RISE: Robust Internet Streaming Evaluation

Richard Alimi, Chen Tian, Xuan Zhang,
Y. Richard Yang, David Zhang (PPLive)

Laboratory of Networked Systems
Yale University

Peer-to-peer Research Group, IETF78

Motivation

P2P live streaming has multiple crucial algorithms

- Topology management
- Piece selection
- Rate control
- etc

Developers continue to improve P2P live streaming

- Tune existing algorithms
- Develop new algorithms

Limitations of Existing Testing Techniques

Lab testing

- Limited in scale
- May not capture real user environment

Modeling and simulation

- May not sufficiently describe actual behavior
- Difficult to model real network environment

Deploy to "test" channel

- Users see poor quality if new algorithms don't work well
- Difficult to control testing scenarios

RISE Objective

Objective: *Test new system with real users*

- Experimental System
  - Developer may control experimental conditions
    - Number of peers, types of peers, arrival times, etc
  - Developer may gather performance metrics
    - Measured performance should be accurate
- Protection of User Experience
  - User should observe at least as good quality as original system
- *Major issue is scalability*
  - Wish to support hundreds of thousands of peers

Key Techniques of RISE

Scalable Streaming Protection

- Existing (*stable*) and experimental algorithms run in parallel
- Goals
  - Protection for user experience
  - Accurate measurements of experimental system

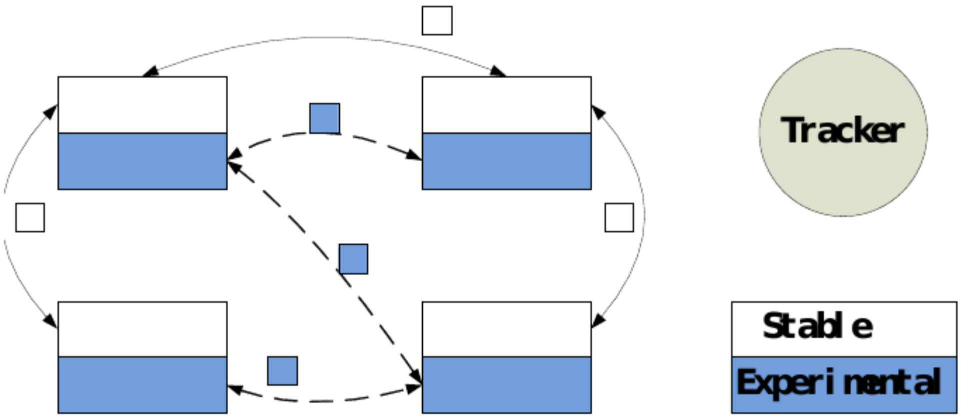
Distributed Experimental Control

- Lightweight, scalable control mechanism
- Goal
  - Developer defines experimental scenario
- *NOTE: only provide brief overview due to time constraint*

Scalable Streaming Protection: Architecture

Stable and Experimental algorithms run in parallel

- Logically separate channels, but same pieces
- Stable will serve as rescue



Problem Formulation

- How do we assign tasks and resources to Stable and Experimental systems to achieve both disruption protection and accuracy?

Scalable Streaming Protection: Requirements

Notations

A	Task assignment (T) and resource assignment
A_{exp}	Tasks and resources assigned to *Experimental*
A_{stable}	Tasks and resources assigned to *Stable*

Two requirements

- R1: Disruption protection $\boldsymbol{Perf(A) >= Perf(A_{stable})}$
- R2: Experimental accuracy   **obtain** $\boldsymbol{Perf(A_{exp})}$

Scalable Streaming Protection: Methods

Scale-invariant streaming

- Identify class of algorithms and settings as *scale-invariant*
  - Simple scheduling assignment to achieve R2 and R3

Virtual Playpoint Shifting

- Used for other algorithms and network settings

Scale-invariant Streaming

For a class of algorithms and network settings, if we

- scale channel (streaming) rate by *α* (e.g., 1/5)
- scale the upload capacities of end-hosts by same *α*

then certain performance metrics remain unchanged

- No need to know relationship between performance and input parameters
- Easier to achieve R1 (protect user experience) with small *α*

Do scale-invariant systems exist?

