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]

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 websoc kernel
  - Operations
    - The receiver notifies the sender of the amount of buffered video data, $h_{buf}$
    - The sender estimates the network congestion level, $cl$
    - The sender calculates the amount of video data, $h_{tgt}$, to avoid buffer underflow and playback interruption based on the $cl$
    - The sender determines the amount of data passed to TCP based on difference $h_{buf}$ and $h_{tgt}$

Transmits video data at a rate far beyond what is necessary
**Congestion Level Estimation**

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

$$c_l(i) = \frac{b_{data}(i)}{MSS} \left( \frac{apend(i) - b_{data}(i)}{baseRTT(i-1)} \right)$$

**Controlling Transfer Rate**

- Application-level window size: amount of data passed to TCP in one RTT
- $b_{data}(i)$: target data size to be buffered
- $b_{buf}(i)$: buffered data size at the receiver
- $rate$: video playback rate

$$apend(i) = \min \left( \frac{RTT_{max}(i-1) \cdot rate + b_{data}(i)}{RT(i-1)} \right)$$

**Simulation Model**

- 5 connections for video streaming
- 100(Mbit/s), 20(sec)
- FTP
- Equivalent to YouTube's software

**Simulation Results: Average Total Throughput of Background Traffic**

- YouTube-like(i) is higher than Proposal
- Proposal is higher than YouTube-like(ii)

**Simulation Results: Average Buffer Underflow Time per Video Streaming**

- Buffer underflow time becomes large
- Buffer underflow does not occur so often

**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
  - Does 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 ?