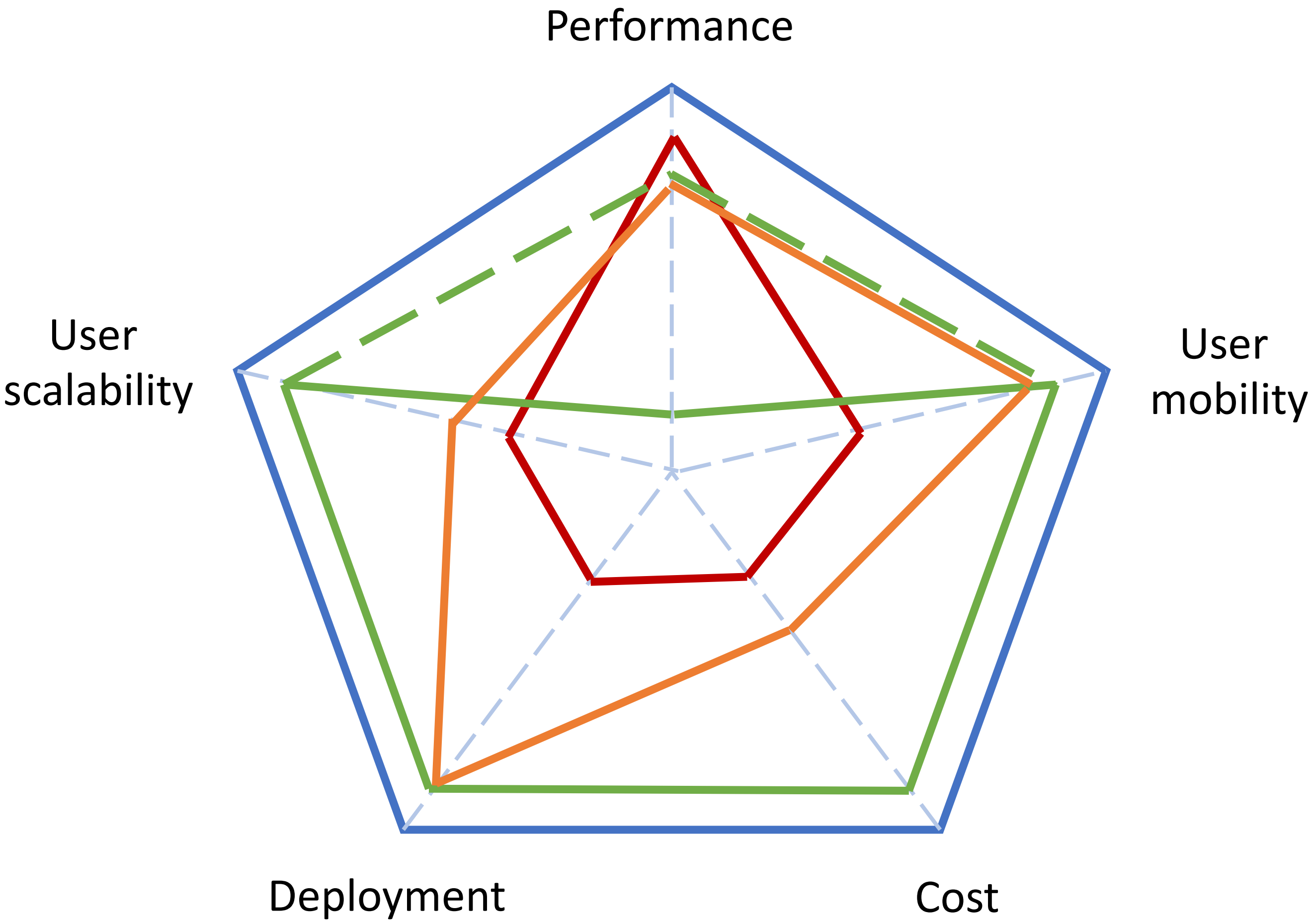
Firefly: Untethered Multi-user VR for Commodity Mobile Devices

Xing Liu, Christina Vlachou, Feng Qian, Chendong Wang, Kyu-Han Kim

UNIVERSITY OF MINNESOTA
Driven to Discover®

Hewlett Packard Labs

Motivation


Performance
User mobility
Cost
Deployment
User scalability

State-of-the-art

- Flashback (Mobisys 2016) – Aggressive prerendering, local memorization.
- Furion (Mobicom 2017) – Pipelining, offloading.

