Optimizing Live Game Streaming Platforms Using Segment-of-Interests

Tao-Ya Fan-Chiang

NMSL, Department of Computer Science, National Tsing Hua University, Taiwan

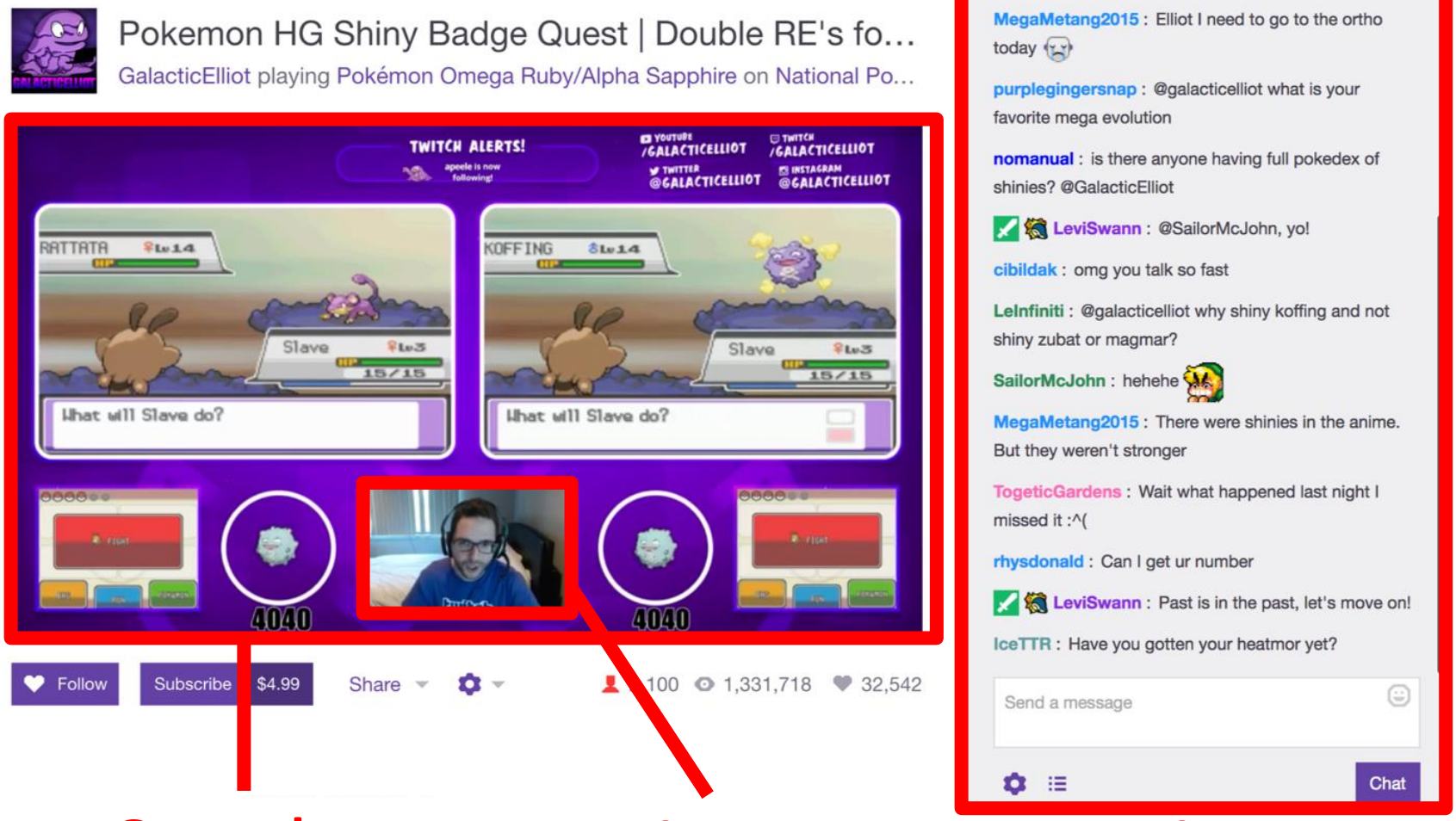
Outline

- Introduction & Motivation
- System Overview
- SoI Detector
- Resource Allocator
- Evaluation
- Conclusion

Outline

- Introduction & Motivation
- System Overview
- SoI Detector
- Resource Allocator
- Evaluation
- Conclusion

Twitch

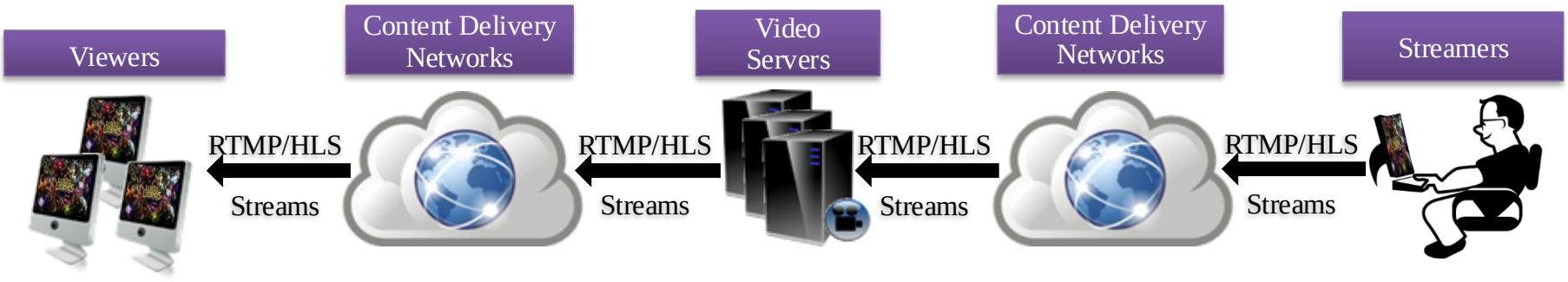


Gameplay

Webcam image

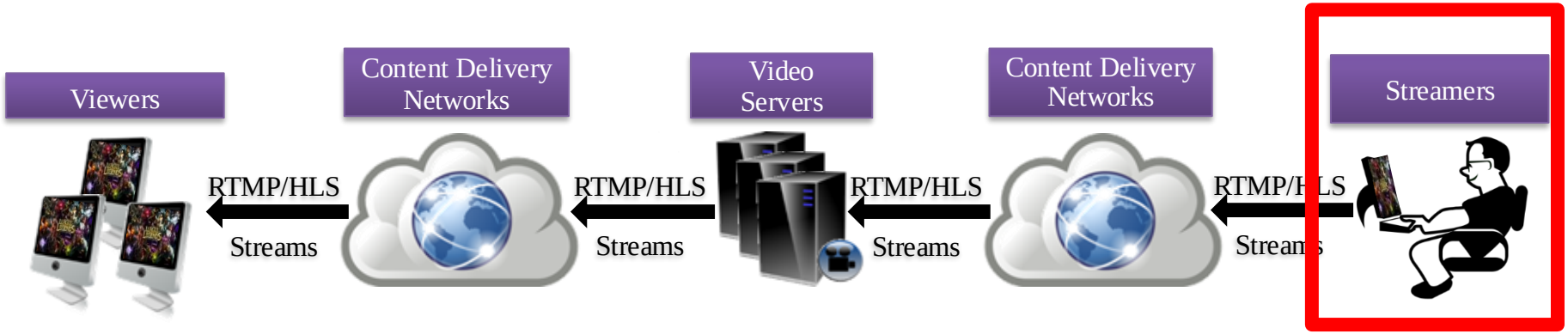
Chat

How Does Live Game Streaming Work?



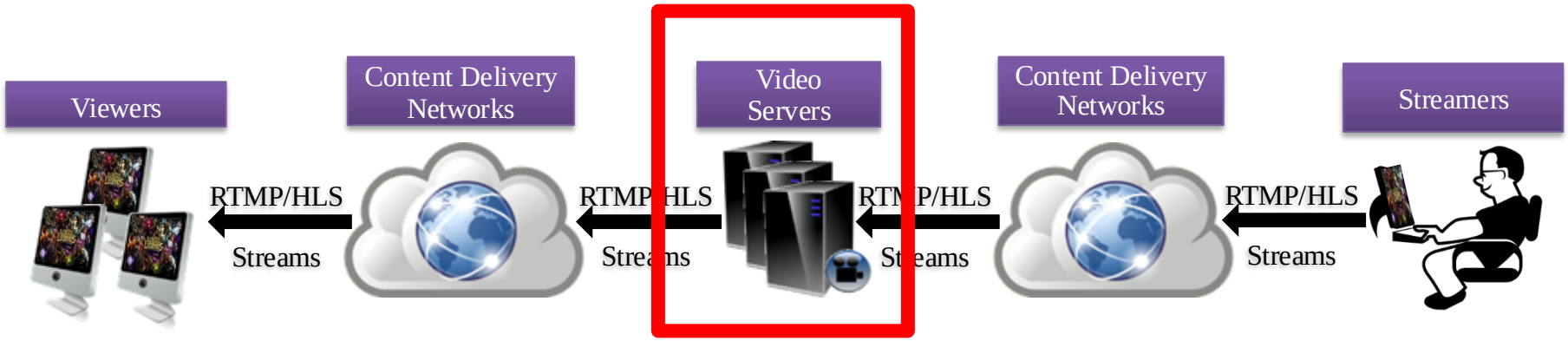
Streamer

- Streamer: send the game stream to video servers
  - Often with streaming software such as OBS
  - Using protocols such as RTMP, HTTP Live Streaming



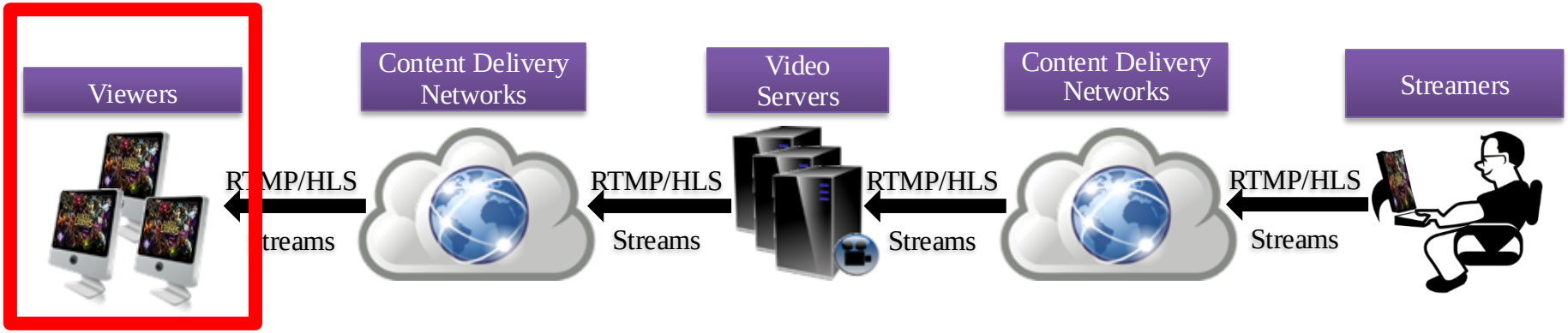
Video Servers

- Relay the stream to viewers
  - Transcode them if necessary



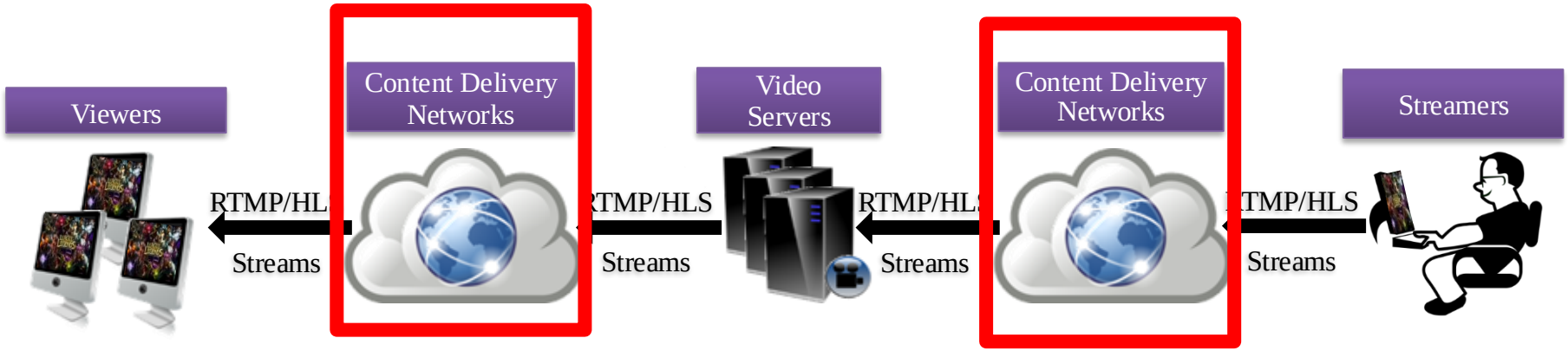
Viewers

- Watch the stream with video players



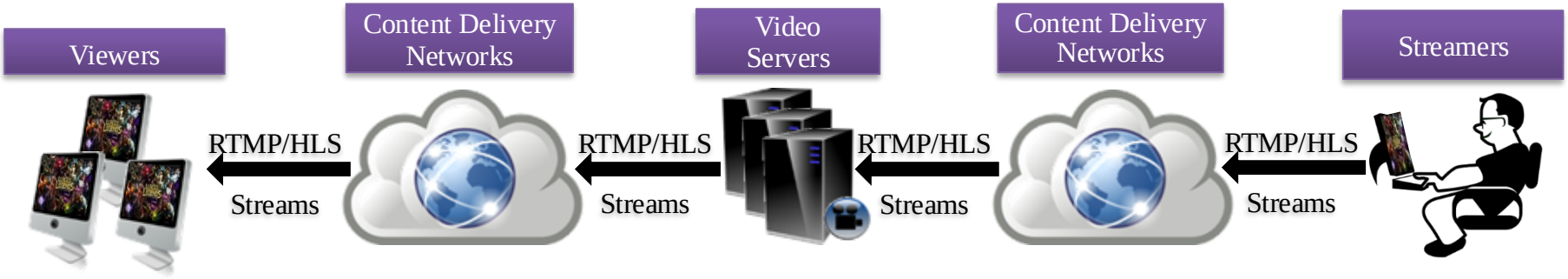
Content Delivery Networks

- Streams often go through CDNs
  - Companies such as Twitch need to pay for the bandwidth (to CDN companies such as Akamai)



Content Delivery Networks

- Streams often go through CDNs
  - Companies such as Twitch need to pay for the bandwidth (to CDN companies such as Akamai)



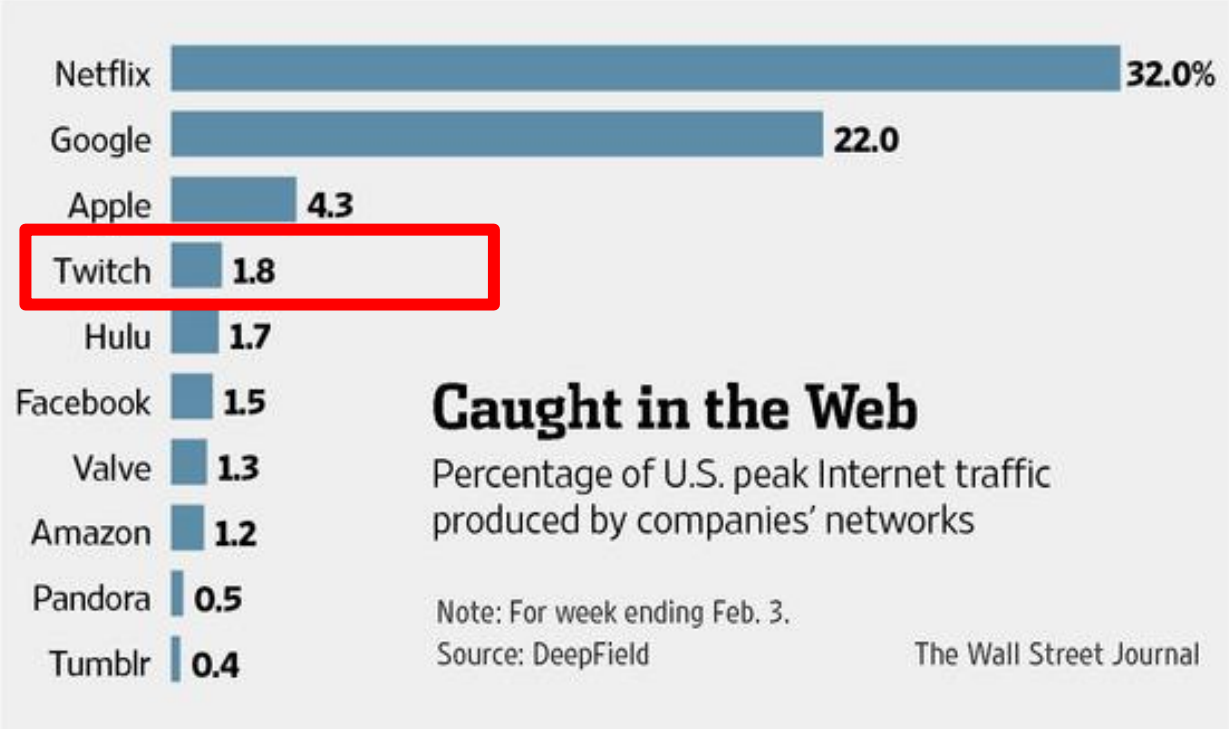
Who are running these services?

Who Are Running These Services?

- Twitch
  - Spin-off from Justin.tv
  - The broadcast platform for most E-sport event
- Ustream
  - The streaming service used by Sony PS4
- YouTube Gaming
  - Launched in Aug, 2015

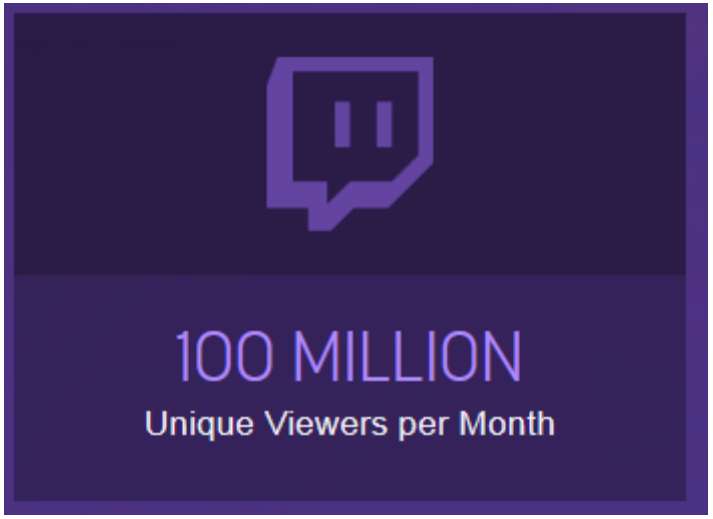
Twitch Is Hot

- 4th largest in peak US Internet traffic



Growing Demands

- Number of viewers doubled in 2014
  - More than twice in 2013



2013 **2014**

Acquired by Amazon.com for $970 million

Everything looks great

WHAT'S THE PROBLEM?

Problem: Lag

- Lag is annoying when watching live streams
  - Rebuffering
  - Other visual artifacts
  - Skipped video segment
- Bad for user experience

Possible Solutions

- Put in more resource
  - Cost money
- Lower the video quality
  - Bad for user experience

Possible Solutions

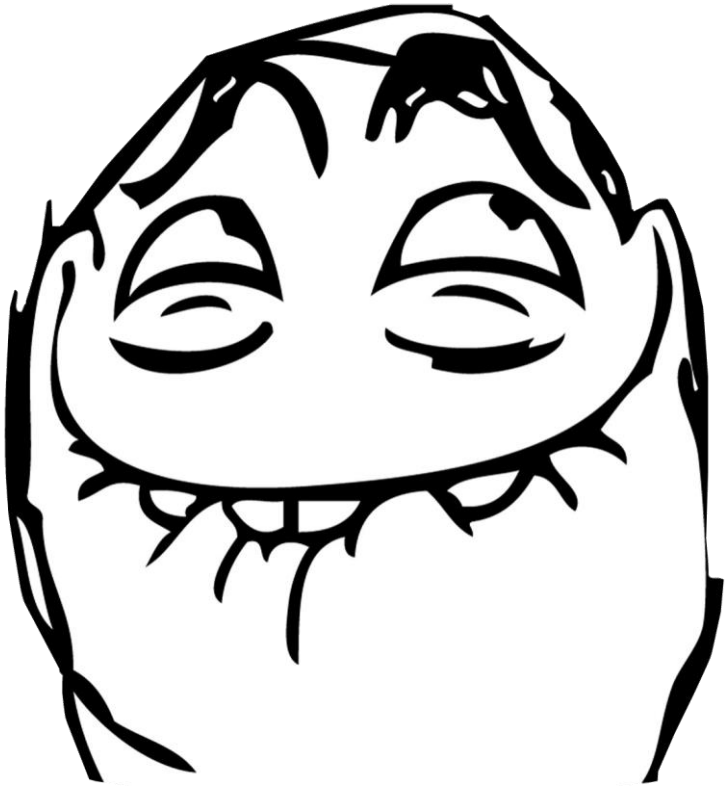
- Put in more resource
  - Cost money
- Lower the video quality
  - Bad for user experience

How to do this without degrading the user experience?