- Positive results for limited settings and metrics
- Experience shows it is difficult to achieve

Virtual Player Platform

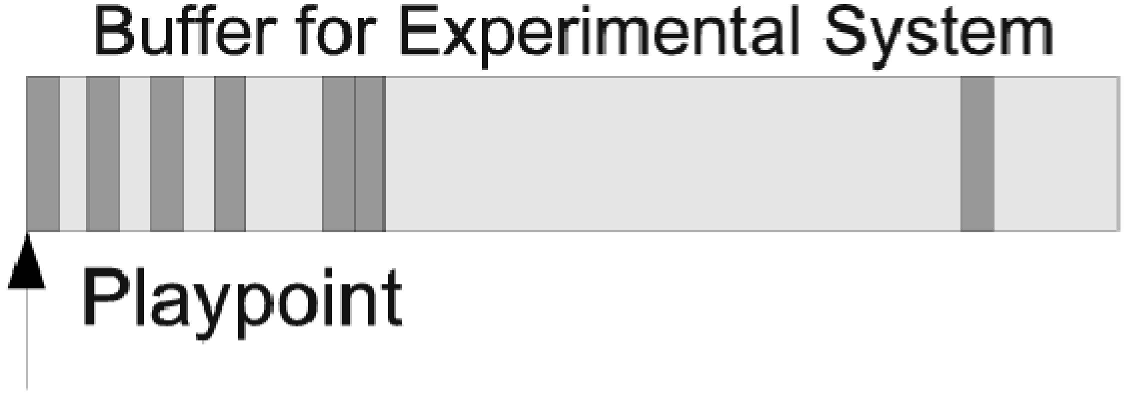
Virtual Player Platform

- Technique to achieve Scalable Streaming Protection

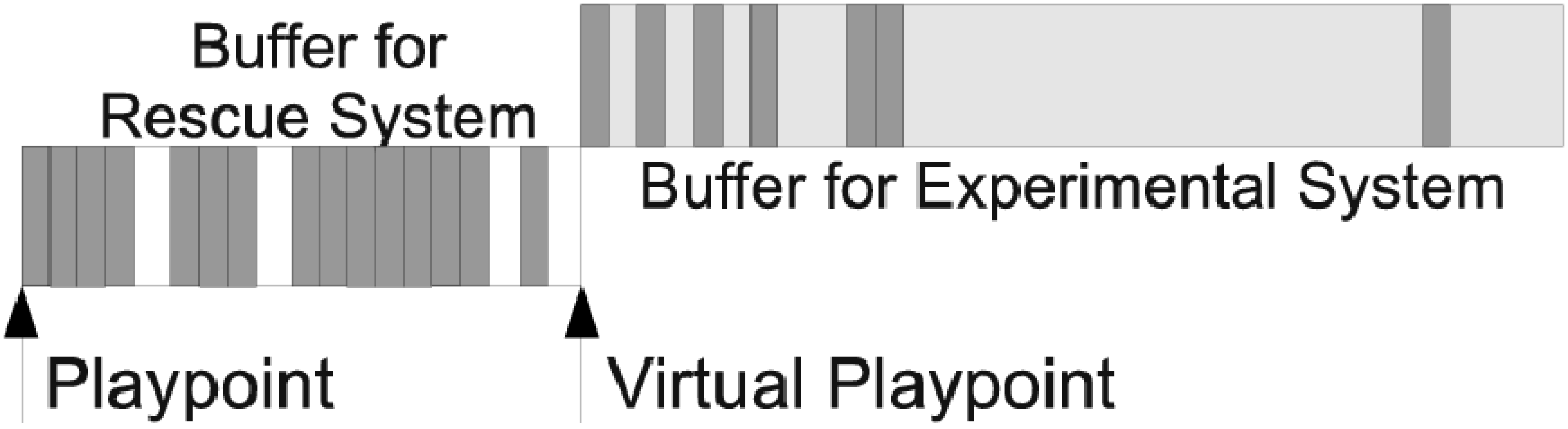
Basic Idea

- Try to give Experimental the same *tasks*, amount of *resources*, lag from *source*, *deadlines*, and *block availability* as if running alone
- When Experimental misses a piece's deadline, task shifted to Stable
- *Stable* given some time ($T_{recover}$) to recover missed pieces
  - User playpoint has lag compared to Experimental (virtual) playpoint