Firefly

- A low cost and easy to deploy colocation multi-user VR system that supports…

✓ *10+ users with mobility*

✓ *High quality VR content*

  ✓ *60 FPS*

  ✓ *Low motion-to-photon latency*

  ✓ *Quad HD*

✓ *Single server/AP, commodity smartphones, cheap VR headsets (e.g. google cardboard)*

- Team training, group therapy, collaborative product design, multi-user gaming…

Challenges

- Weak mobile GPU
- Energy/heat constrains
- Heterogeneous computing capabilities
- Multi-user scalability
  - Client-server load split
  - Single AP bandwidth limitation

Outline

- Overview
- Firefly System Components
- Evaluation
- Summary

High Level Architecture

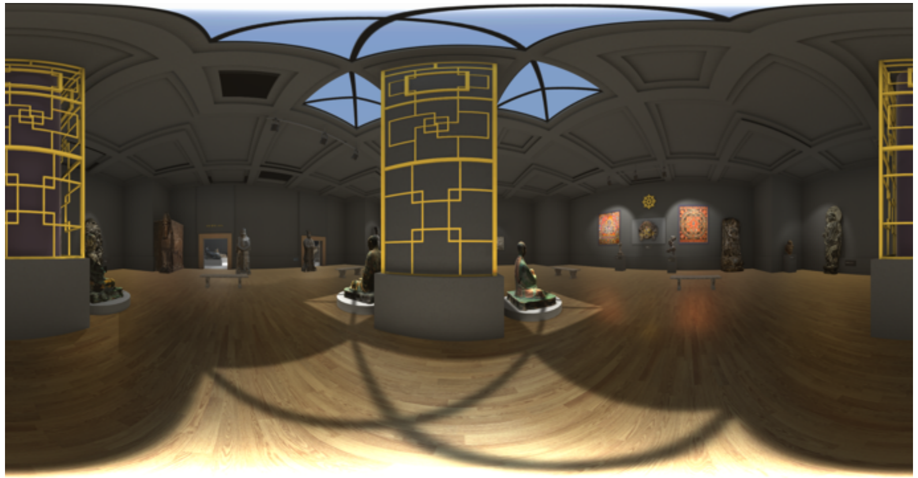
- A Serverless Design
  - full-fledged client rendering
  - far from being powerful enough
- Edge offloading
  - server real-time rendering, streamed as encoded VR frames
  - high encoding overhead for single server (~150 FPS for Quad HD)
- Performs One-time, Exhaustive Offline Rendering
  - Offline: prepare all possible VR viewports, encodes as video stream
  - Online: streams based on VR motion
  - eliminates rendering/encoding overhead, scales to tens of users, at the cost of high mem, disk and network usage.

Outline

- Overview
- Firefly System Components
- Evaluation
- Summary

Offline Rendering Engine

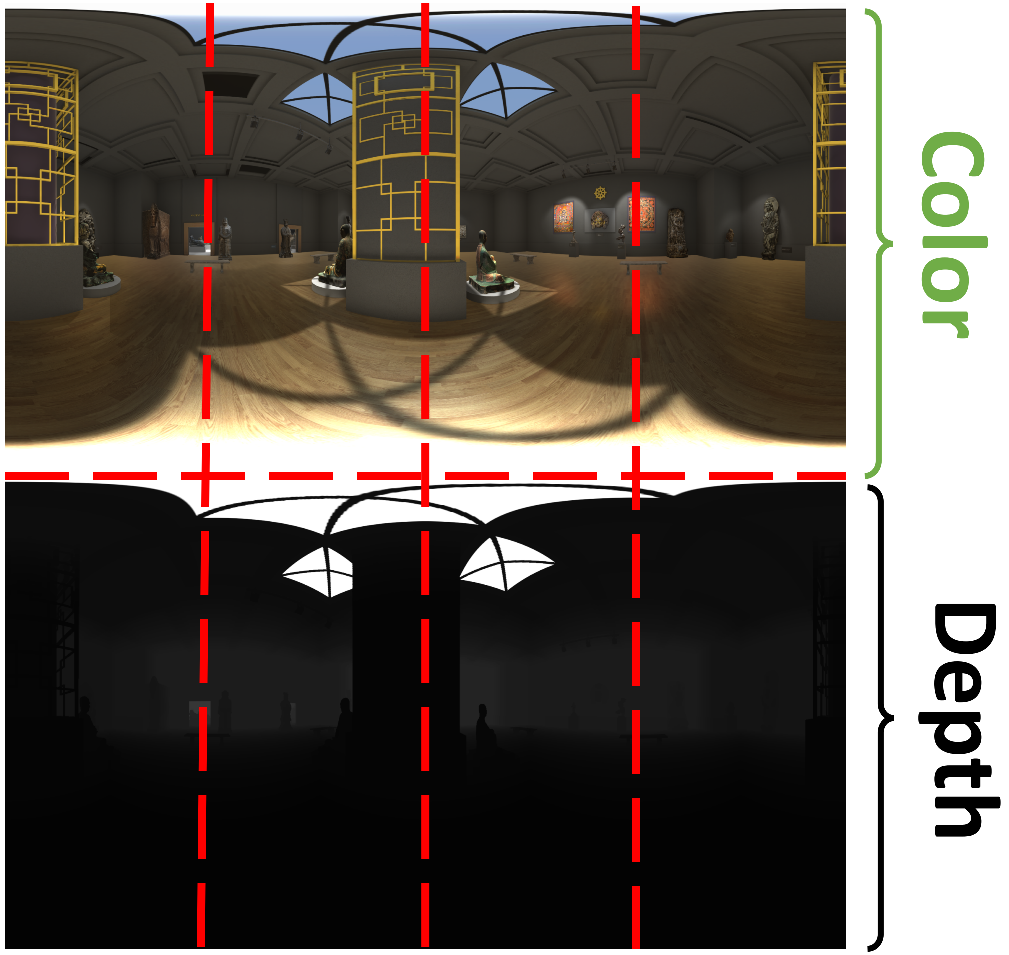
- Populates the content DB by…
  - Discretizing whole VR space into grids
  - At each grid, renders a *panoramic* VR frame (360° view)