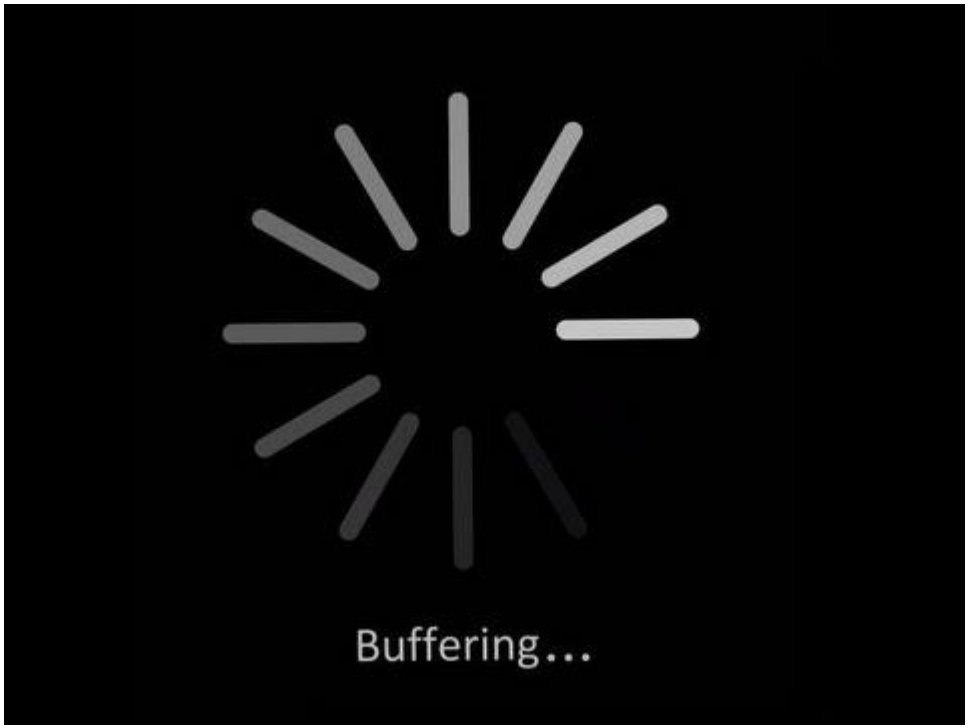
LAG AT DIFFERENT TIME CAUSE DIFFERENT LEVEL OF DEGRADATION IN USER EXPERIENCE

At Clutch Time

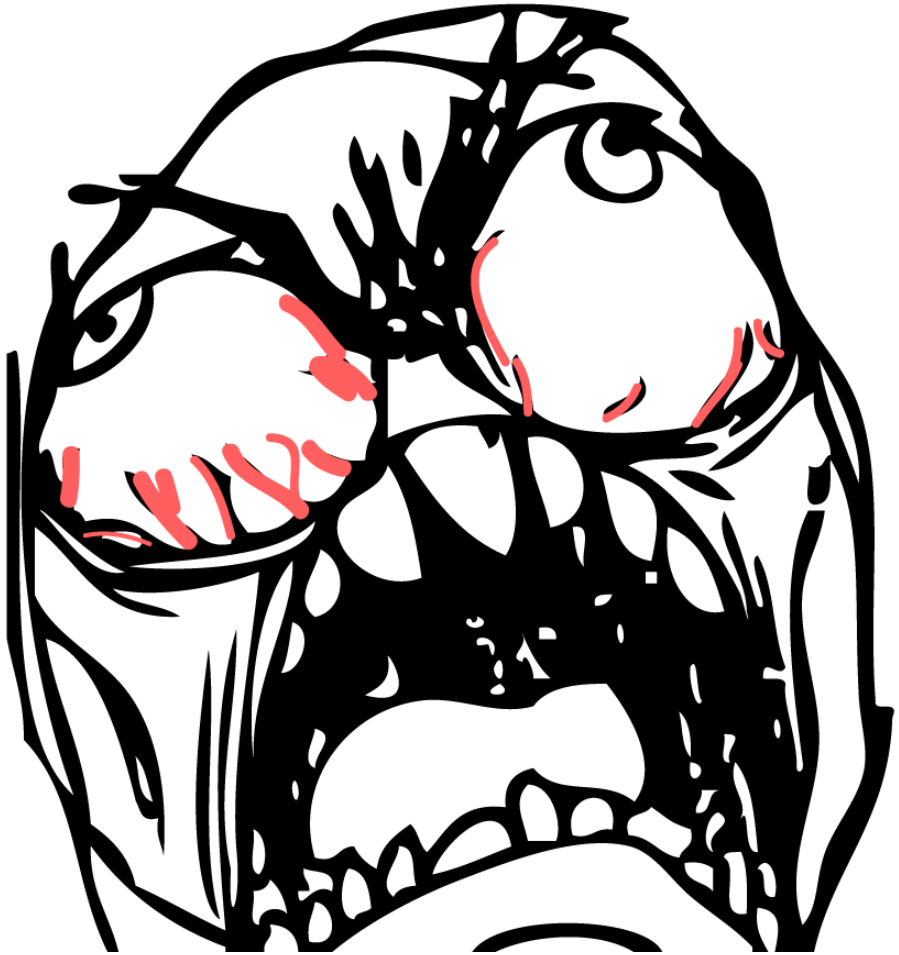
At Clutch Time



At Clutch Time

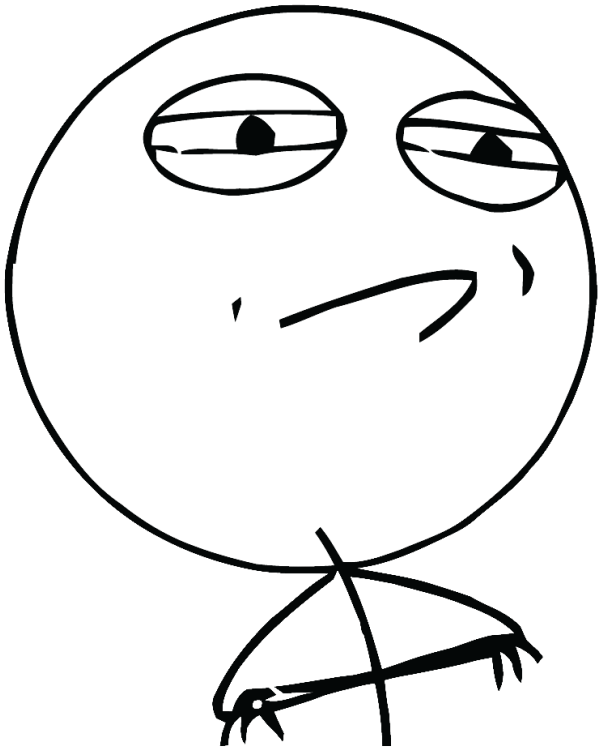


At Clutch Time

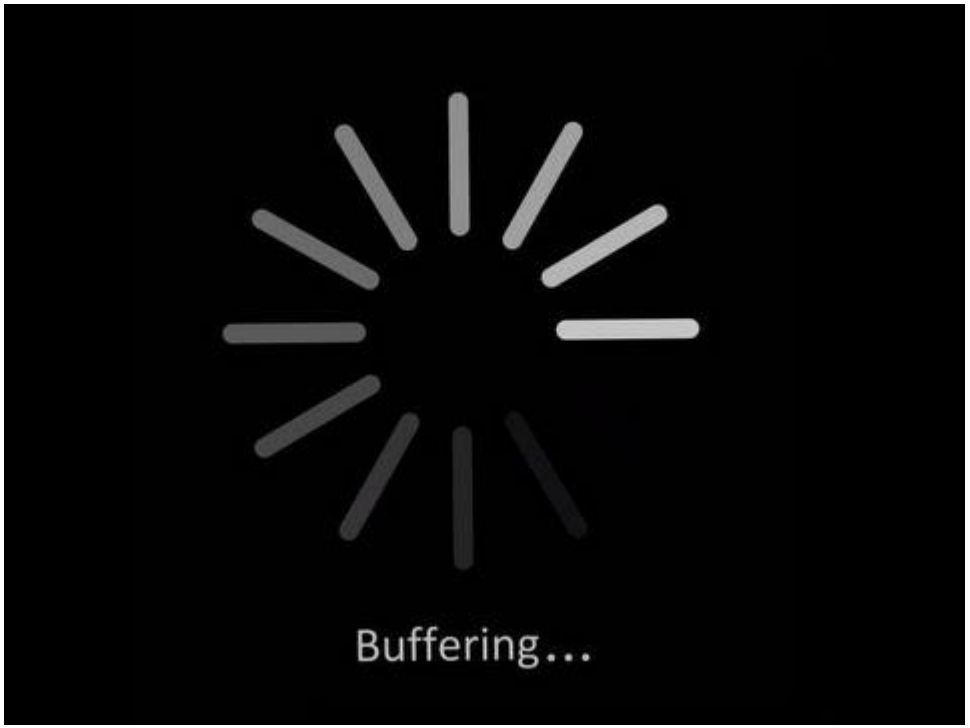


At Normal Time

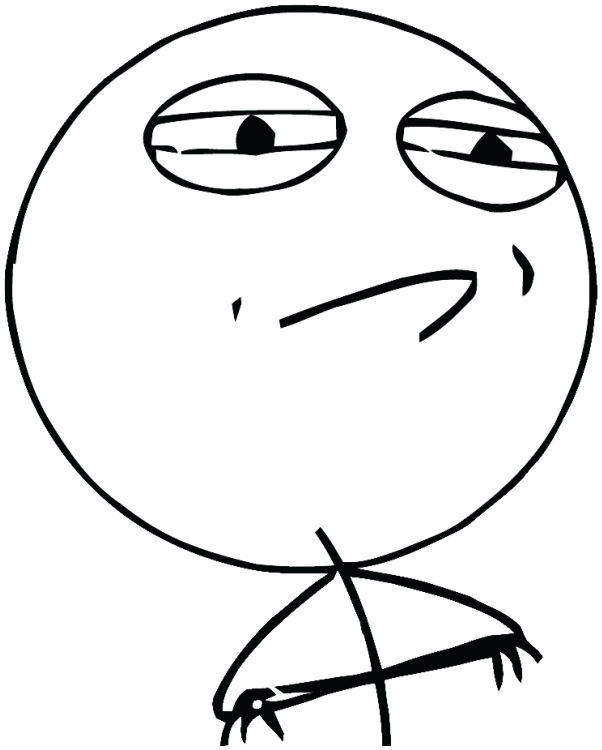
At Normal Time



At Normal Time



At Normal Time



Not that bad

Concept of Segment of Interest (SoI)

- Different segments have different importance
  - We refer to the important one as Segment of Interest (SoI)
- We should put in more resource to deliver SoI
  - Sacrifice non-SoI if we have to

Contributions

- Develop SoI-driven streaming platform
- Answer two questions
  - How to detect SoI efficiently?
  - How to allocate resources among channels?

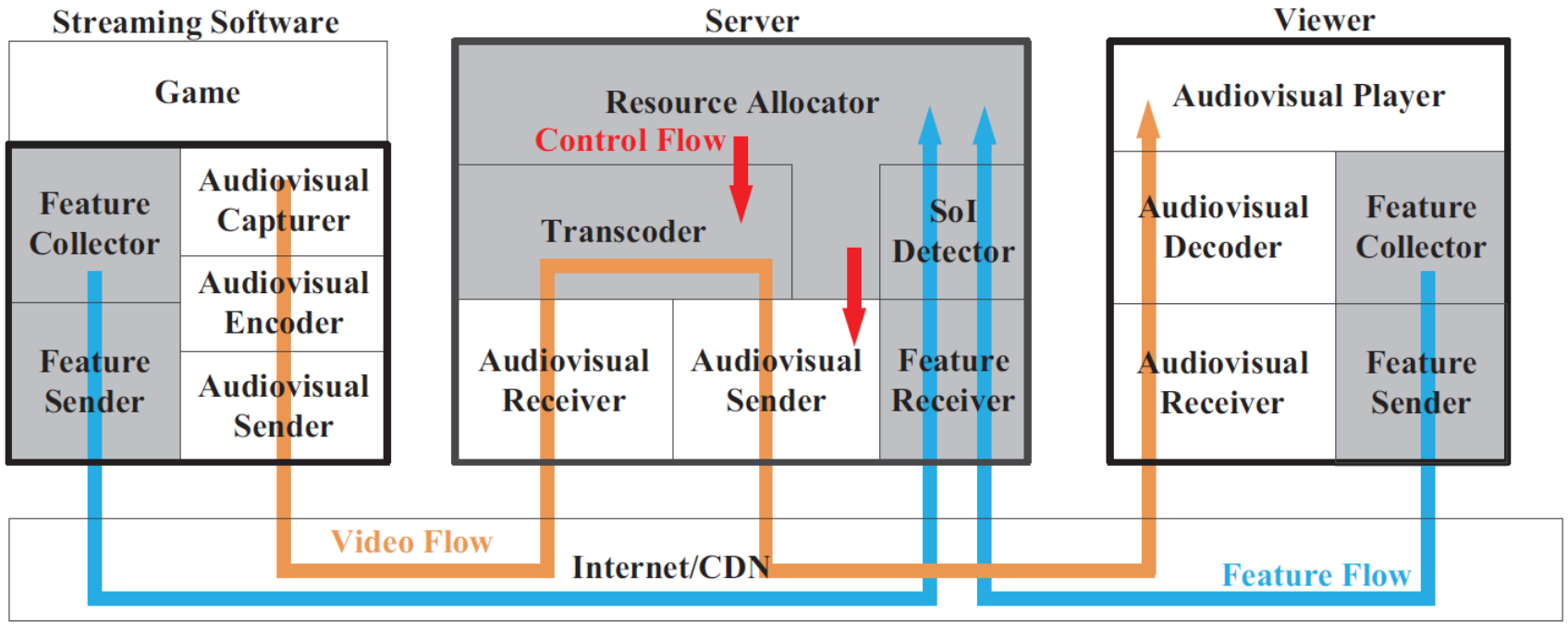
Outline

- Introduction & Motivation
- System Overview
- SoI Detector
- Resource Allocator
- Evaluation
- Conclusion

Outline

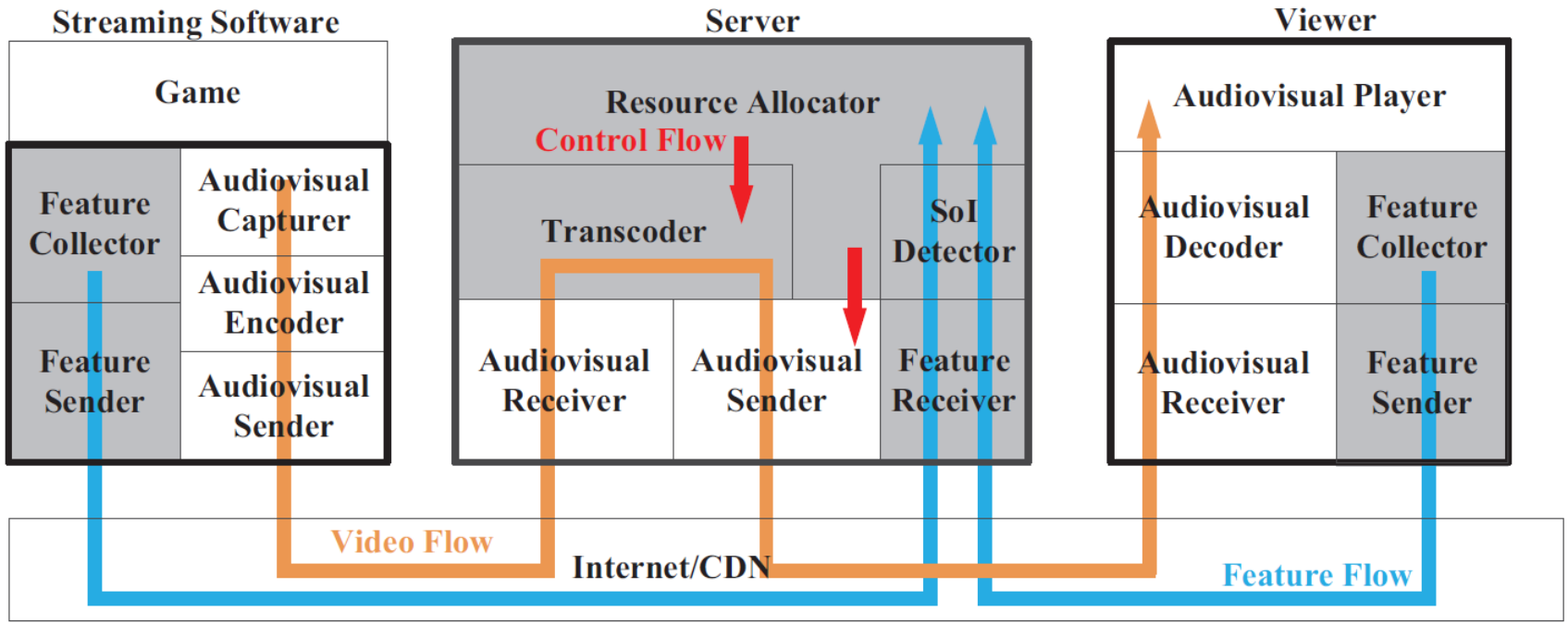
- Introduction & Motivation
- System Overview
- SoI Detector
- Resource Allocator
- Evaluation
- Conclusion