Virtual Player: Basic Idea



Virtual Player: Basic Idea



Virtual playpoint shifts with true playpoint

Piece responsibility flow

- Experimental → Stable
- ~~Stable → Experimental~~

Virtual Player Analysis

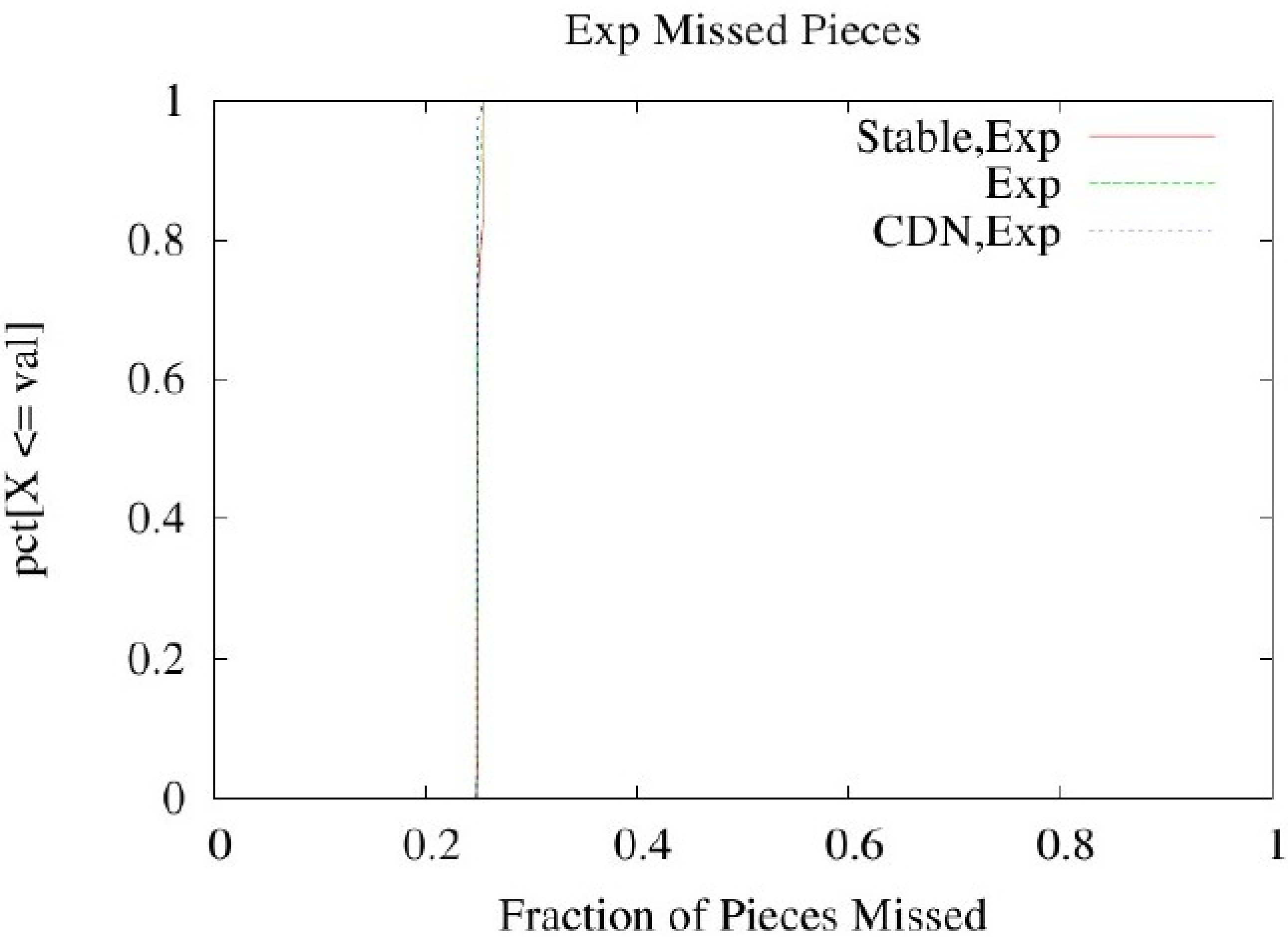
Experimental Accuracy

- High accuracy when experimental algorithm performs well
- Measured performance is lower bound if Stable triggered
  - Due to resource competition