Client
Projection

Offline Rendering Engine

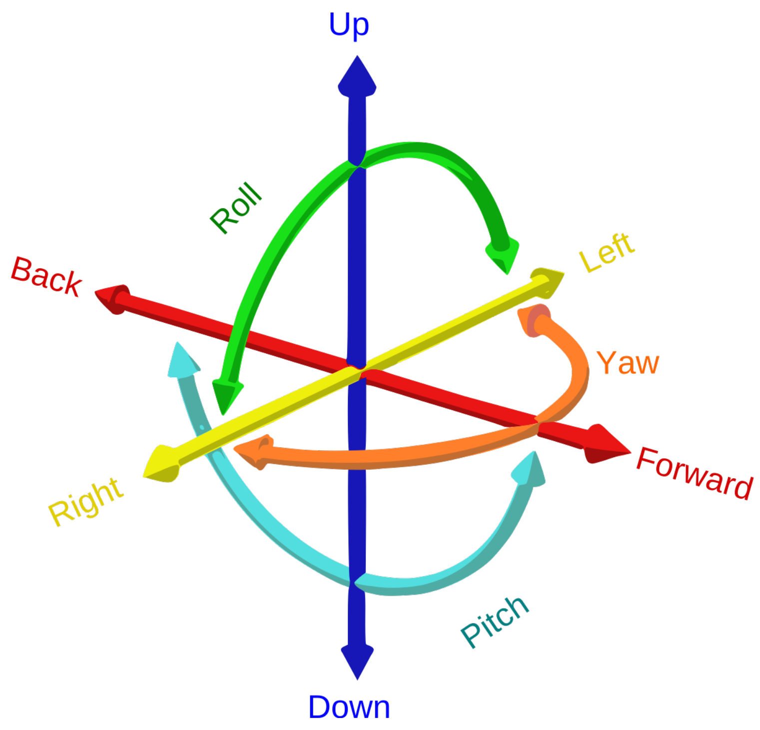
- Tiles
  - Independently transmitted & decoded
  - Streams at tile level
    - Finer fetching granularity
    - Bandwidth saving
- *Office* vs. *Museum*
  - Map size: 30 X 30 m
  - Grid size: 5cm
  - Size: 137GB vs. 99GB



Mega Frame

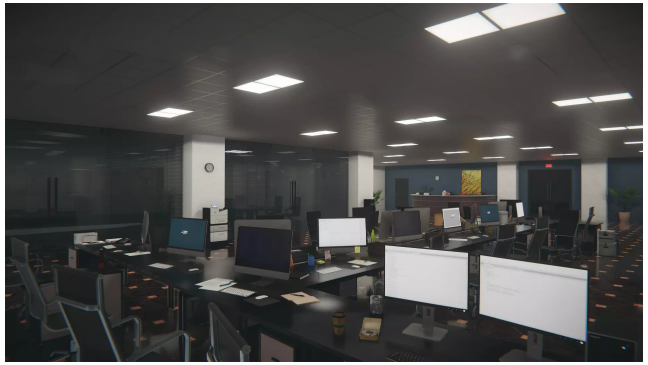
How to fetch tiles?