System Architecture



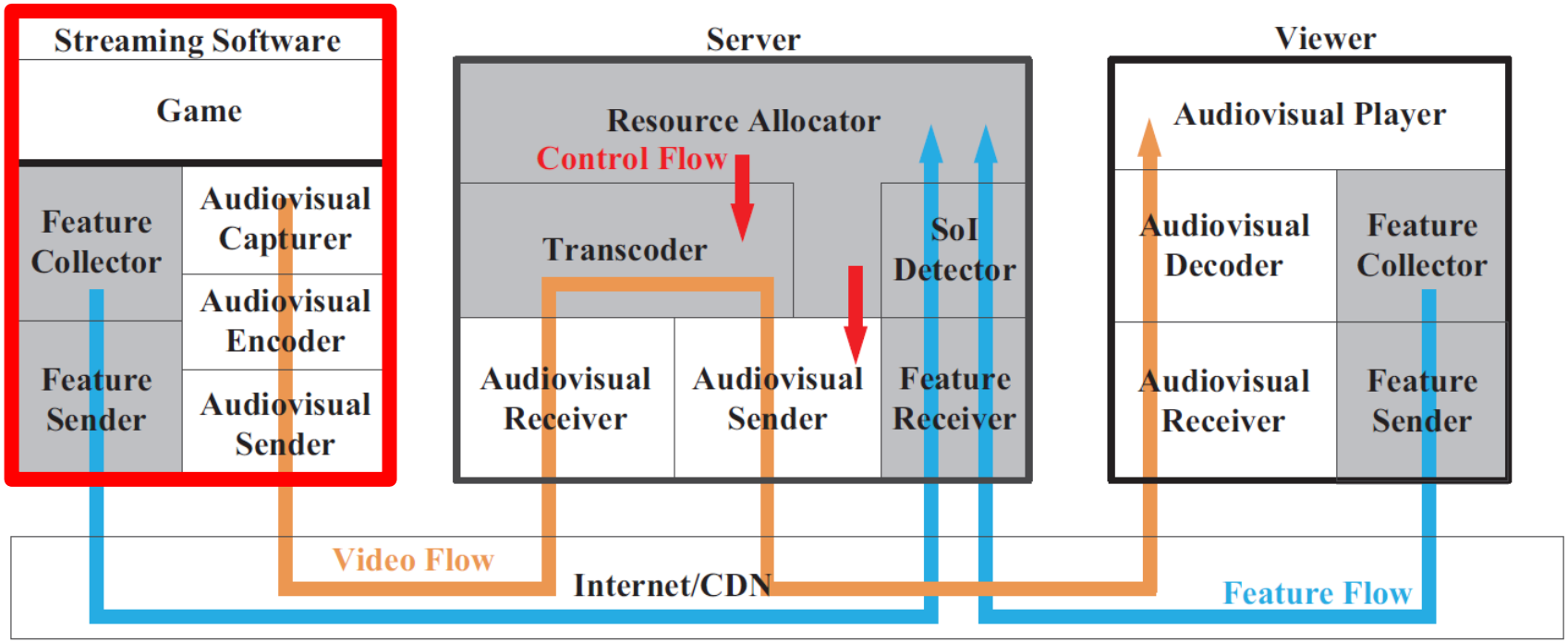
Streaming software

- Stream the gameplay to server



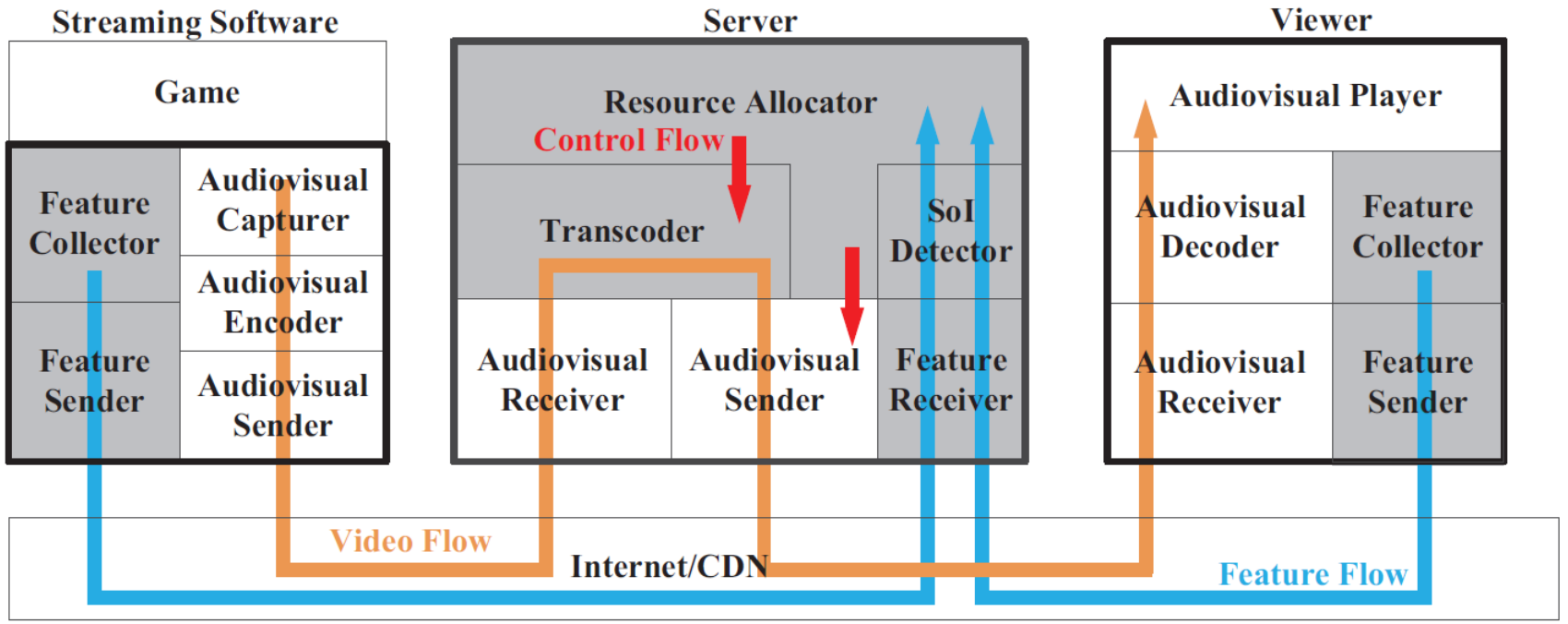
Streaming software

- Stream the gameplay to server



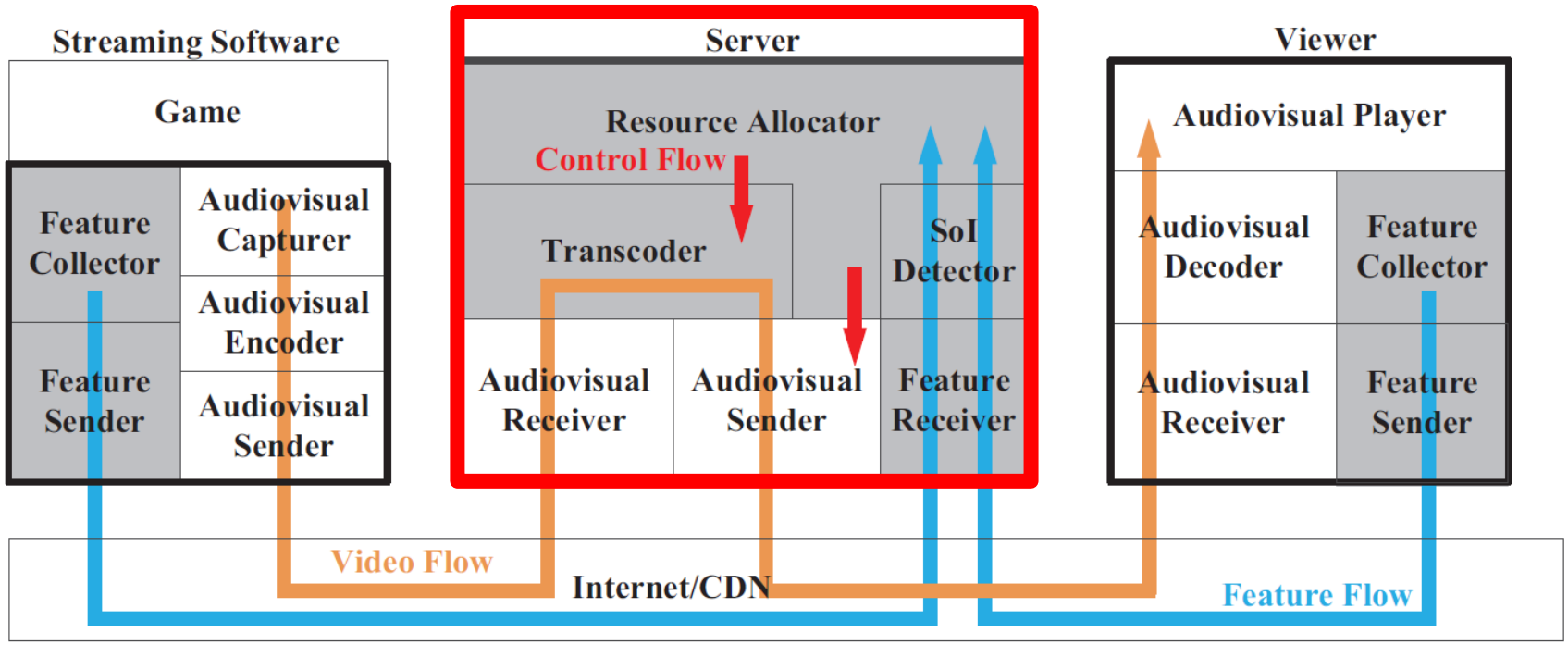
Streaming Server

- Relay the stream to viewers
- Transcode if necessary



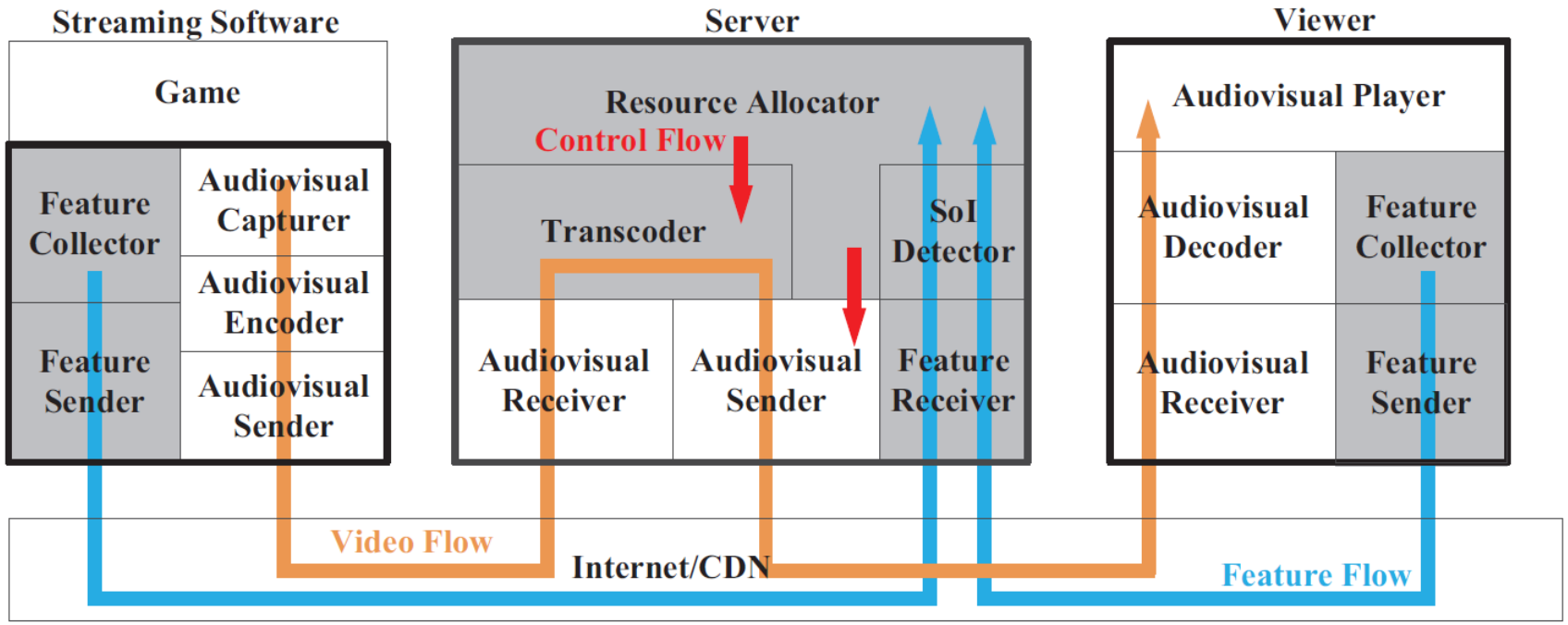
Streaming Server

- Relay the stream to viewers
- Transcode if necessary



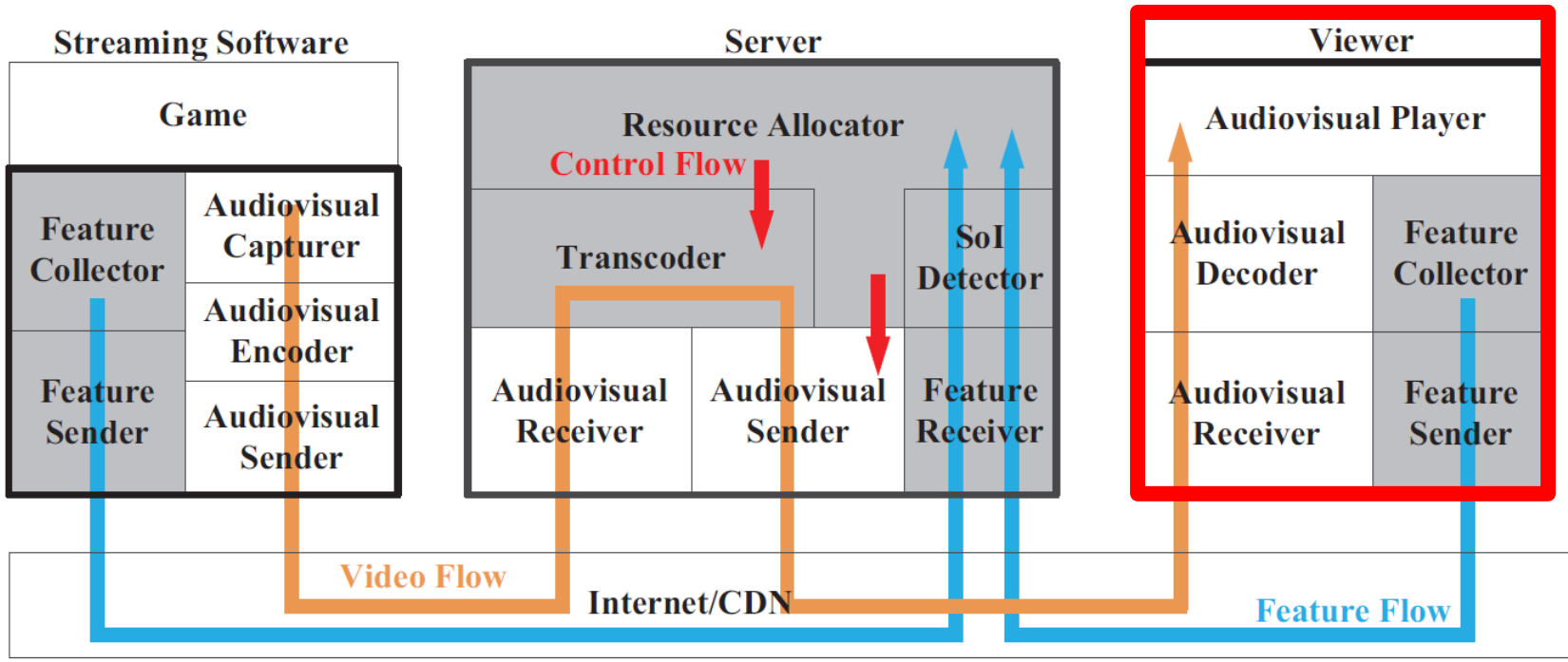
Viewer

- Replay the gameplay stream



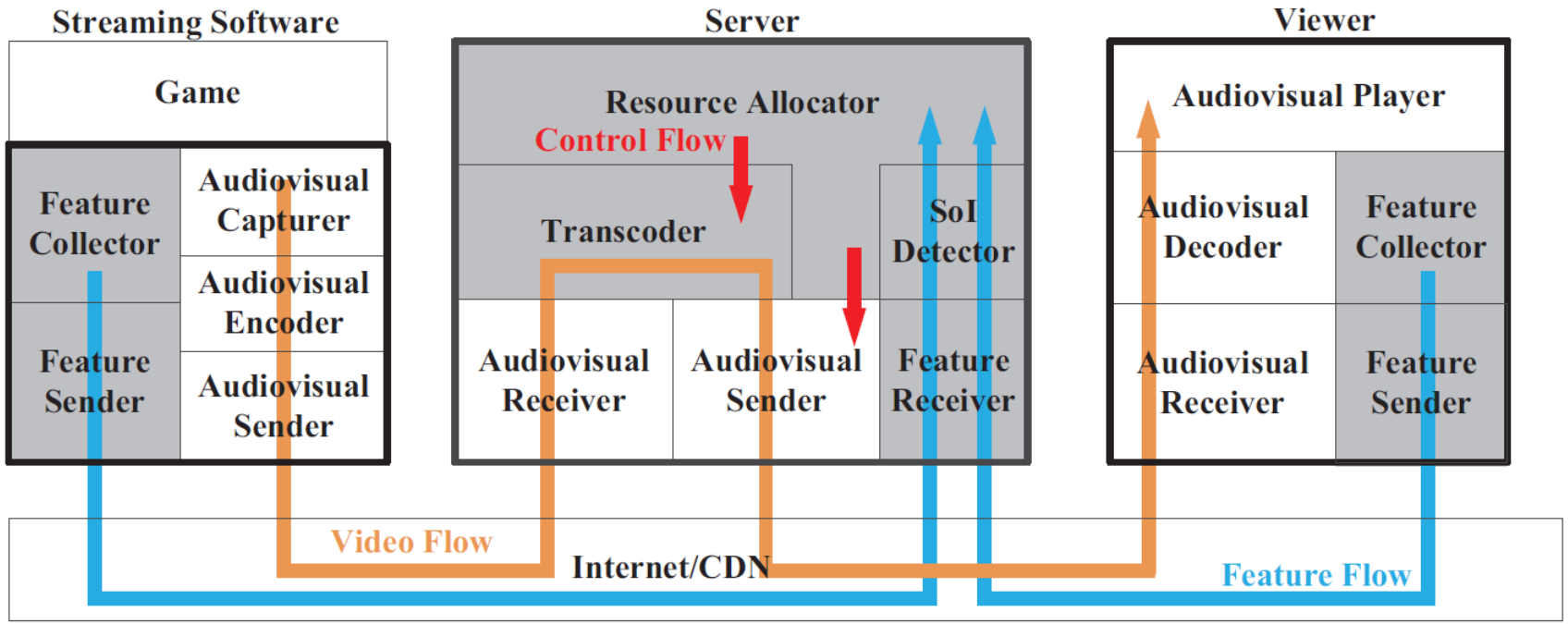
Viewer

- Replay the gameplay stream



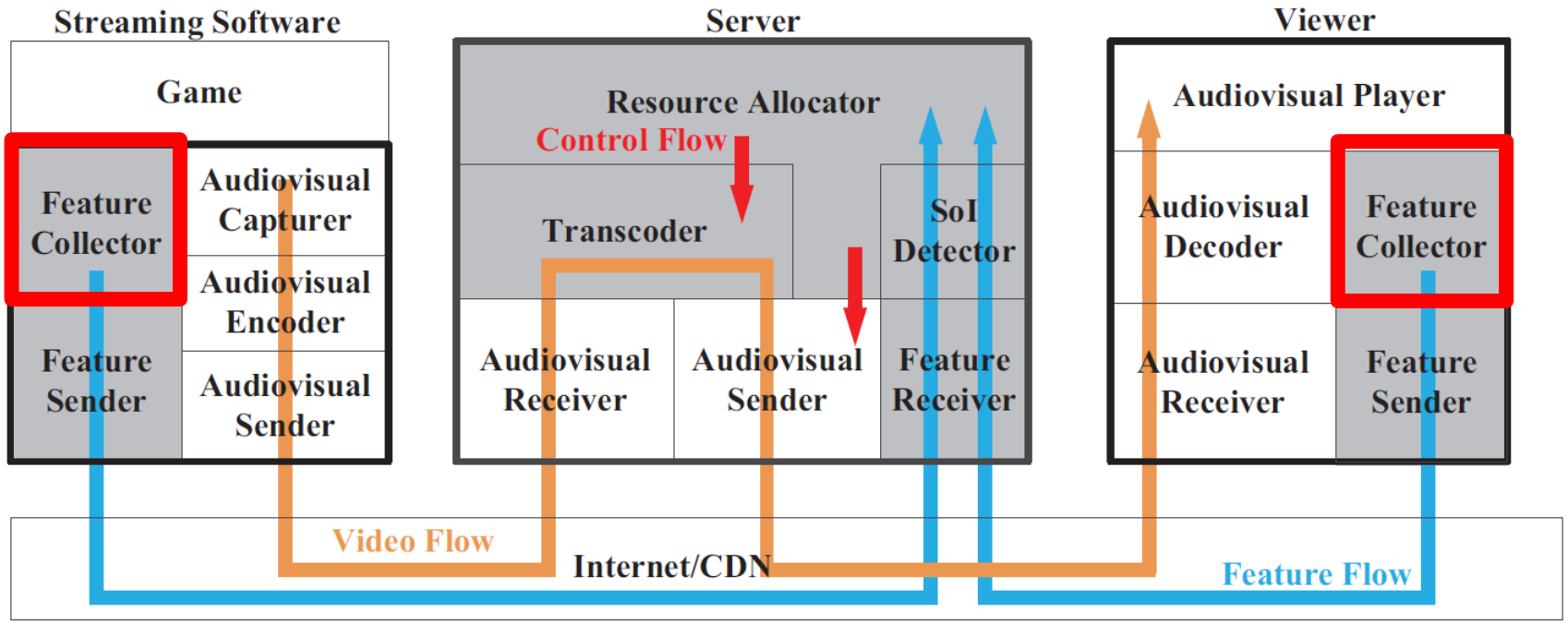
Feature Collector

- Collect the features from streamers
  - Such as CPU usage, foreground window name…
- Also from viewers



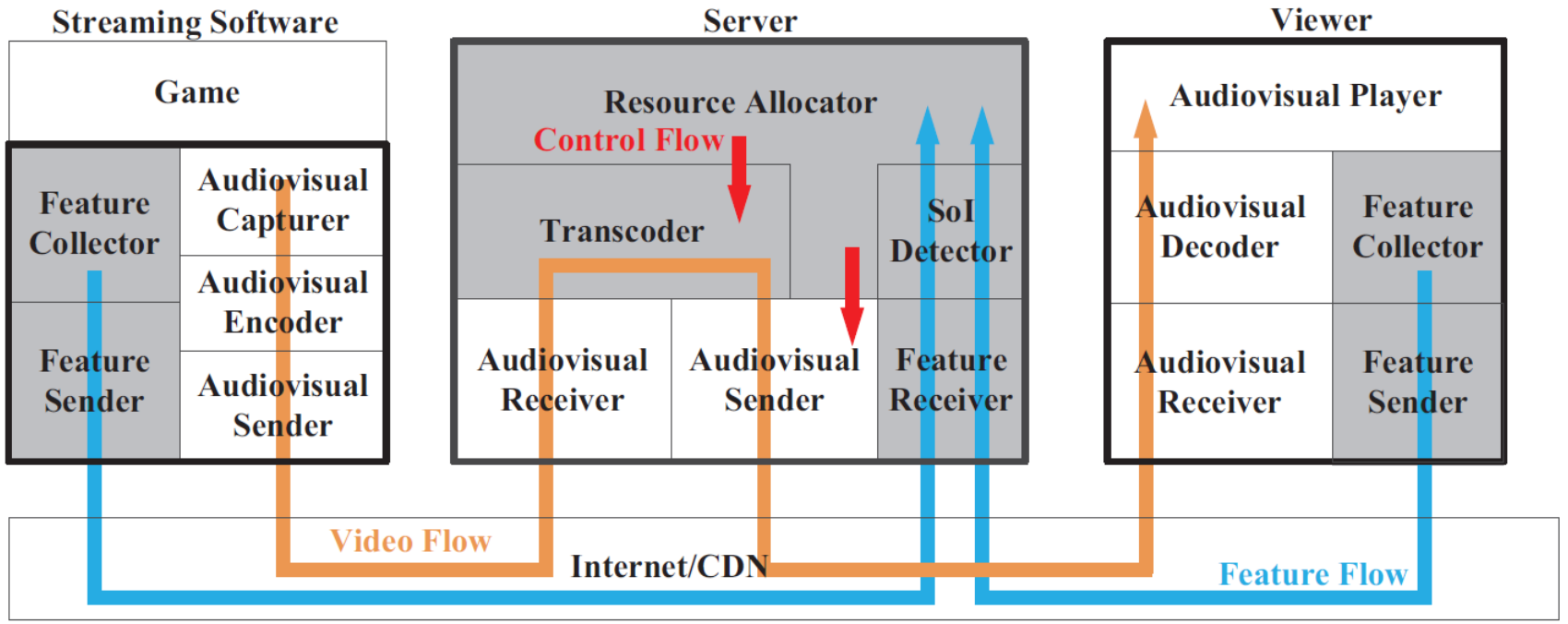
Feature Collector

- Collect the features from streamers
  - Such as CPU usage, foreground window name…
- Also from viewers



