender estimates the network congestion level, cl
  - The sender calculates the amount of video data , b_{tgt}, to avoid buffer underflow and playback interruption based on the cl
  - The sender determines the amount of data passed to TCP based on difference b_{dst} and b_{tgt}

Control of the proposed mechanism operates in the unit of one RTT.

Congestion Level Estimation

- Estimate the number of packets queued throughout the network
  - Index of the network congestion level
- Network congestion level *cl(i)*:

$$cl(i) = \frac{baseRTT(i-1)}{MSS}\left(\frac{apwnd(i-1)}{baseRTT(i-1)} - \frac{apwnd(i-1)}{RTT(i-1)}\right)$$

Application–level Window Size

Maximum Segment Size

Minimum Round-Trip Time

Current Round-Trip Time

Controlling Transfer Rate

Application-level window size: amount of data passed to TCP in one RTT

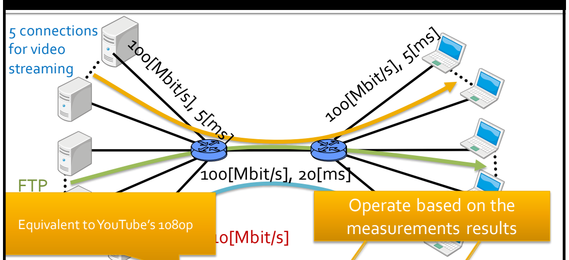
$b_{tgt}(i)$: target data size to be buffered
$b_{dst}(i)$: buffered data size at the receiver
rate: video playback rate

$$apwnd(i) = \min\left(\max\left(RTT_{max}(i-1)\cdot rate + b_{tgt}(i) - b_{dst}(i), MSS\right), 2apwnd(i-1)\right)$$

Minimum window size. To prevent reseing of TCP congestion window

Maximum window size. To keep from increasing transfer rate too rapidly

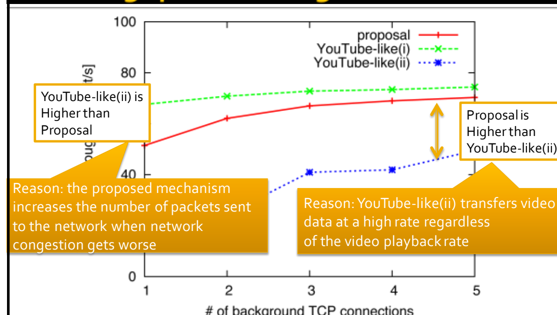
Simulation Model



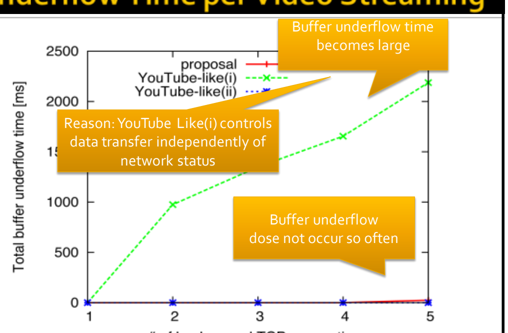
Video Sequence: Video Playback Rate 3.6 [Mbit/s] Playback Time 10 [m]

Compare proposal, YouTube-like(i), and YouTube-likes(ii)

Simulation Results: Average Total Throughput of Background Traffic



Simulation Results: Average Buffer Underflow Time per Video Streaming



Conclusion and Future Works

- Conclusion
  - Investigate data transfer mechanisms of the current video streaming services using TCP
    - Show transfer rate is much higher than video playback rate
  - Propose a new data transfer mechanism to resolve this problem
    - Controls data transfer at an application-layer
  - Simulation results show proposed mechanism
    - suppresses the occurrence of buffer underflow
    - dose not unnecessarily divert bandwidth from background traffic
- Future works
  - Evaluate the performance of the proposed mechanism in a real network
  - Extend the proposed mechanism
    - Operate solely by a sender-side application

Thank You ! & Question ?