Overhead

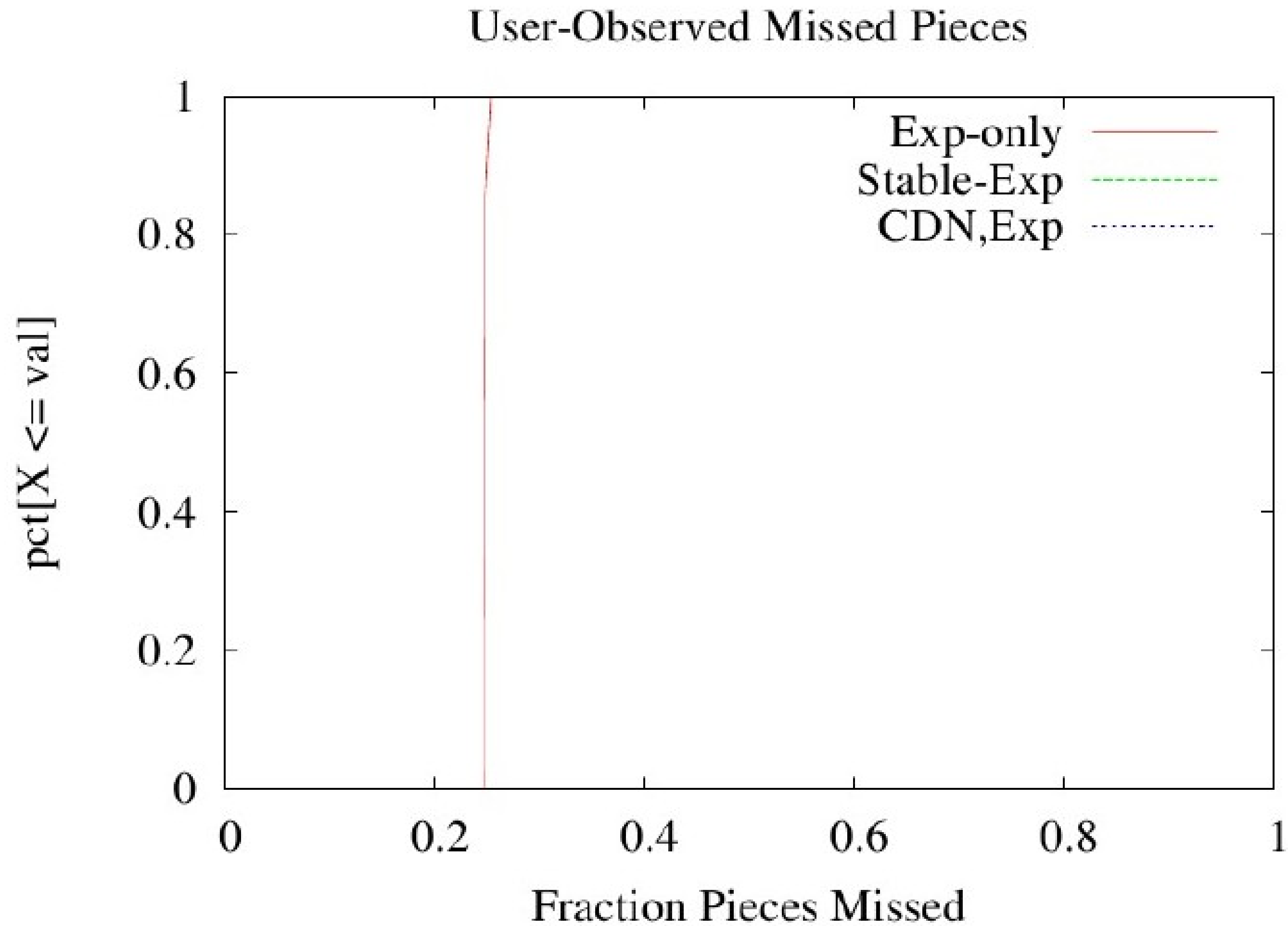
- Additional lag from source may not be tolerable in all cases

Virtual Player Evaluations



Accurate measurement of missed piece ratio

Virtual Player Evaluations



Stable system successfully recovers missed pieces

Experimental Control

Objective

- Allow developer to define experiment conditions

Experiment *scenario* defined by

- Peers selected to run experimental algorithm
  - Peers identified by properties (estimated capacity, location, etc)
- Arrival behavior
  - "Discretized" version of non-homogeneous Poisson process
  - Example: developer wishes to experiment with flash crowd
- Departure conditions
  - Example: to model user behavior (depart after 2nd freeze within 3 mins)

Experimental Control: Making it Distributed

Basic Idea

- Tracker distributes scenario *parameters* to selected peers
  - Each peer gets same schedule
  - May be distributed via P2P overlay, tracker keepalive, CDN, etc
- Peers *locally* compute arrival time based on schedule
  - Resulting arrival process approximates target

Challenges

- Handling peers that prematurely depart
  - Example: user terminates the client software
  - Must handle detection and replacement of peer

Conclusions

RISE aims to enable evaluation of streaming algorithms with real users

Current and future work

- Continue design and implementation of experimental control
- Continue exploration with scale-invariant systems
- Framework for debugging system components
  - Inference model to determine performance bottlenecks