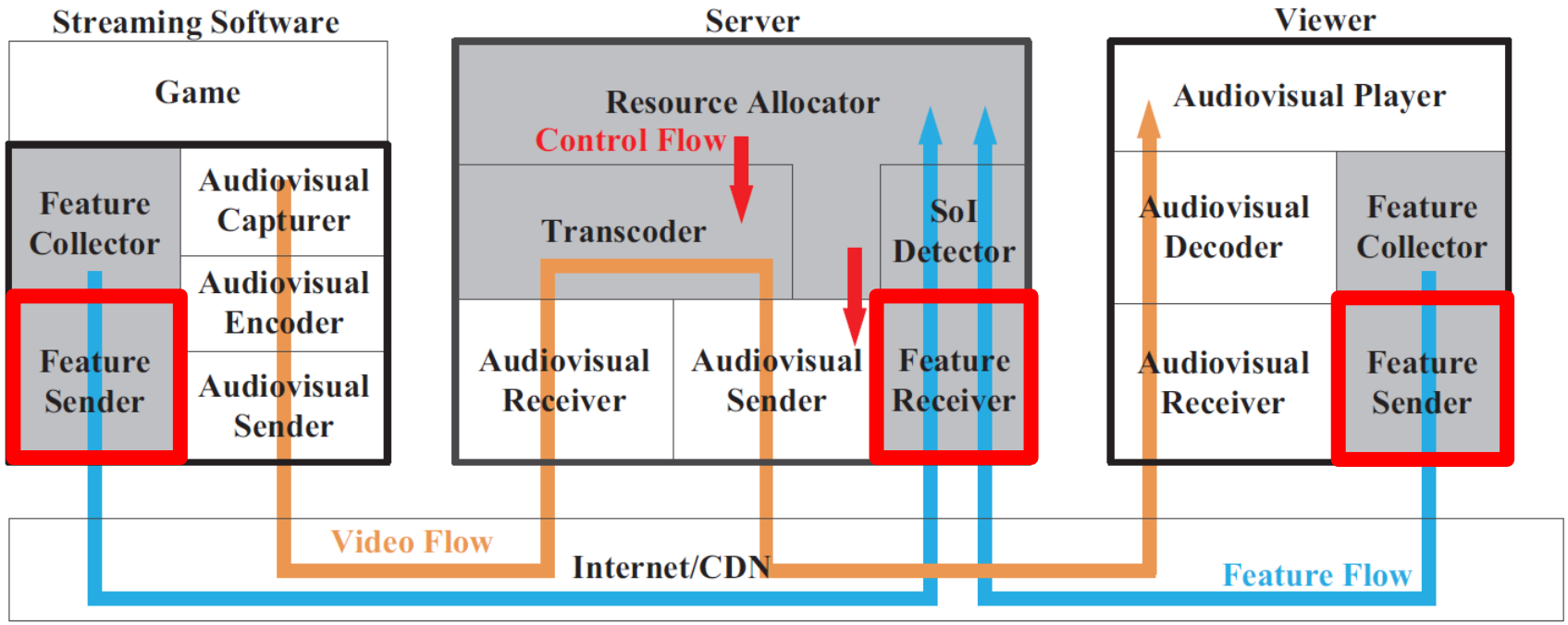
Feature Sender/Receiver

- Send/receive the collected features



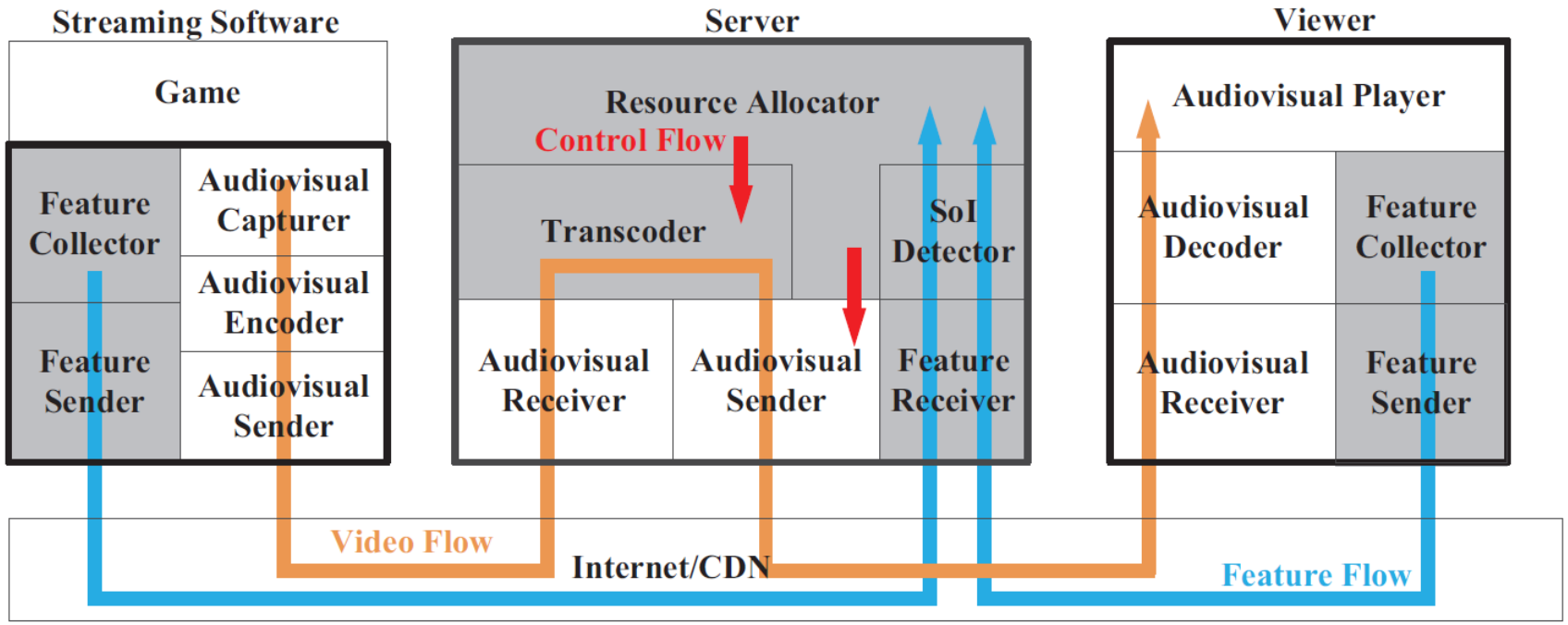
Feature Sender/Receiver

- Send/receive the collected features



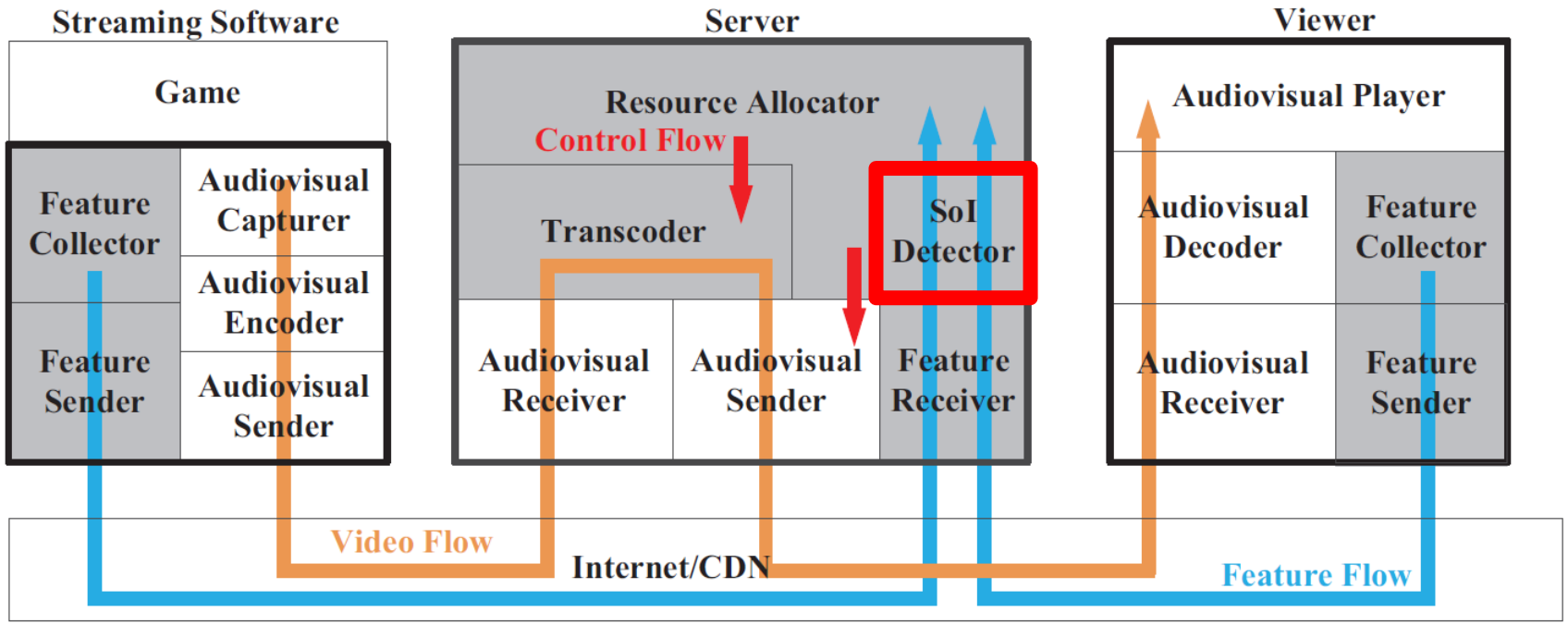
SoI Detector

- Detect SoI from the collected features



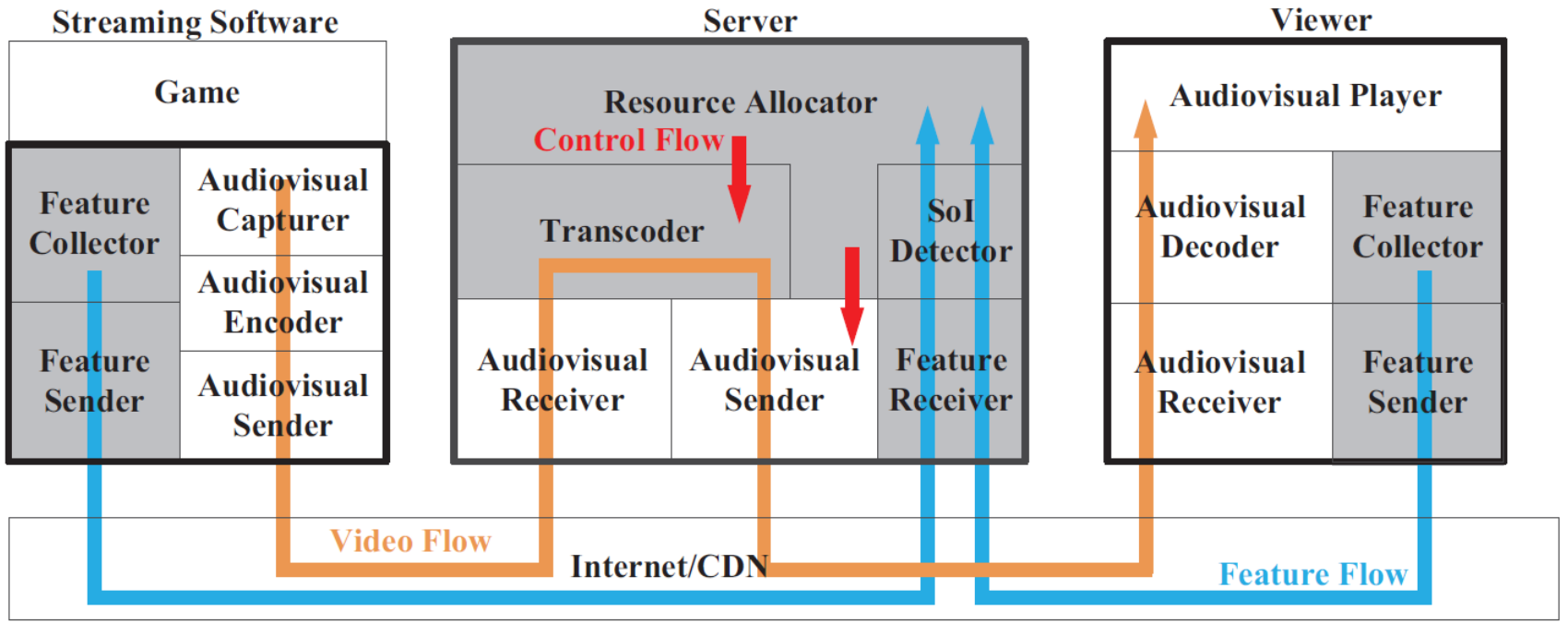
SoI Detector

- Detect SoI from the collected features



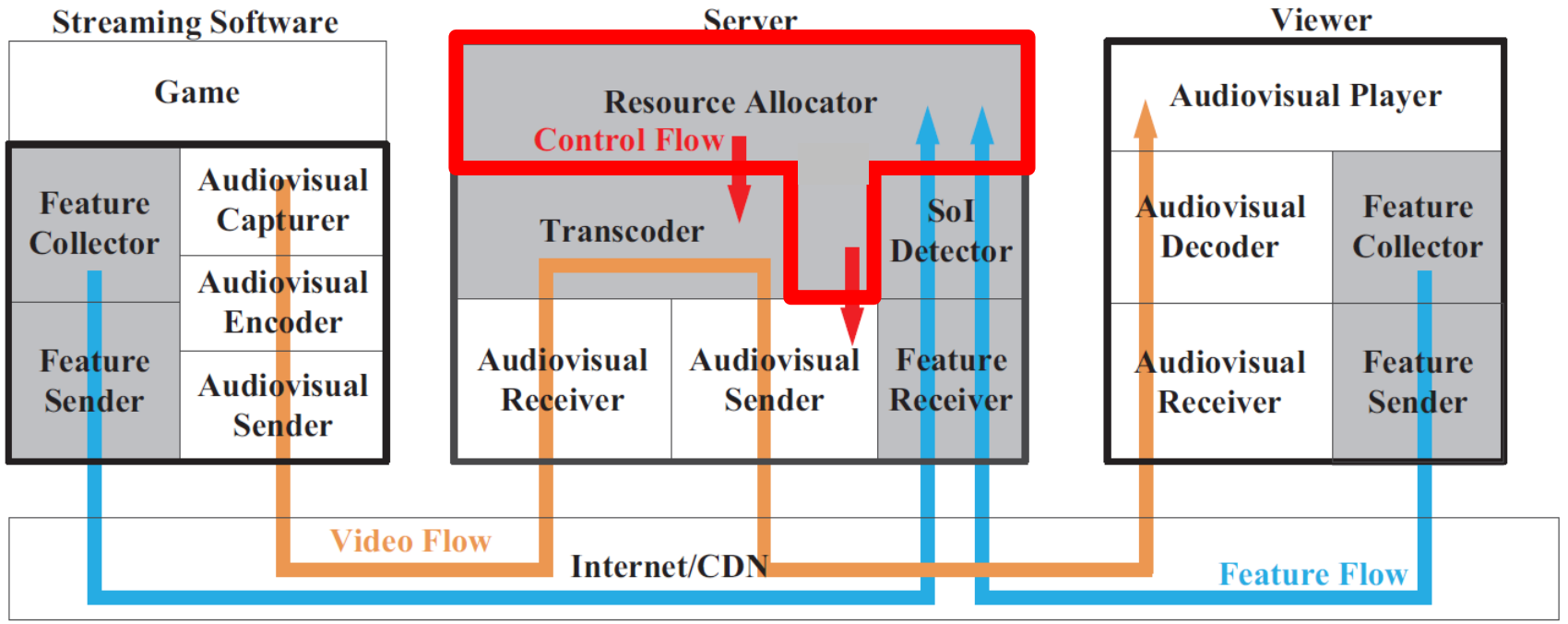
Resource Allocator

- Allocate resources between different streams in the system



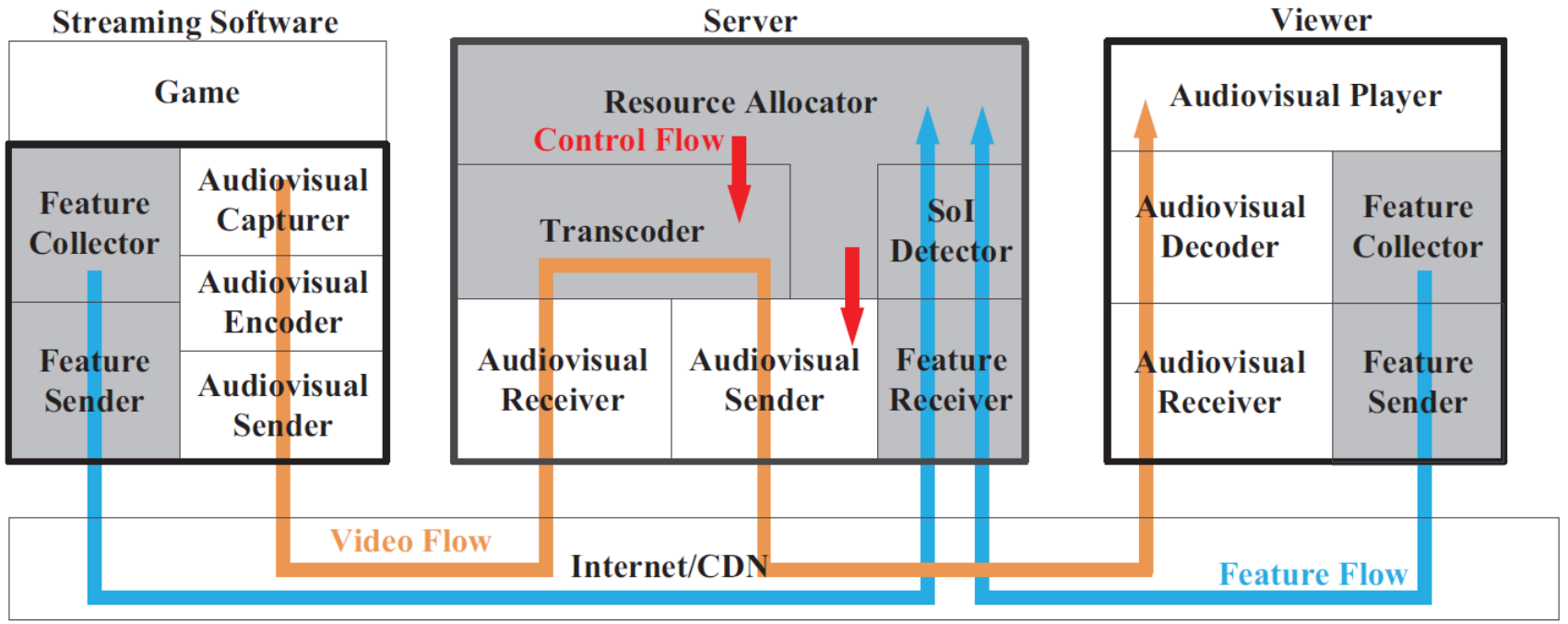
Resource Allocator

- Allocate resources between different streams in the system



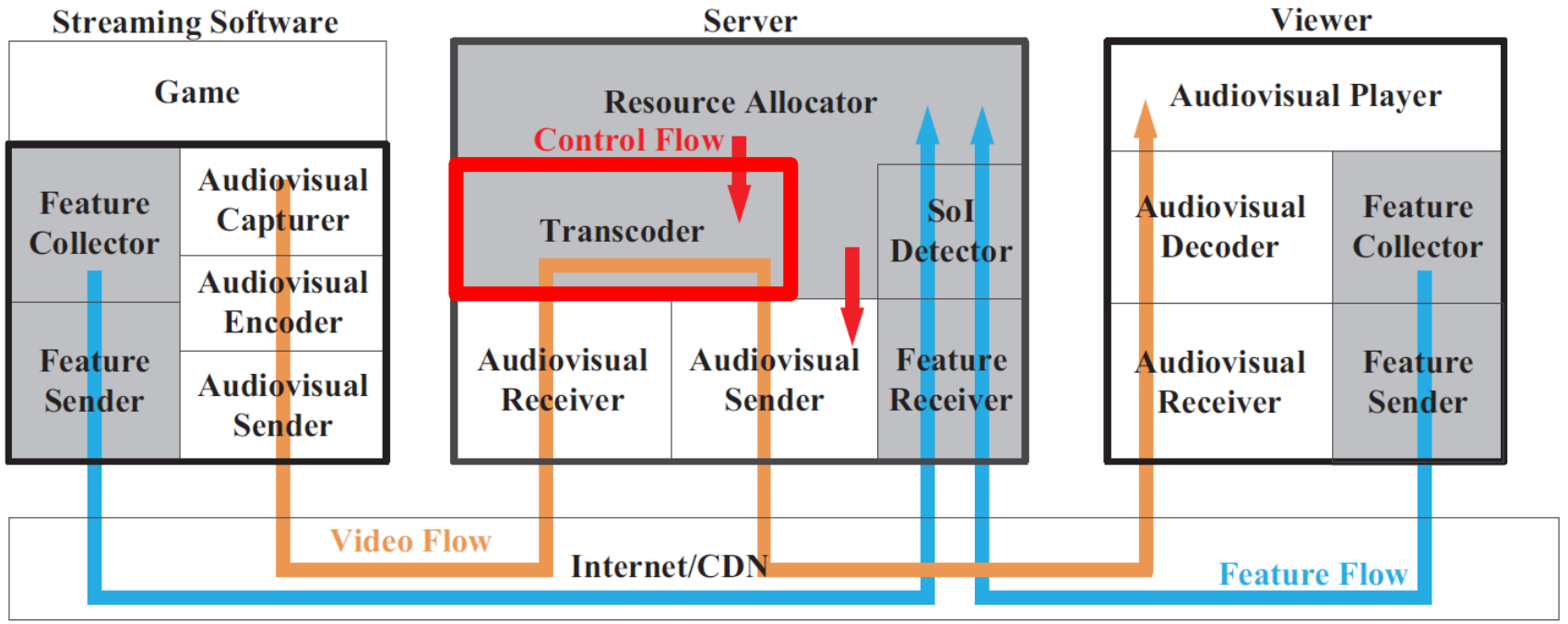
Transcoder

- Transcode the stream



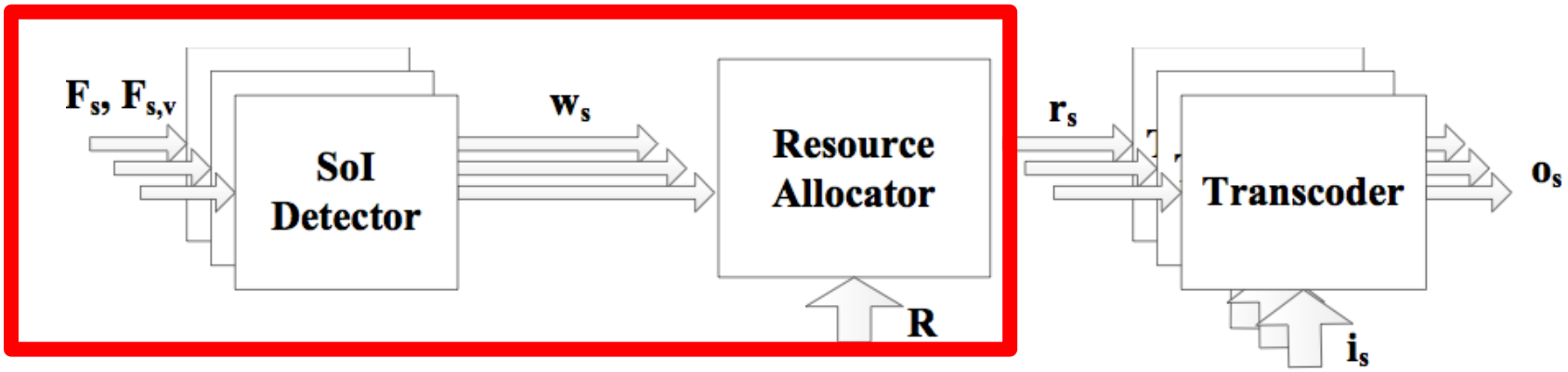
Transcoder

- Transcode the stream



Core Components

- SoI Detector
- Resource Allocator



Outline

- Introduction & Motivation
- System Overview
- SoI Detector
- Resource Allocator
- Evaluation
- Conclusion

Outline

- Introduction & Motivation
- System Overview
- Sol Detector
- Resource Allocator
- Evaluation
- Conclusion

SoI Detection

- How to detect SoI?
  - How to detect if viewers are interested?
- From viewers?
  - Ask viewers to manually tell us? Too annoying
  - Automatically? Require mass deploy, privacy issue

SoI Detection

- How to detect SoI?
  - How to detect if viewers are interested?
- From viewers?
  - Ask viewers to manually tell us? Too annoying
  - Automatically? Require mass deploy, privacy issue

Detect from streamers

How To Detect From Streamer?

- Detect from content (video) [1]?
  - Computationally intensive, not real-time friendly
  - "The segment at 3 hours ago is valuable, to the viewers who've watched it 3 hours ago"
- Detect from other features Our work
  - Features that can be obtained in real-time

[1]. W. Chu and Y. Chou. Event detection and highlight detection of broadcasted game videos. In *Proc. of ACM Workshop on Computational Models of Social Interactions: Human-Computer-Media Communication (HCMC'15)*, Brisbane, Australia, 2015

SoI Detecting Problem

- How to determine SoI weight using features
  - From streamers
  - From viewers (if presented)
- Model the problem as
  - Classification, estimate 0/1
  - Regression, estimate [0, 1]

Features Collected

- CPU usage
- GPU usage
- Context switch
- Streaming bitrate
- Keyboard/mouse input event
- Number of face in webcam
- System sound magnitude
- Microphone sound magnitude
- Name of foreground window

All can be obtained in real-time

But identifying the relations is hard…

Machine Learning

- Use machine learning algorithms
  - Identify the relations between features and SoI
- Which machine learning model to use?[1]
  - Deep Learning?
  - Support Vector Machine?
  - Random Forest?
  - Gradient Boosting Tree?

[1]. Yaser S. Abu-Mostafa, Malik Magdon-Ismail, and Hsuan-Tien Lin. Learning from Data: A Short Course. AMLBook, March 2012

Selected Model

- We choose
  - Random Forest
  - Gradient Boosting Tree
- Both are ensemble models of decision trees
- Efficient in training
  - Unlike SVM and Deep Learning
- Popular
  - Applied in multiple fields

Hyperparameter

- RF-based
  - ***N***, Number of trees
  - ***X***, Maximum number of feature in splitting
  - ***H***, Maximum Depth
- GBT-based
  - ***N***, Number of trees
  - ***E***, Shrinkage (learning rate)
  - ***H***, Maximum Depth
  - ***M***, Subsample

Solution Approach

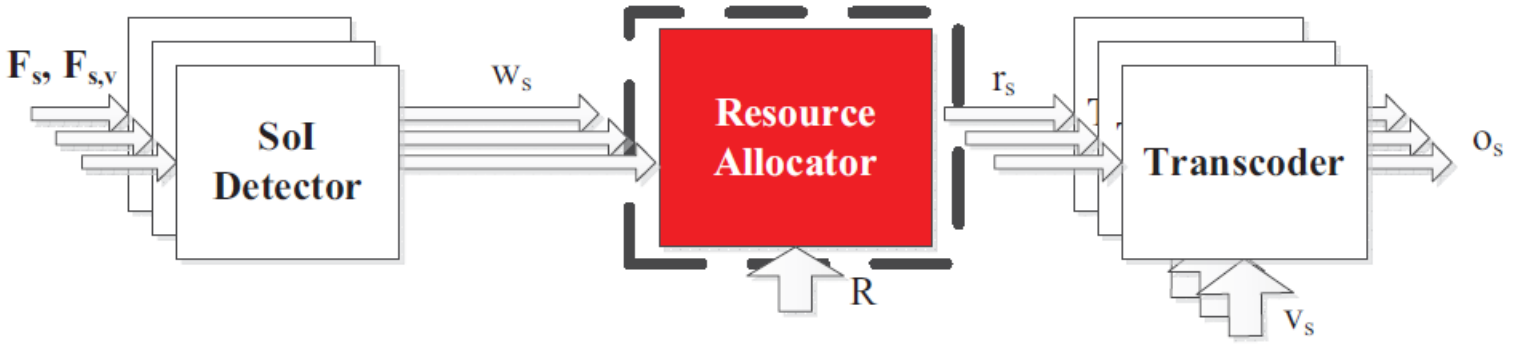
- Implemented in Python
  - Scikit-learn for Random Forest
  - XGBoost for Gradient Boosting Tree
- Using both regressor and classifier variant of them
  - RFR-based & GBTR-based algorithm
  - RFC-based & GBTC-based algorithm
- We conduct supervised learning
  - Details are given in evaluation

Outline

- Introduction & Motivation
- System Overview
- SoI Detector
- Resource Allocator
- Evaluation
- Conclusion

Resource Allocation Problem

- How to allocate resources among streams to maximize viewing quality q_s?
  - Take r_s as the decision variable
- Leverage SoI weight w_s



Connect r_s With q_s

- We adopt a rate-distortion model [1]

$$d_s = D_{0,s} + \frac{\theta_s}{r_s - R_{0,s}}$$

- d_s: distortion in Mean Square Error (MSE)
- $D_{0,s}$, ϑ_s, $R_{0,s}$: model parameters obtained by non-linear regression

[1] X.Zhu, E.Setton, and B.Girod. Congestion-distortion optimized video transmission over ad hoc networks. *Signal Processing: Image Communication*, 20(8):773–783, 2005

Quantifying Viewing Quality

- We use Peak-Signal-to-Noise Ratio (PSNR)
  - $\boldsymbol{q_s = 10 \log_{10} \frac{255^2}{d_s}}$
- The proposed algorithm can utilize other quality metrics with monotonically increasing property

Problem Formulation

$$\text{maximize} \sum_{s=1}^{S} q_s w_s = \sum_{s=1}^{S} 10 \log_{10} \frac{255^2}{D_{0,s} + \frac{\theta_s}{r_s - R_{0,s}}} w_s$$

$$\text{s.t.} \sum_{s=1}^{S} r_s w_s \leq R;$$

$$r_s \in R^+, \forall s = 1, 2, ..., S.$$

Problem Formulation

$$\text{maximize} \sum_{s=1}^{S} q_s w_s = \sum_{s=1}^{S} 10 \log_{10} \frac{255^2}{D_{0,s} + \frac{\theta_s}{r_s - R_{0,s}}} w_s$$

$$\text{s.t.} \sum_{s=1}^{S} r_s w_s \leq R;$$

$$r_s \in R^+, \forall s = 1, 2, ..., S.$$

Objective: Maximize the viewing quality of all the viewers in SoI

Problem Formulation

$$\text{maximize} \sum_{s=1}^{S} q_s w_s = \sum_{s=1}^{S} 10 \log_{10} \frac{255^2}{D_{0,s} + \frac{\theta_s}{r_s - R_{0,s}}} w_s$$

$$\text{s.t.} \sum_{s=1}^{S} r_s w_s \leq R;$$

$$r_s \in R^+, \forall s = 1, 2, ..., S.$$

Under the constraint that the total bandwidth usage is under R

Derive Closed Form Formula

- Leverage Lagrangian Multiplier method
  - Added $\boldsymbol{\lambda}$
- We get Lagrangian function ***L***

$$\sum_{s=1}^{S}(10\log_{10}\frac{255^2}{D_{0,s}+\frac{\theta_s}{r_s-R_{0,s}}}w_s)+\lambda((\sum_{s=1}^{S}r_sw_s)-R)$$

- Take partial derivative of ***L***

$$\frac{\partial\mathscr{L}}{\partial\lambda}=(\sum_{s=1}^{S}r_sw_s)-R;$$

In order to get the extreme value of L

$$\frac{\partial\mathscr{L}}{\partial r_s}=\lambda+\frac{10\theta s}{\log 10(r_s-R_{0,s})^2(D_{0,s}+\frac{\theta_s}{r_s-R_{0,s}})}$$

S of them

Solving the Formula

- Let $\frac{\partial \mathscr{L}}{\partial r_s} = 0$
- After derivation

$$r_s = R_{0,s} + \frac{-(\lambda \log 10\theta_s) - \sqrt{(\lambda \log 10\theta_s)^2 - 40\lambda \log 10\theta_s^2}}{2(\lambda \log 10 D_{0,s})}$$

- Combine the results above with

$$\frac{\partial \mathscr{L}}{\partial \lambda} = (\sum_{s=1}^{S} r_s w_s) - R = 0$$

The result of the other partial derivative

- We can obtain the optimal $\boldsymbol{\lambda}$
  - Then derive the optimal bitrates for streaming to each viewer ***v*** of streamer ***s***
- **Slow to solve** **There are S + 1 formulas**

A Real-Time Approximation Algorithm

- Solve the problem efficiently
  - With a controllable error of $\boldsymbol{\varepsilon}$
- Called **SoI-based R-D Optimized Algorithm (SoI RDO)**

SoI RDO Algorithm

Algorithm 1 Efficient SoI RDO Algorithm

1: $U \leftarrow 0$, $L \leftarrow -1$ // Upper/lower bounds
2: $\epsilon \leftarrow 10^{-7}$ // Default error
3: **while** $L + \epsilon < U$ **do**
4: $\quad \lambda = (U + L)/2$
5: $\quad$ **for** $\forall s \in S$ **do**
6: $\quad\quad$ Derive all the r_s using λ and Eq. (5)
7: $\quad$ **if** $\exists r_s$, such that $q_s \in \mathbb{C}$ or $r_s < R_L$ or $r_s > R_U$ **then**
8: $\quad\quad$ Adjust U or L accordingly
9: $\quad$ **else if** $\sum_{s \in S} r_s w_s > R$ **then**
10: $\quad\quad U = \lambda$
11: $\quad$ **else**
12: $\quad\quad obj = \sum_{s \in S} q_s w_s$
13: $\quad\quad L = \lambda$
14: if *obj* is undefined, return no answer, o.w. return λ and r_s.

Binary search on proper λ value

Derive all r_s & q_s using λ this round

Check if all r_s and q_s are valid

Check if use too much bandwidth

Calculate objective function

Lemma 1

Lemma 1 (Approximation Gap)**.** *SoI RDO algorithm always finds* $\boldsymbol{\lambda}$ *within a gap of* $\boldsymbol{\varepsilon}$ *to the optimal* $\boldsymbol{\lambda}^*$ *under the given bandwidth constraint* $\boldsymbol{R}$.

- Reason why the quality metrics need to have monotonically increasing property.

Lemma 2

Lemma 2 (Complexity). *Algorithm 1 runs in* $O(S \log \epsilon^{-1})$

- $\log \boldsymbol{\varepsilon^{-1}}$ round of search in worst case
- Each round need to do
  - Derive $\boldsymbol{r_s}$, $\boldsymbol{q_s}$ for every streamer $\boldsymbol{s}$ $O(\boldsymbol{S})$
  - Check if $\boldsymbol{r_s}$ is valid for every streamer $\boldsymbol{s}$ $O(\boldsymbol{S})$
  - Calculate the total consumed bandwidth $O(\boldsymbol{S})$
  - Calculate the objective value $O(\boldsymbol{S})$
- $O(\log \boldsymbol{\varepsilon^{-1}})$ x $(O(\boldsymbol{S}))$ = $O(\boldsymbol{S} \log \boldsymbol{\varepsilon^{-1}})$

Outline

- Introduction & Motivation
- System Overview
- SoI Detector
- Resource Allocator
- Evaluation
- Conclusion

Outline

- Introduction & Motivation
- System Overview
- SoI Detector
- Resource Allocator
- Evaluation
- Conclusion

Open Source Testbed

- Streaming software: OBS [1]
- Streaming server: NGINX [2] with RTMP plug-in
- Viewer: SMPlayer [3]

[1] https://obsproject.com/
[2] http://nginx.org
[3] http://smplayer.sourceforge.net/

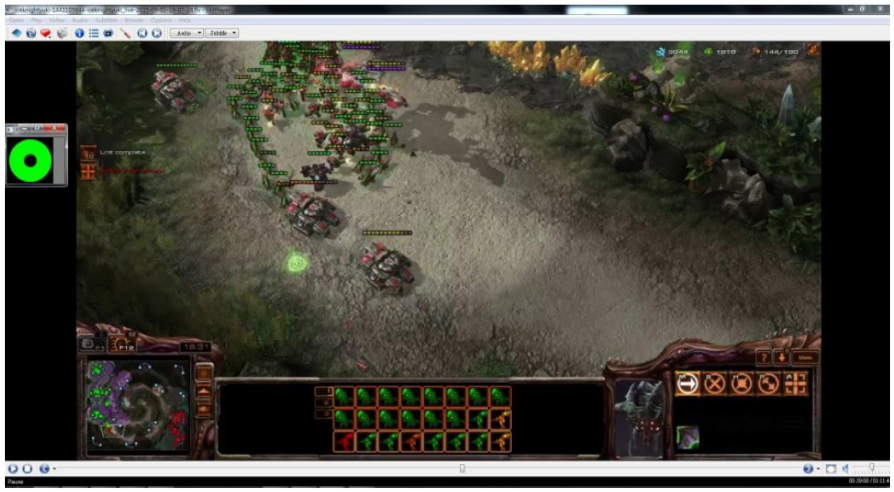
SOI DETECTOR EXPERIMENT SETUP

Collecting the Dataset

- Collected at a League of Legend tournament
- 50 participants, 10 matches
  - 100 video & trace
  - 207004 seconds in total
- 81 valid video & trace
  - 162842 samples

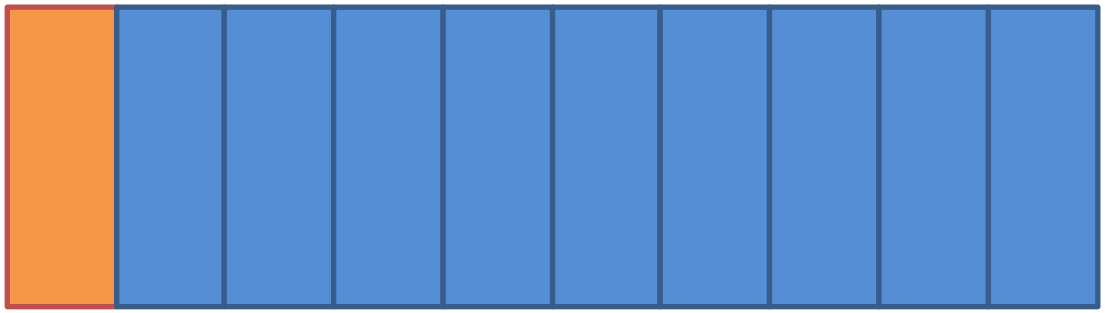
Marking SoI Ground Truth

- We recruited 17 viewers to mark SoI manually
  - 10 videos
  - With a modified video player
- Collected 74 logs
  - Cover 27010 samples



Partition the Dataset

- 90% as training dataset
  - Used in hyperparameter tuning
  - 18677 samples
- The rest 10% as evaluation dataset
  - 2076 samples



Performance Metrics

- Classification
  - F-measure = $2 \cdot \frac{precision \cdot recall}{precision + recall} \in [0, 1]$
  - Training Time
- Regression
  - R-squared score $\in [1, -\infty]$
  - Training Time

Hyperparameter Tuning

- 10-fold cross validation
- Grid search
- RF-based
  - ***N***: [30 ,60, 120, 240, 480]
  - ***X***: [5, 10, 20, 40]
  - ***H***: [10, 20, 40, 80, 160]
- GBT-based
  - ***N***: [5, 10, 20, 40, 80]
  - ***E***: [0.01, 0.05, 0.1, 0.2, 0.4]
  - ***H***: [5, 10, 20, 40, 80]
  - ***M***: [0.5, 0.6, 0.7, 0.8, 0.9]

The Optimal Hyperparameter

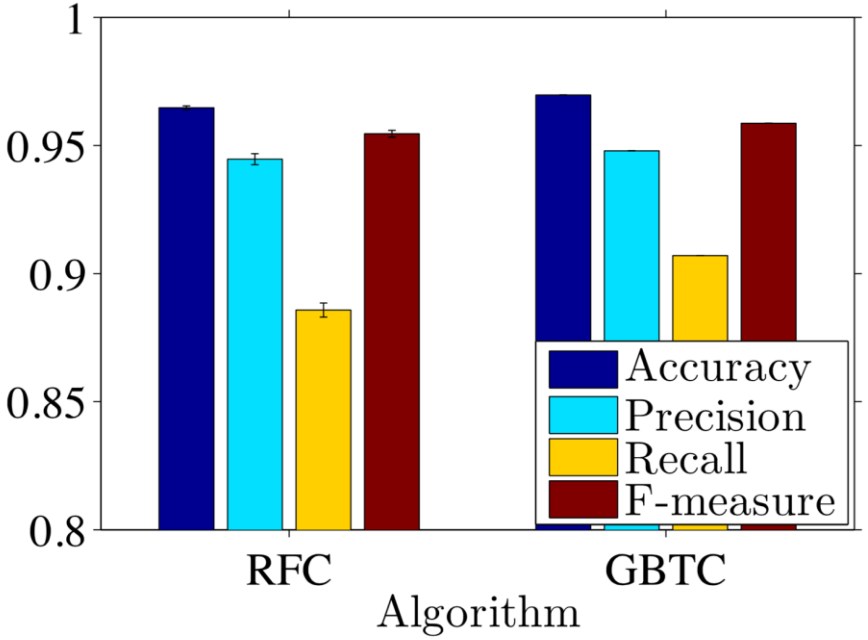
- RFR-based algorithm
  - ***N***: 240, ***X***: 5, ***H***: 40
- GBTR-based algorithm
  - ***N***:80, ***E***: 0.2, ***H***: 10, ***M***: 0.9
- RFC-based algorithm
  - ***N***: 480, ***X***: 5, ***H***: 80
- GBTC-based algorithm
  - ***N***: 80, ***E***: 0.2, ***H***:80, ***M***: 0.7

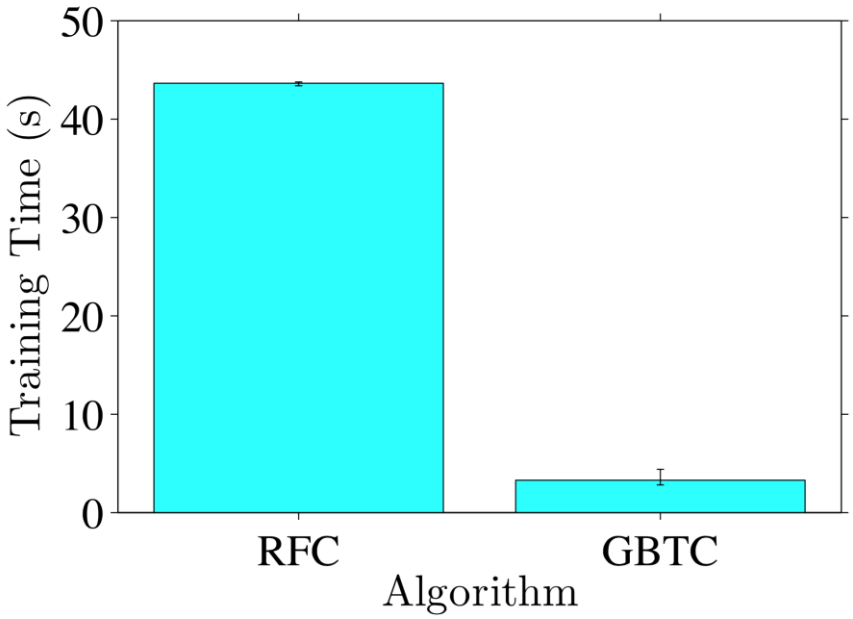
Use them in evaluation

SOI DETECTOR EXPERIMENT RESULT

RFC-based & GBTC-based Algorithm

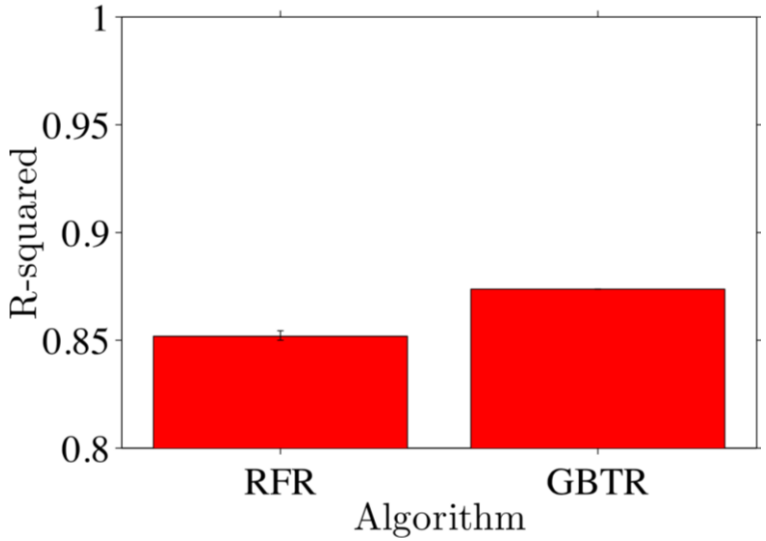
- Over 0.95 F-measure score
- Both terminated under 42 seconds
  - Less than 5 seconds for GBTC

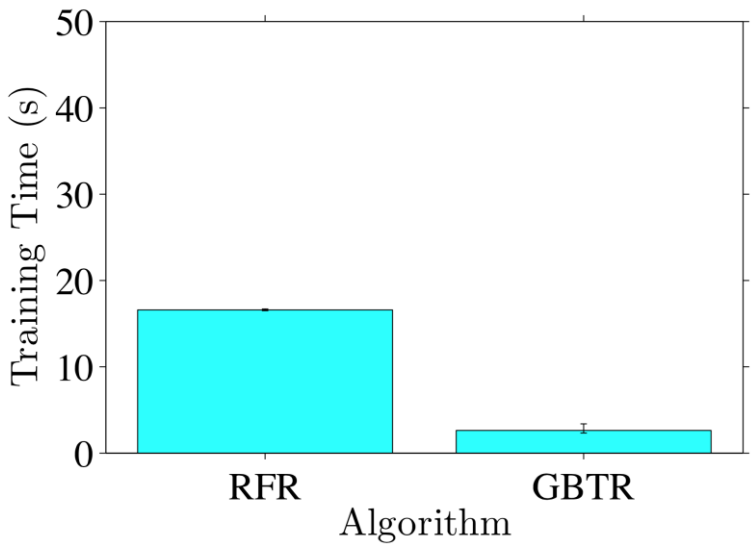




RFR-based & GBTR-based algorithm

- Over 0.85 in R-squared
- Both algorithm terminated under 20 seconds





Observation

- All algorithms provide good result
  - GBT-based algorithm generally gives better result
- GBT-based algorithm have shorter training time
  - Not what we expect
  - XGBoost leverage up to 62 cores in our experiment
  - Scikit-learn's RF implementation only use single core

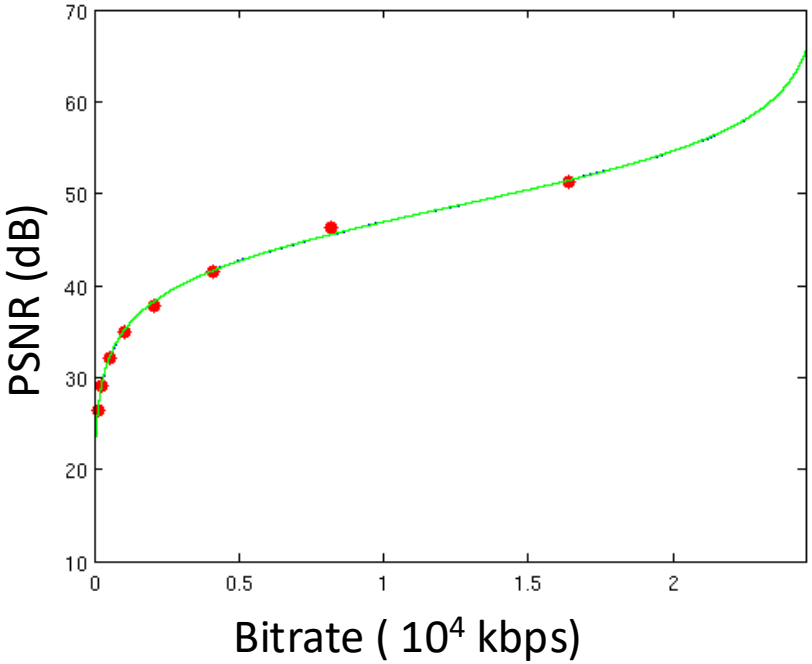
RESOURCE ALLOCATOR EXPERIMENT SETUP

Allocator Simulation Setup

- We captured 8 game streamed on our testbed
  - Age of Empires II
  - Spellweaver
  - Hearthstone: Heros of Warcraft
  - Minecraft
  - Starcraft II
- Resolution between 720p and 1080p

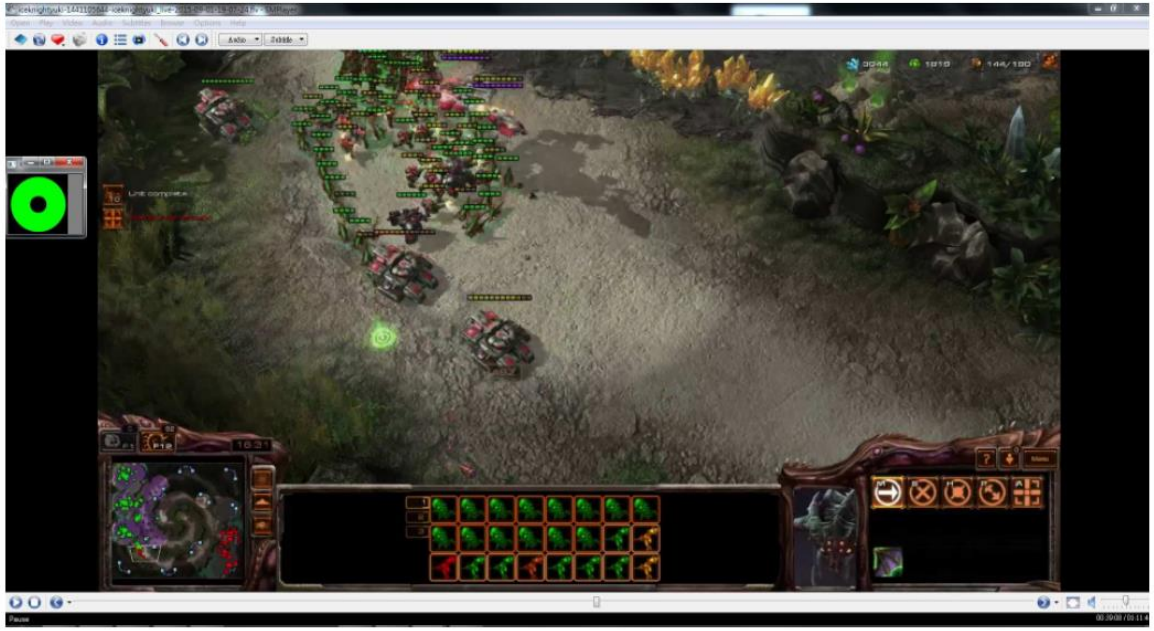
Allocator Simulation Setup (cont.)

- Transcode them into different bitrates
  - Obtain the PSNR value
  - Perform non-linear regression to get the R-D model parameters



Allocator Simulation Setup (cont.)

- We recruited several viewers to watch the recorded live stream session
  - Mark SoI
  - 14 logs in total



Simulator & Baseline

- Simulator is implemented in Python
- SoI RDO algorithm is implemented in C
- Two baseline solution
  - **Equal Share (ES)**: equally divides bandwidth between all viewers **State-Of-The Art live game streaming solution**
  - **R-D Optimized (RDO)**: optimized in R-D sense, but does not take SoI into consideration

Quantify the benefit of SoI driven allocation

- Also an implementation of optimal algorithm
  - In Matlab, as a benchmark of SoI RDO

Performance Metrics

- Expected quality
  - The objective value reported by algorithm
- Actual quality
  - The objective value reported by simulator
- Running time
  - Runtime of solving the resource allocation problem
- Expected consumed bandwidth
  - Reported by algorithm
- Actual consumed bandwidth
  - Reported by simulator
- Network overhead
  - Network traffic generated by feature senders

Experiment Parameters

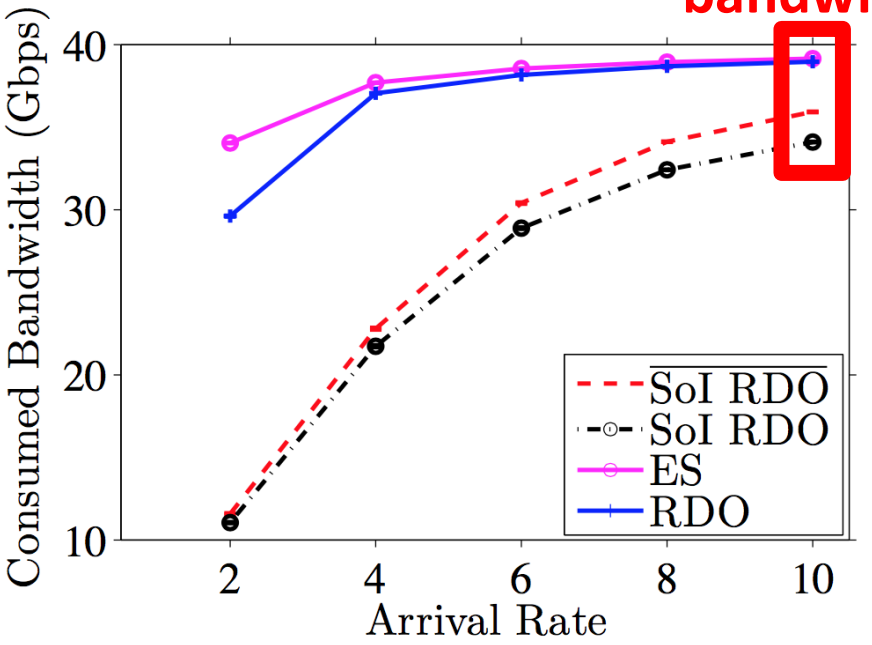
- 6 hour long simulations
- Poisson arrival rate of viewers in each stream
  - {2, 4, 6, 8, 10} per minutes
- Number of stream
  - {16, 32, 64, 128, 256}
- Total outbound bandwidth
  - {10, 20, 40, 80, 160} Gbps
- Interval of invoking resource allocator
  - {1, 2, 5, 10, 60} seconds

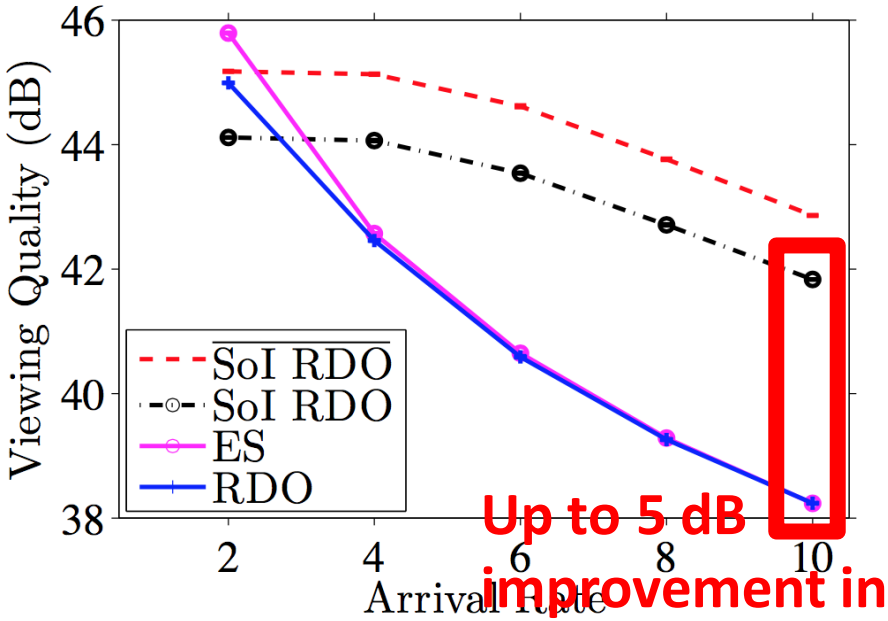
RESOURCE ALLOCATOR RESULTS

Our Algorithm Outperforms RDO and ES

- Different viewer arrival rate

With less consumed bandwidth



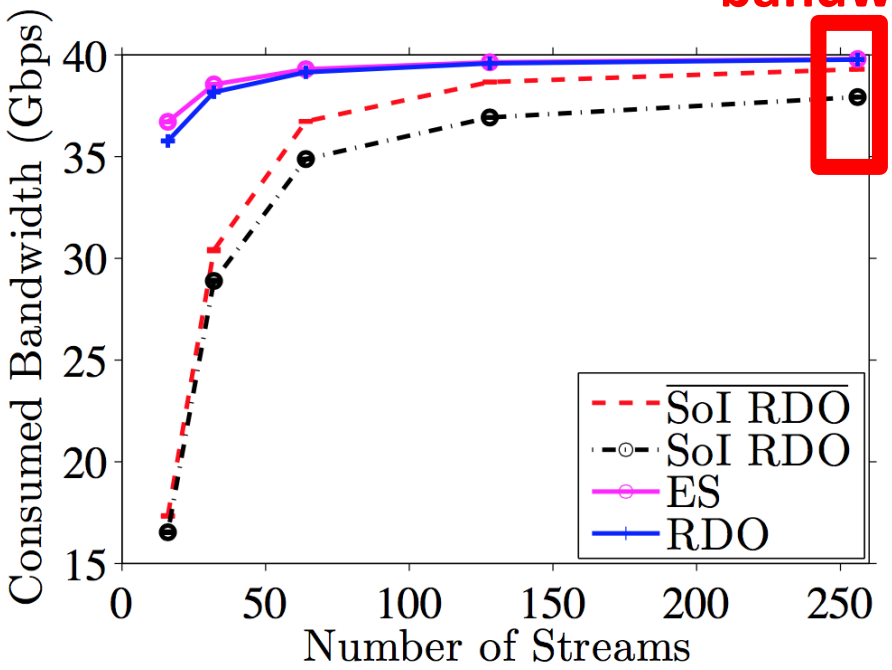


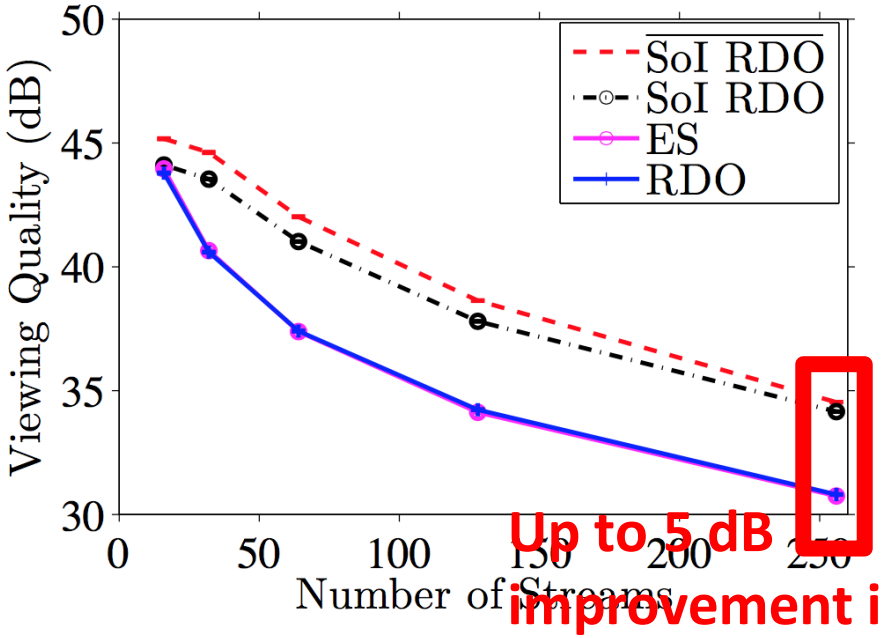
Up to 5 dB improvement in viewing quality

Our Algorithm Outperforms RDO and ES

- Different number of stream

With less consumed bandwidth

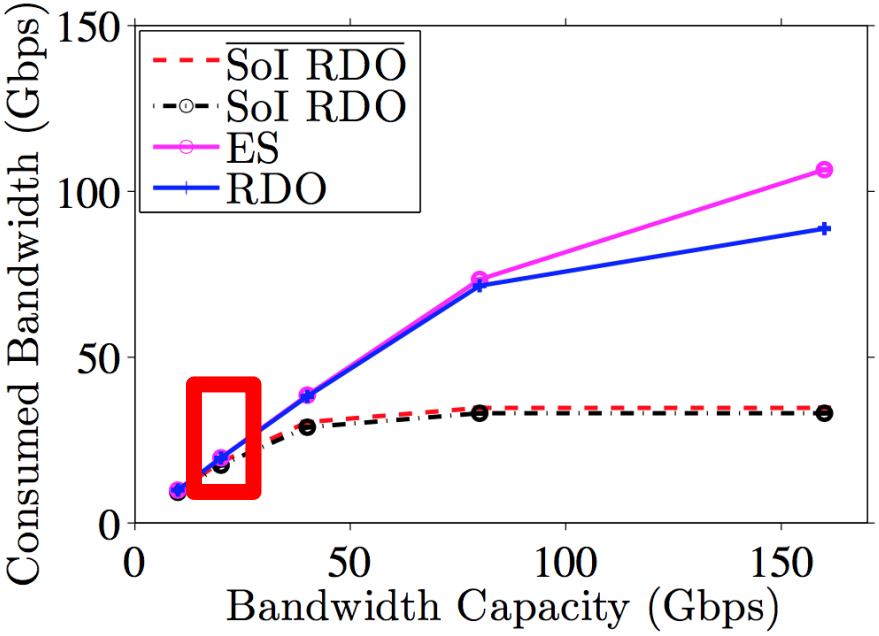


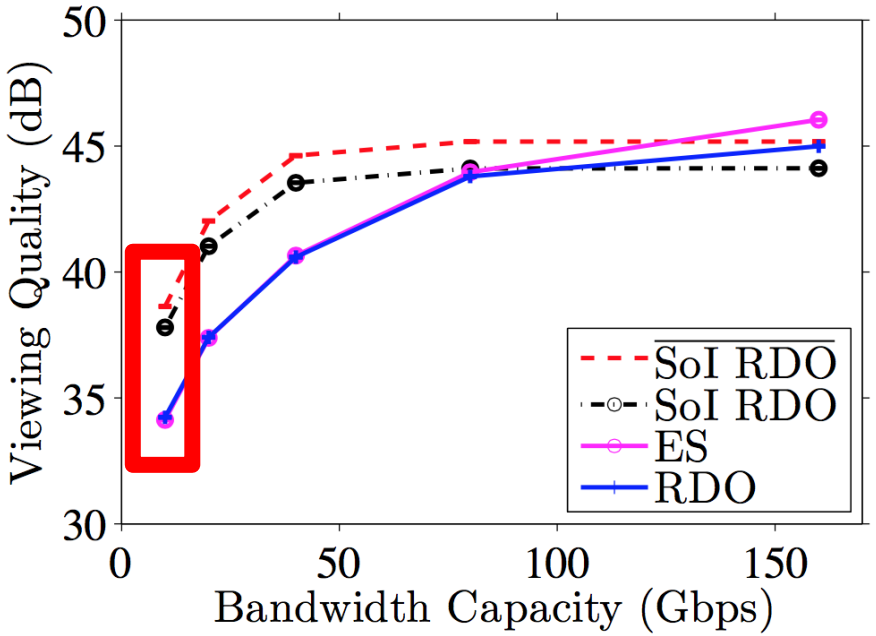


Up to 5 dB improvement in viewing quality

Our Algorithm Outperforms RDO and ES

- Different bandwidth capacity

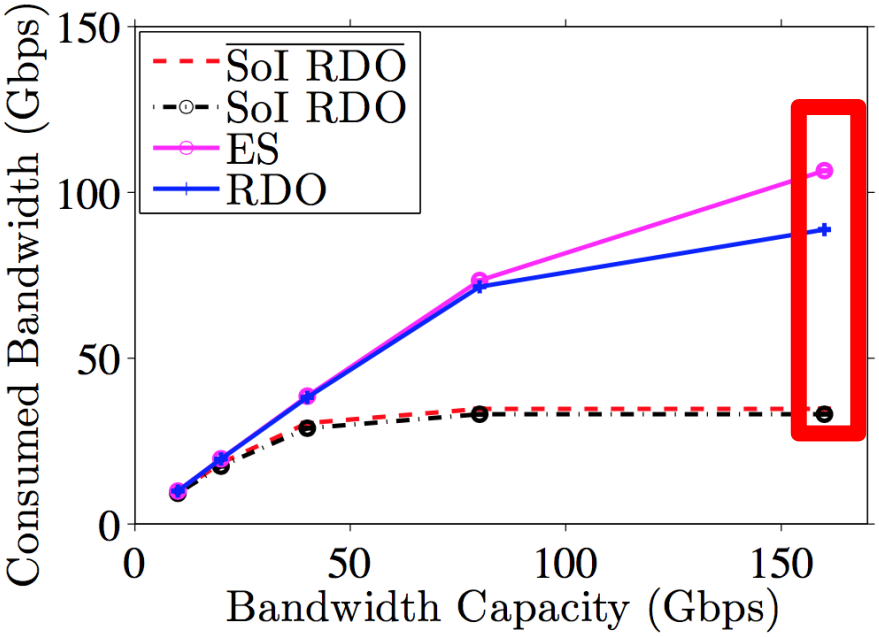


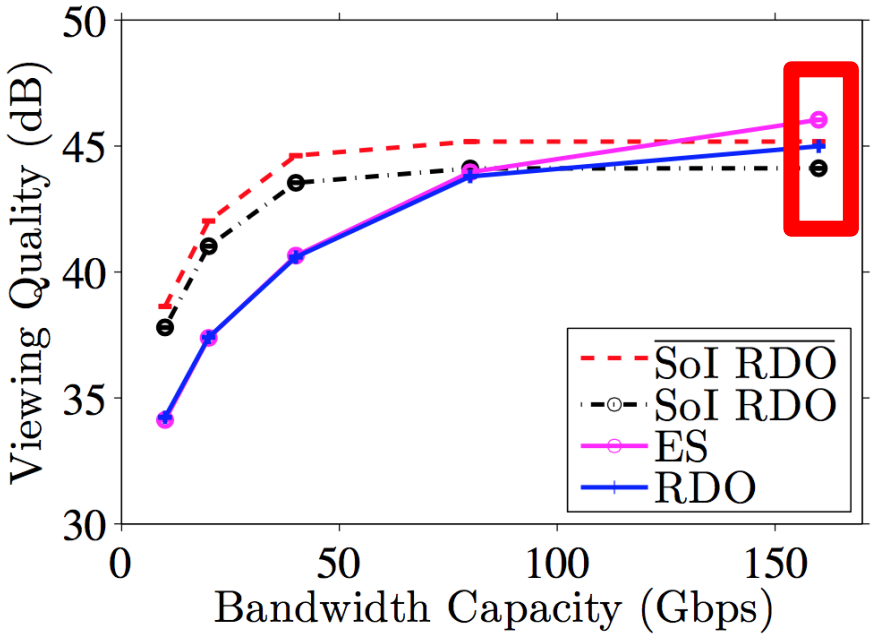


Better quality under the same bandwidth usage

Our Algorithm Outperforms RDO and ES

- Different bandwidth capacity

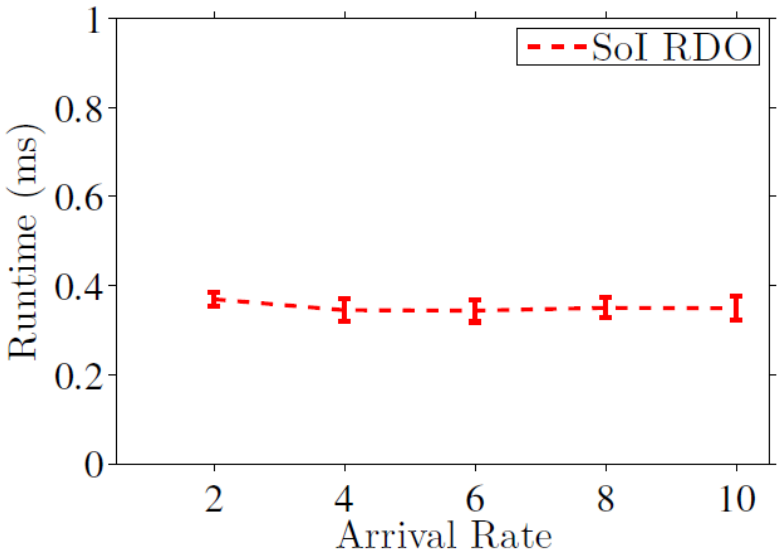


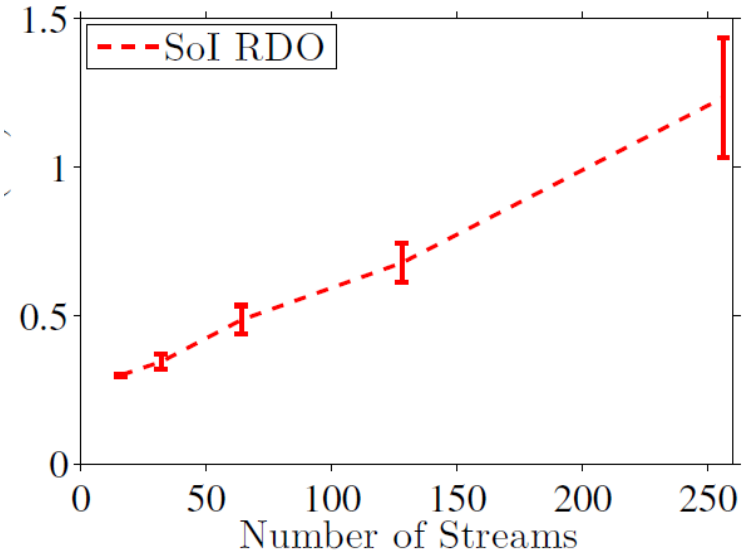


Slightly outperformed by ES and RDO, but that was with significantly larger bandwidth consumption

Our Algorithm Scales Well

- Always below 1.5 ms
  - 100+ thousand viewers
- Matlab solution may take over 6 minutes
  - With 58% chance of feasible solution





Outline

- Introduction & Motivation
- System Overview
- SoI Detector
- Resource Allocator
- Evaluation
- Conclusion

Outline

- Introduction & Motivation
- System Overview
- SoI Detector
- Resource Allocator
- Evaluation
- Conclusion

Conclusion

- We proposed the concept of **Segment of Interest (SoI)**
  - New possibility for optimization
- We build an open source testbed
  - Collect real world traces
- We develop accurate SoI detector for MOBA game
  - Up to 0.95 F-measure and 0.87 R-squared score
- Efficient SoI driven resource allocation algorithm
  - Outperforms the current solution up to 5 dB

A SHORT LIVE DEMO OF OUR SYSTEM

Future Work

- Consider device capability in SoI detector
  - Only homogeneous device in evaluation
- Leverage features collected from viewers
  - More fine-grained optimizations
- Resource allocator with finer transcoder control
  - Not only bandwidth, e.g., with FPS, resolution options
- Integrate with a real transcoder
- Provide API for deep integration with game engines

Research Highlight

- **T. Fan-Chiang**, H. Hong and C. Hsu, "Segment-of-Interest Driven Live Game Streaming: Saving Bandwidth Without Degrading Experience", in **Proc. of ACM Annual Workshop on Network and Systems Support for Games (NetGames '15)**
- Y. Huang, M. Lee, **T. Fan-Chiang** and C. Hsu, "Minimizing Flow Initialization Latency in Software Define Networks", in **Prof. of Asia-Pacific Network Operations and Management Symposium (APNOMS'15)**
- H. Hong, **T. Fan-Chiang**, C. Lee, K. Chen, C. Huang and C. Hsu, "GPU Consolidation for Cloud Games: Are we There Yet?", in **Proc. of ACM Annual Workshop on Network and Systems Support for Games (NetGames '14)**
- C. Mao, **T. Fan-Chiang**, M. Lee and C. Hsu, "Taming Flow Initialization Delay in Multi-Site Software Defined Enterprise Networks", in preparation for **IEEE/ACM Transaction on Networking (TNET)**
- TMM submission (based from this thesis)

THANK YOU

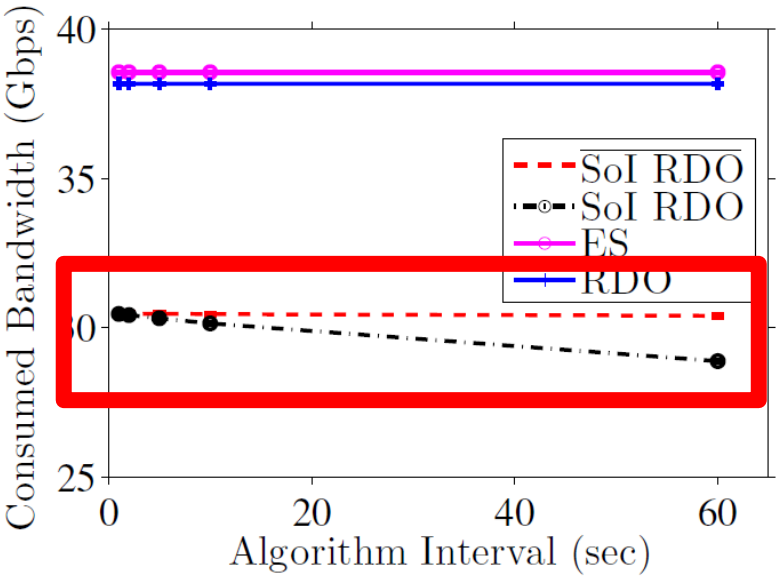
Q & A

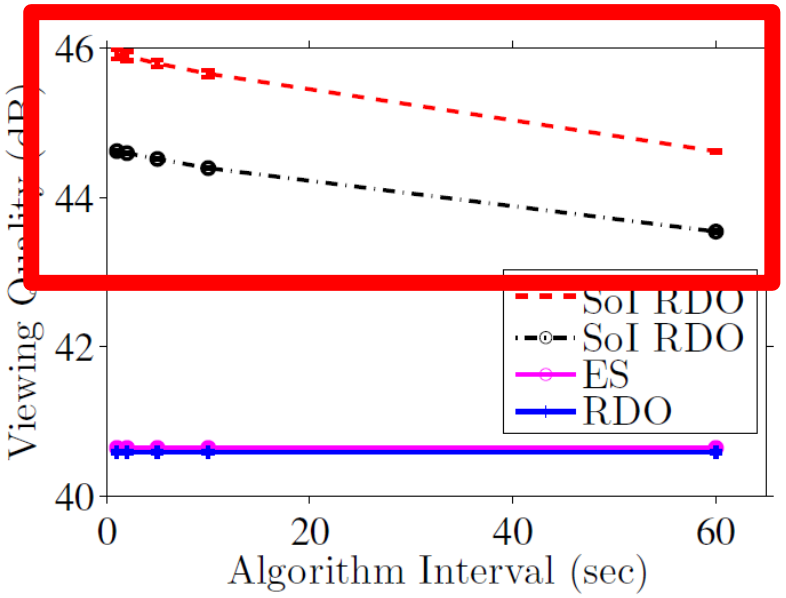
tyfanchiang92@gmail.com

BACKUP SLIDES

Implication Of Calling Interval

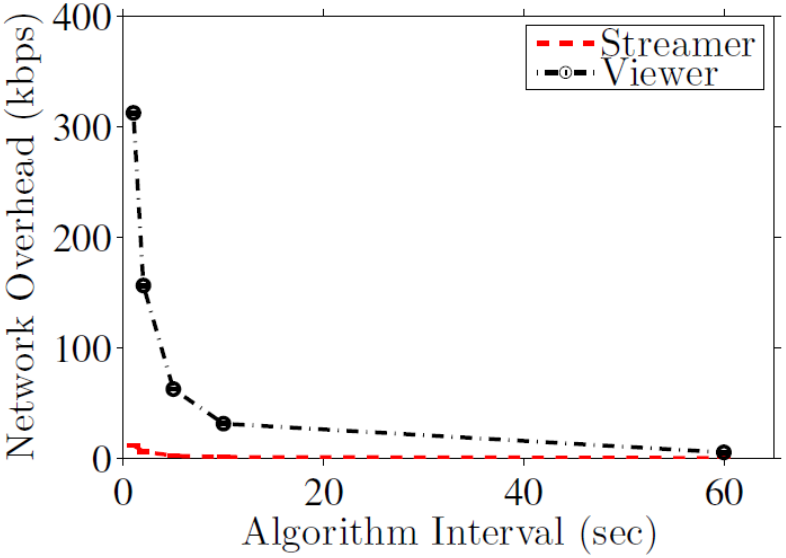
- Slightly better quality and more bandwidth consumption

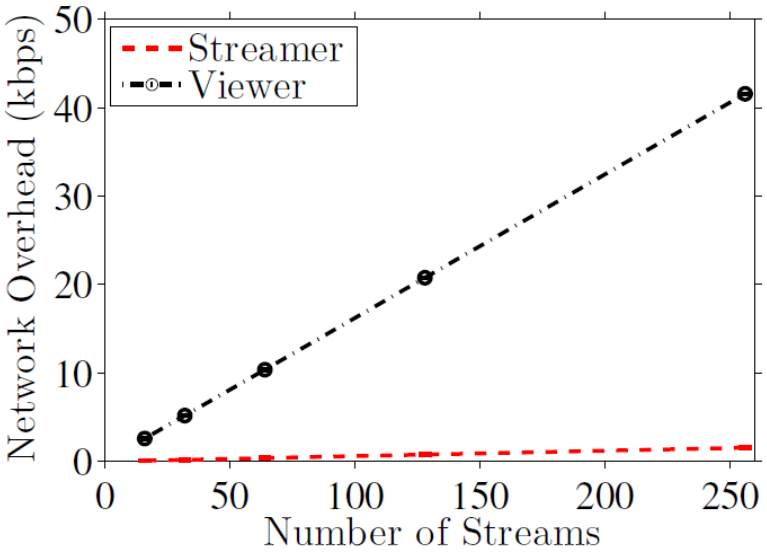




Negligible Network Overhead

- Just around 300 kbps with 12000+ viewers



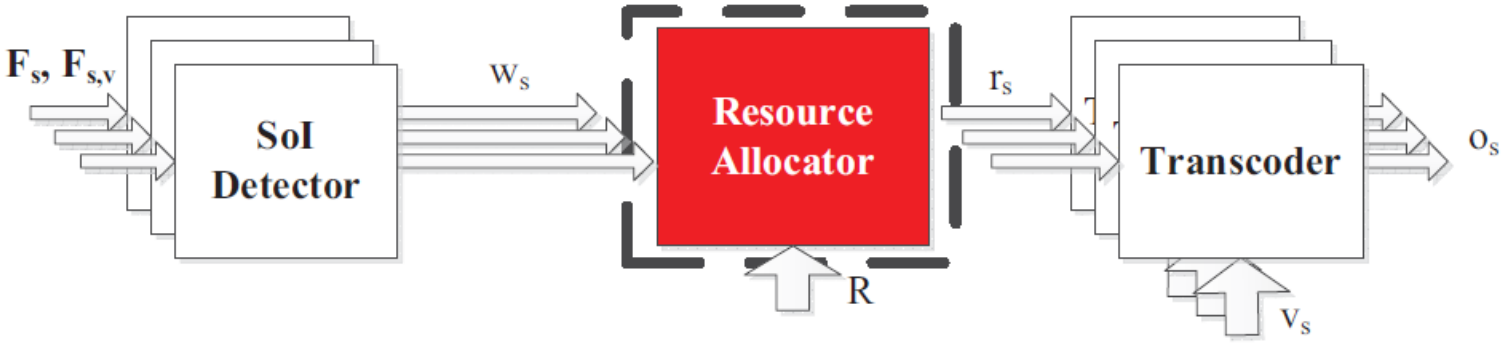


SoI-Driven Streaming Platform

- Develop SoI-driven streaming platform
- Answer three question
  - How to detect SoI efficiently?
  - How to allocation resources among channels?
  - How to perform transcode on the streaming videos?

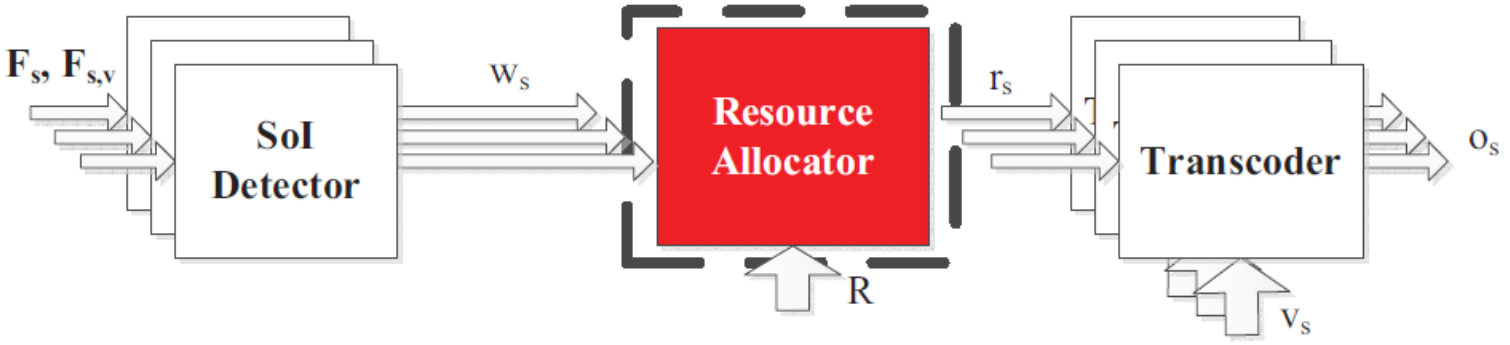
Notations

- The system collects
  - A set of features F_s from streamer s
  - A set of features $F_{s,v}$ from viewer v of streamer s
- SoI detector take F_s and $F_{s,v}$ as input
  - Output SoI weight w_s for current segment

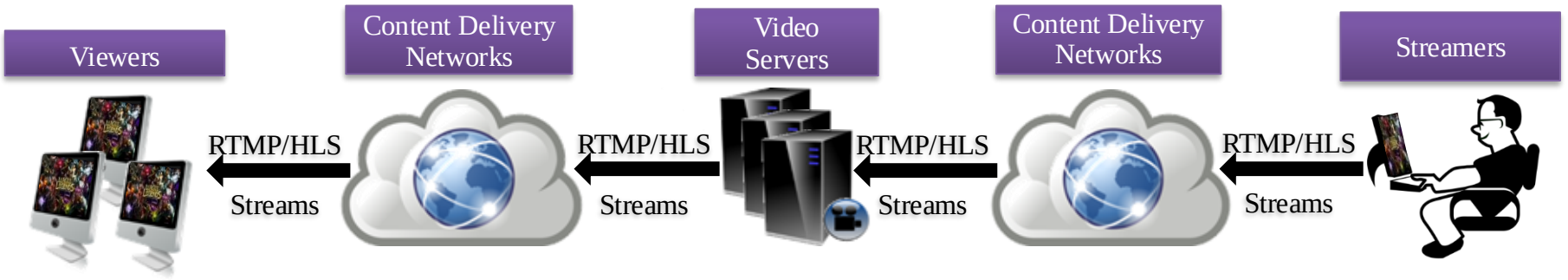


Notations

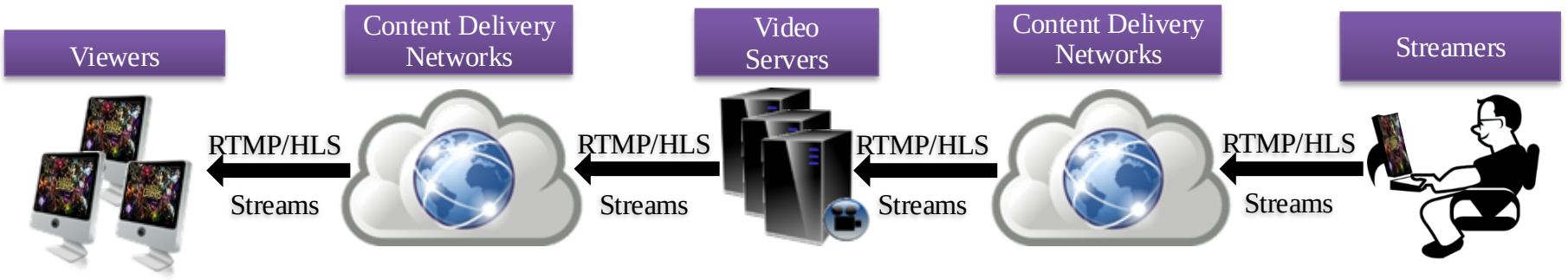
- Resource allocator take all the w_s and bandwidth capacity R as input
  - Output r_s as the bitrate for video segment v_s from s
- Transcoder use r_s as parameter to transcode v_s
  - Output o_s stream to viewers



WHY DO LAG HAPPENS?

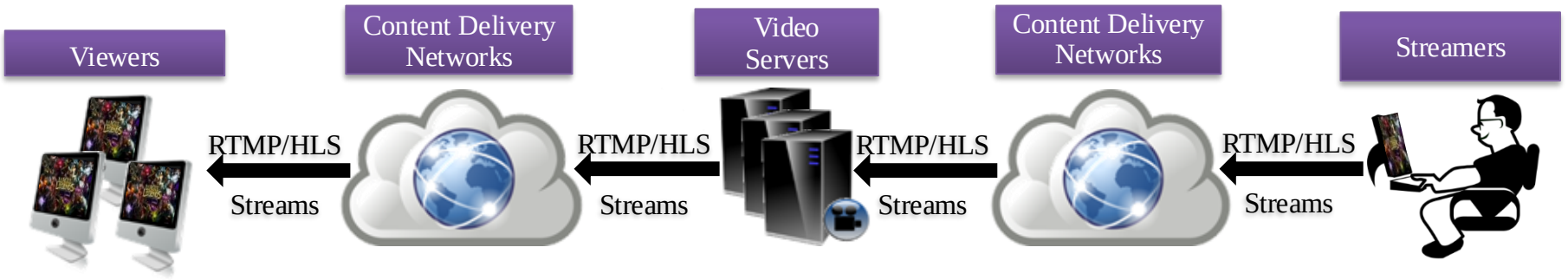


Reason 1: Server Too Busy



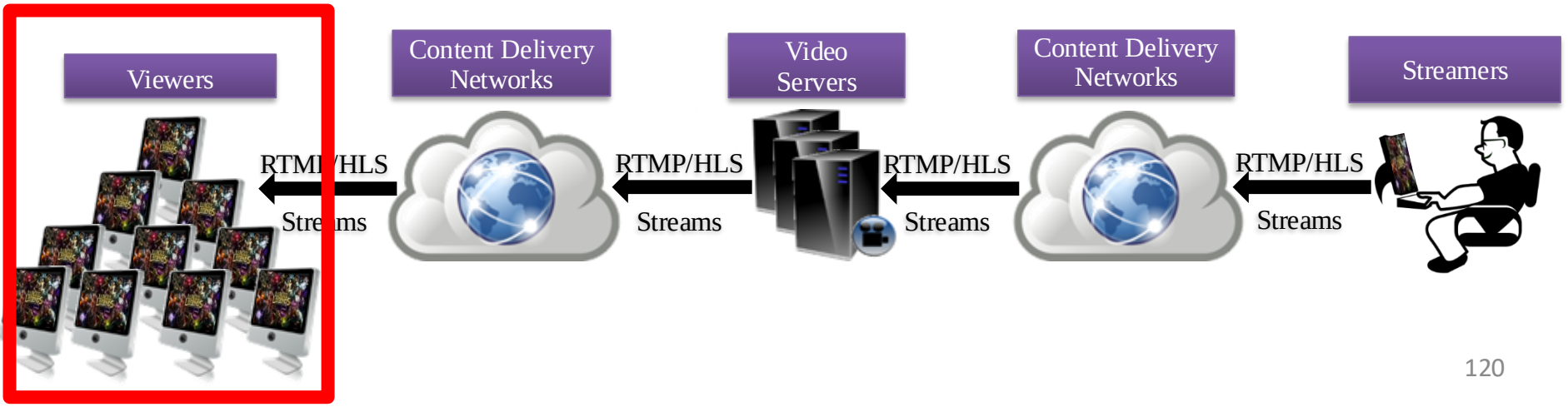
Reason 1: Server Too Busy

- Too many viewers



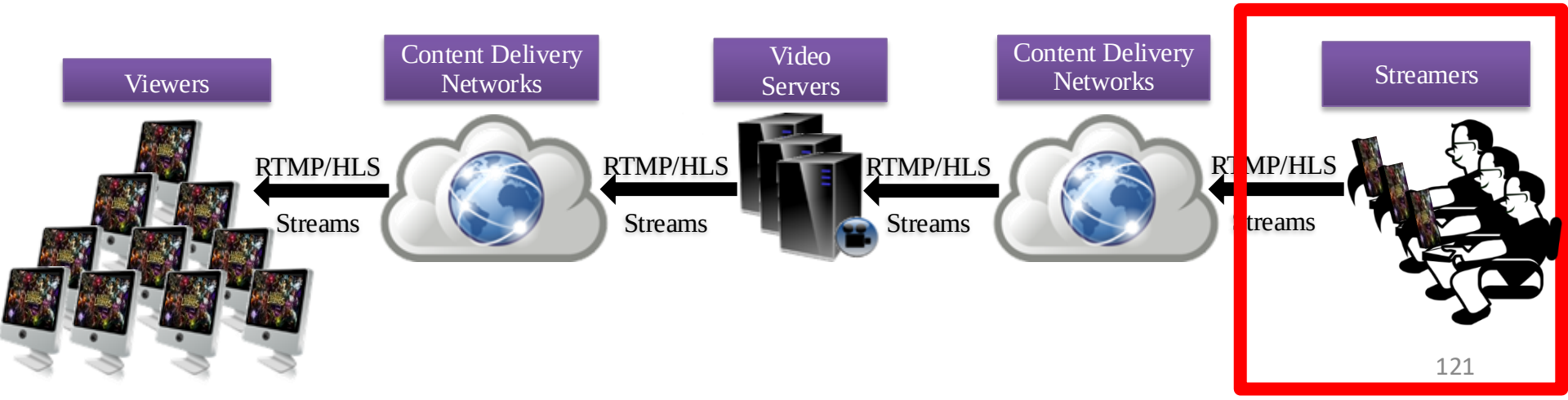
Reason 1: Server Too Busy

- Too many viewers



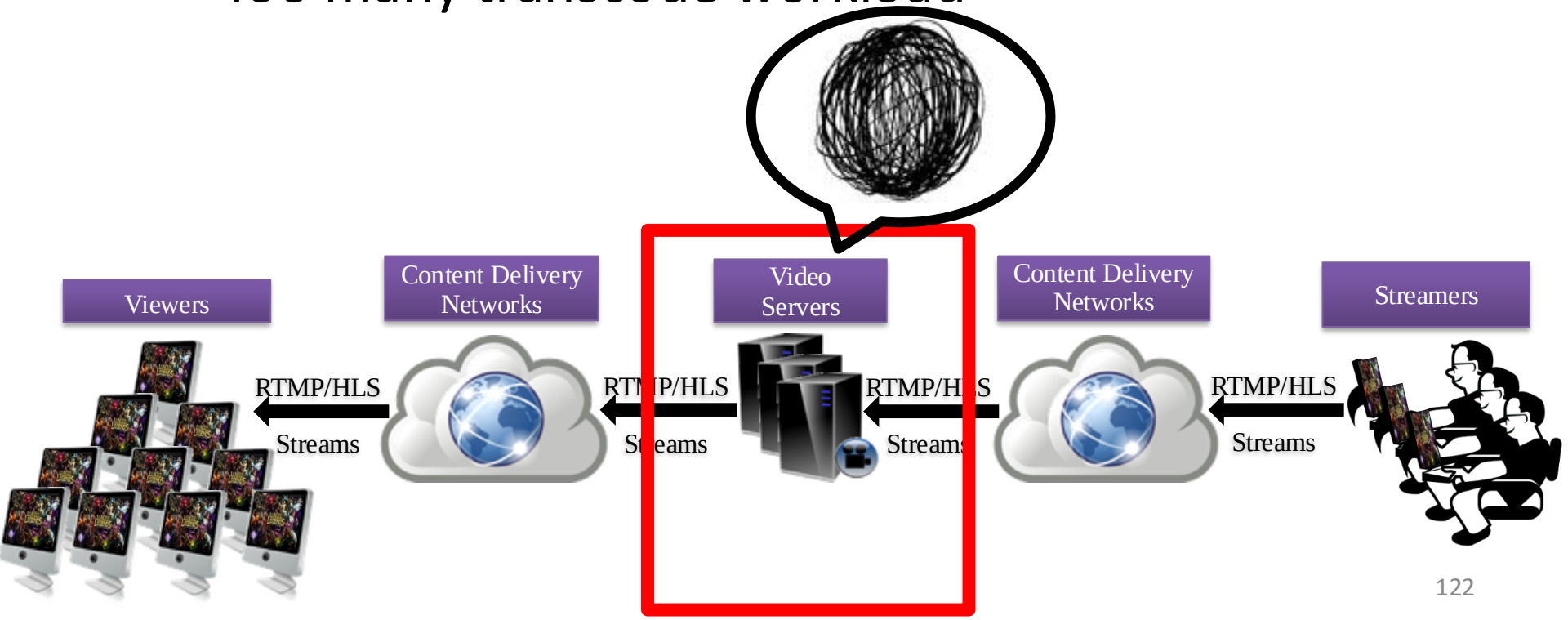
Reason 1: Server Too Busy

- Too many viewers
- Too many streamers
  - Too many transcode workload



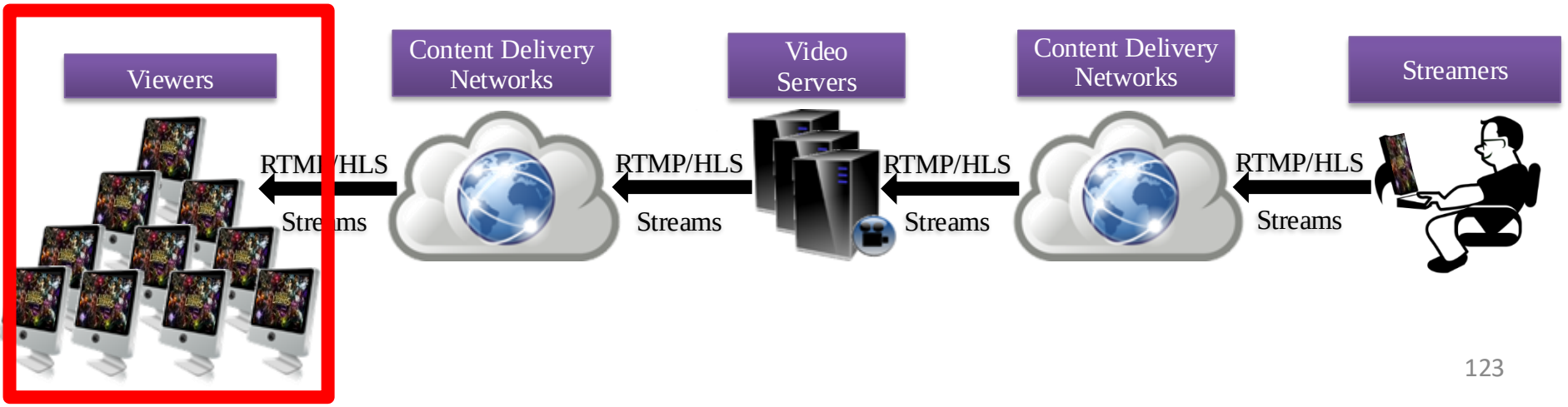
Reason 1: Server Too Busy

- Too many viewers
- Too many streamers
  - Too many transcode workload



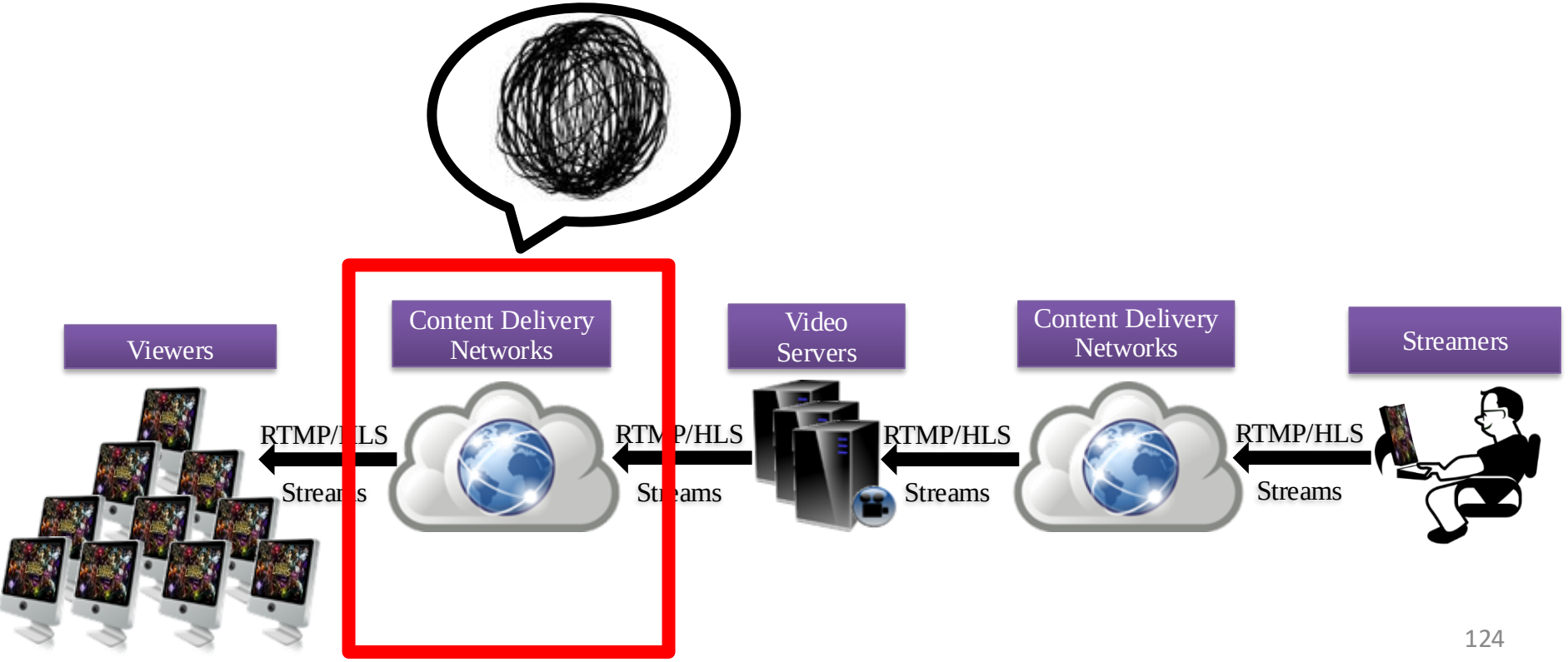
Reason 2: Outbound Bandwidth

- Too many viewers



Reason 2: Outbound Bandwidth

- Too many viewers



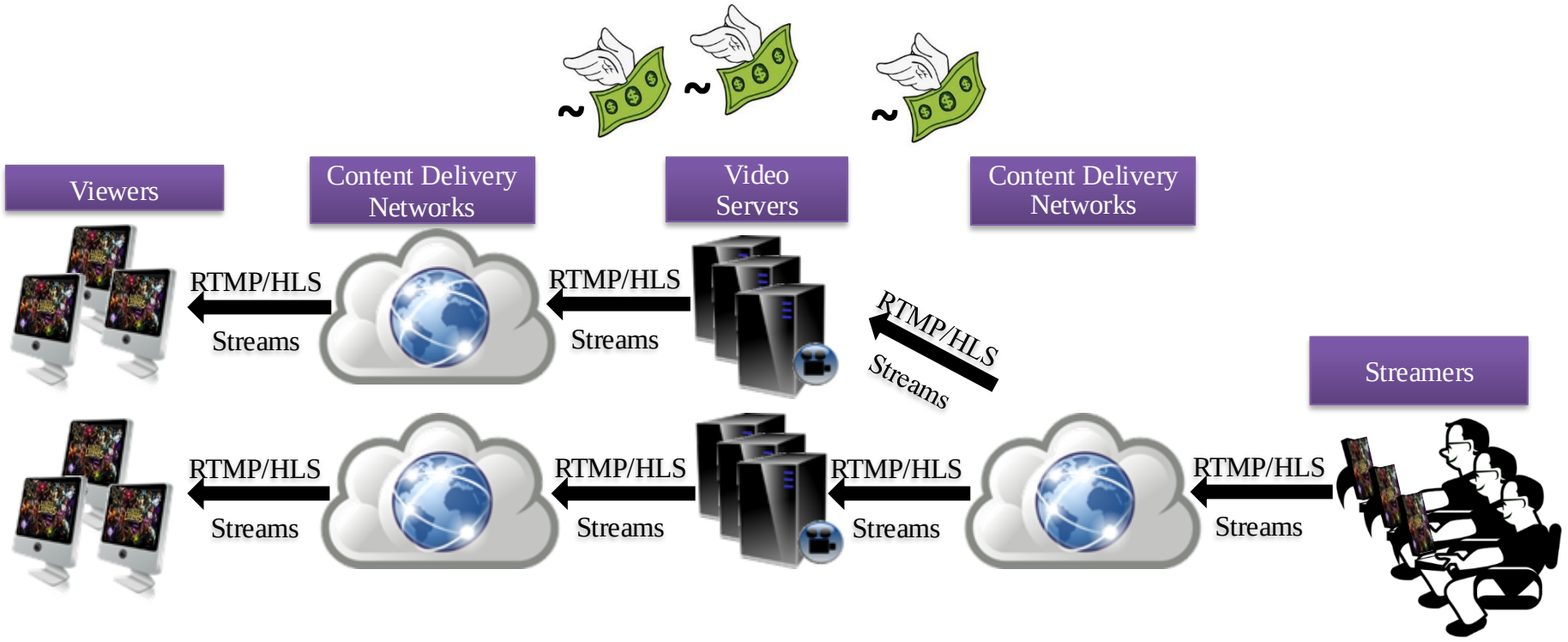
POSSIBLE SOLUTION

Add more resource

Reduce the video quality

Add More Resource

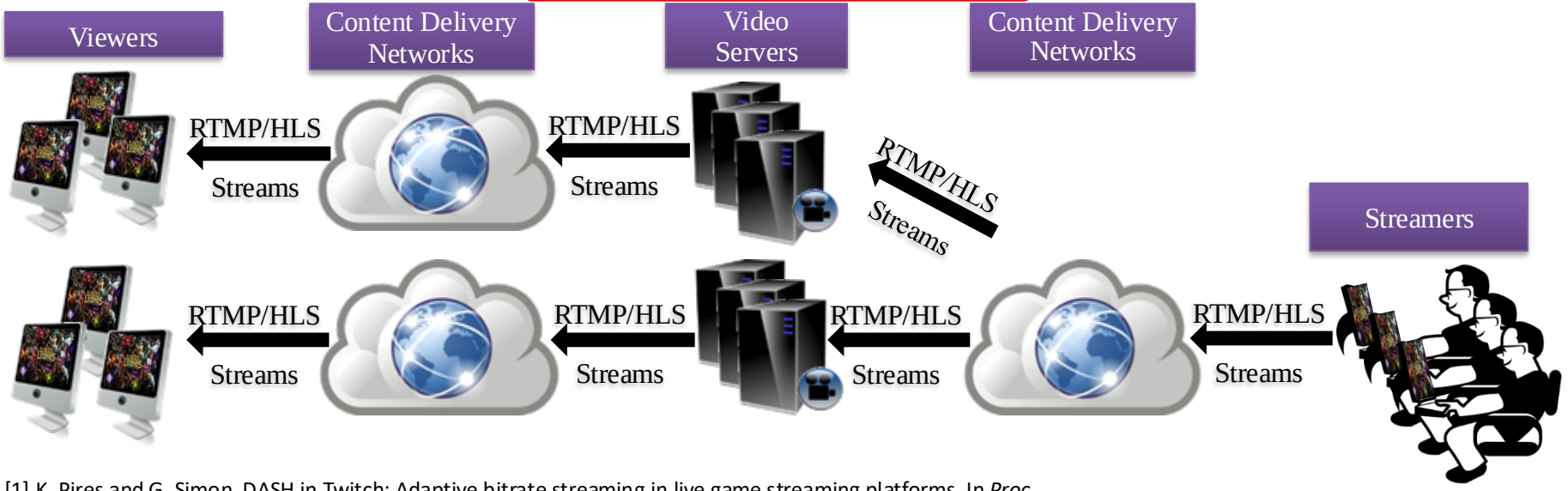
- Twitch upgraded it's transcoding server in 2014
  - Also doubled it's CDN bandwidth in Europe region



Add More Resource

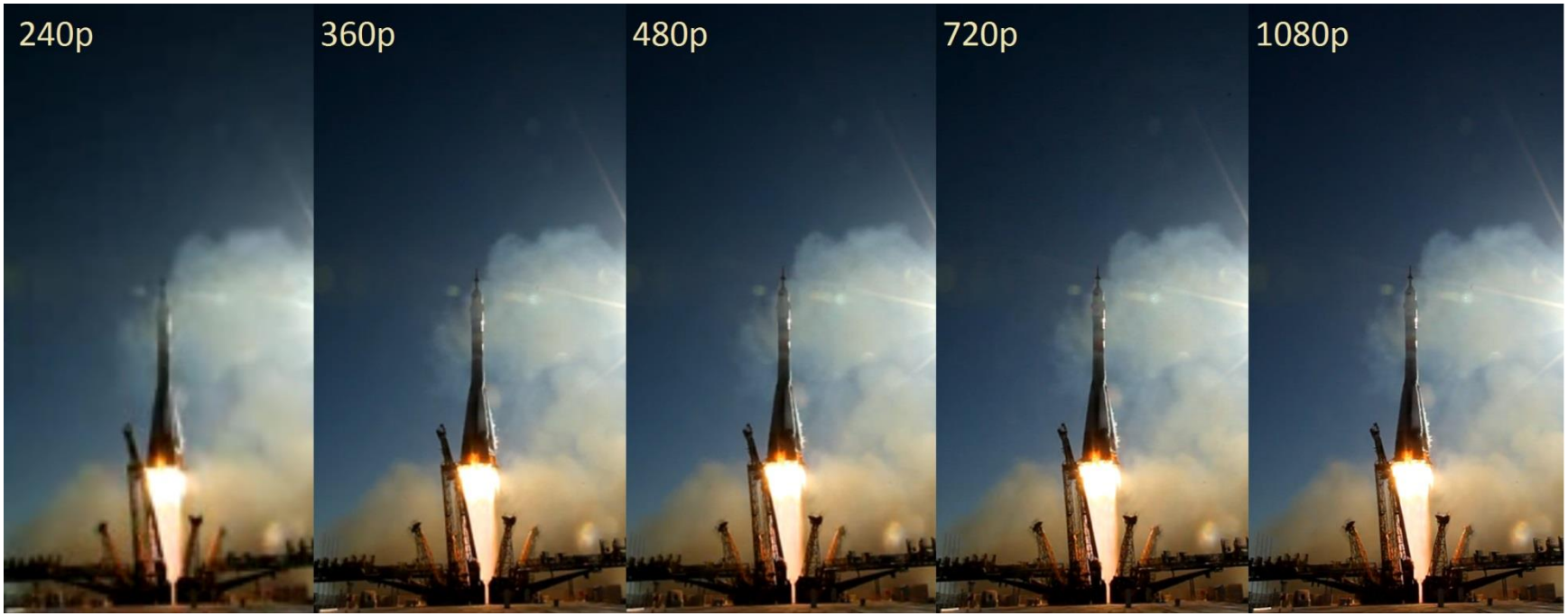
- Twitch have avg 1Tbps bandwidth usage in 2014 [1]
  - May cost as much as $ 11.34 million per month

Bad for business

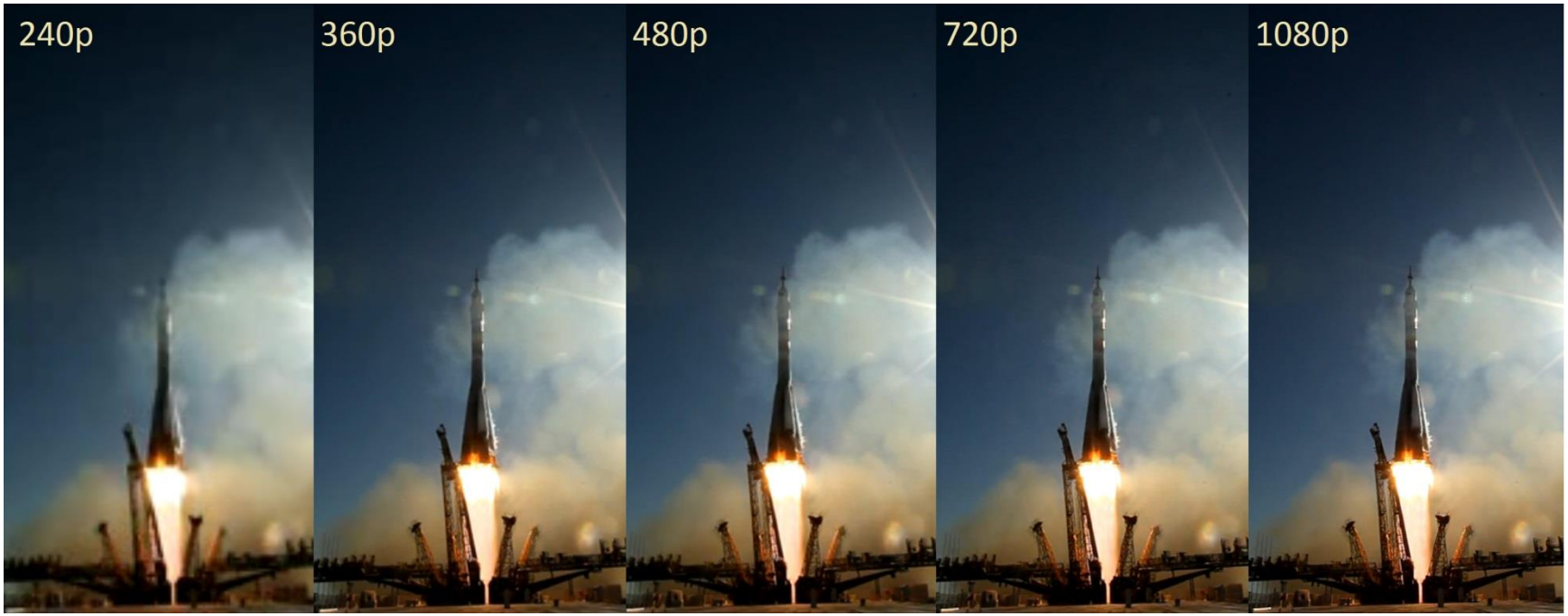


[1] K. Pires and G. Simon. DASH in Twitch: Adaptive bitrate streaming in live game streaming platforms. In *Proc. of ACM Workshop on Design, Quality and Deployment of Adaptive Video Streaming (VideoNext'14)*

Lower Video Quality



Lower Video Quality



How to do this without degrading user experience?

How To Leverage SoI

- Devote more resource into streaming SoI
  - Lower the quality of non-SoI if necessary
- Higher user experience
- Lower bandwidth consumption

How To Leverage SoI

- Devote more resource into streaming SoI
  - Lower the quality of non-SoI if necessary
- Higher user experience
- Lower bandwidth consumption

Wait, how to determine SoI?

Detecting SoI

- The game is running on the machine of streamer
  - Unlike other kind of live streaming
- Collect features in the system
  - CPU/GPU usage, keyboard input event…e.t.c.
  - Use these features to help us determine