- 6 degree of freedom (DoF)
  - Translational
  - Rotational
  - *(x, y, z, yaw, pitch, roll)* -> tile *x*
- Fetch based on user's VR motion
  - End-to-end latency estimation: *3ms + 30ms + 34ms + 3ms = 70ms*
    req trans decode render
  - Motion-to-photon latency requirement: *1000ms / 60FPS = 16.7ms*

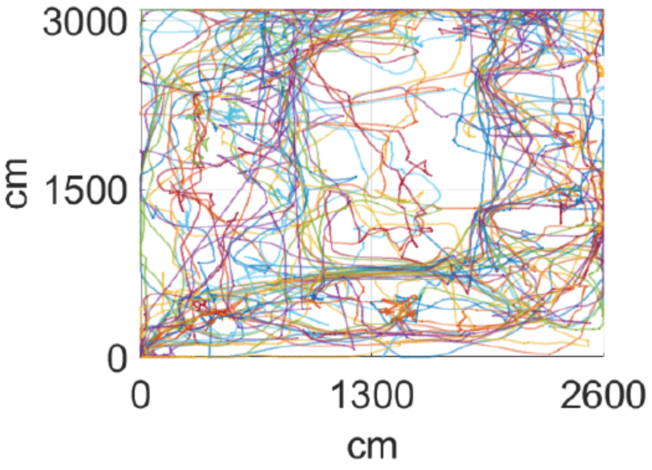


Understanding VR Motion Predictability

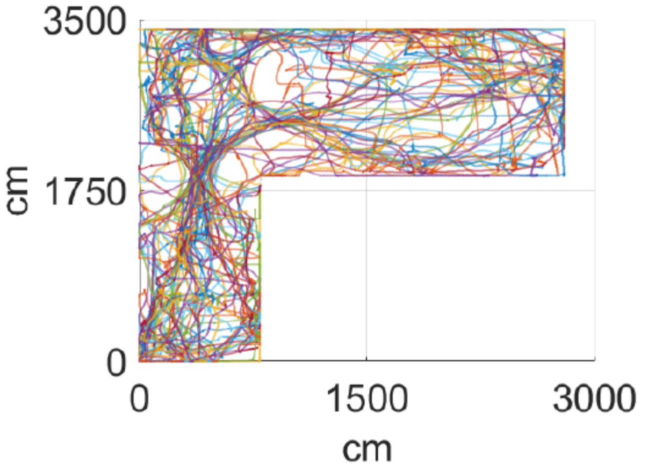
- VR user motion data collection
  - 25 participants
  - Unity API (*Office*, *Museum*)
  - 6-DoF motion enabled by Oculus Rift
  - 6-DoF trajectory recorded
  - 50 5-min VR trajectory traces



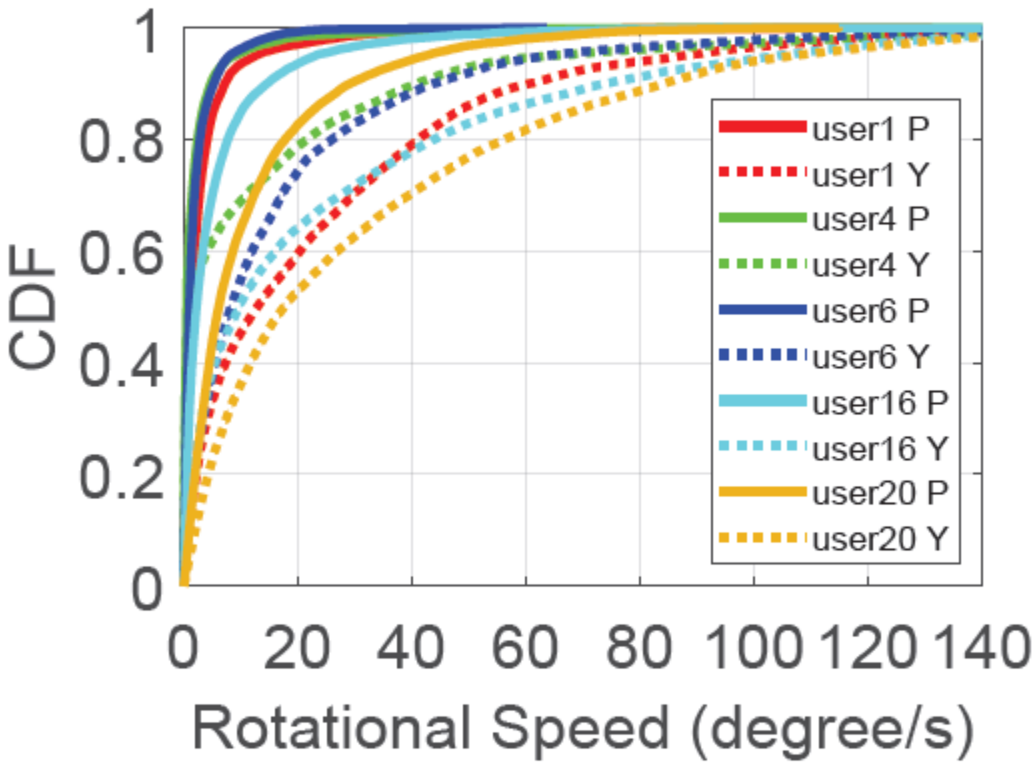
VR User Motion Profile



Museum

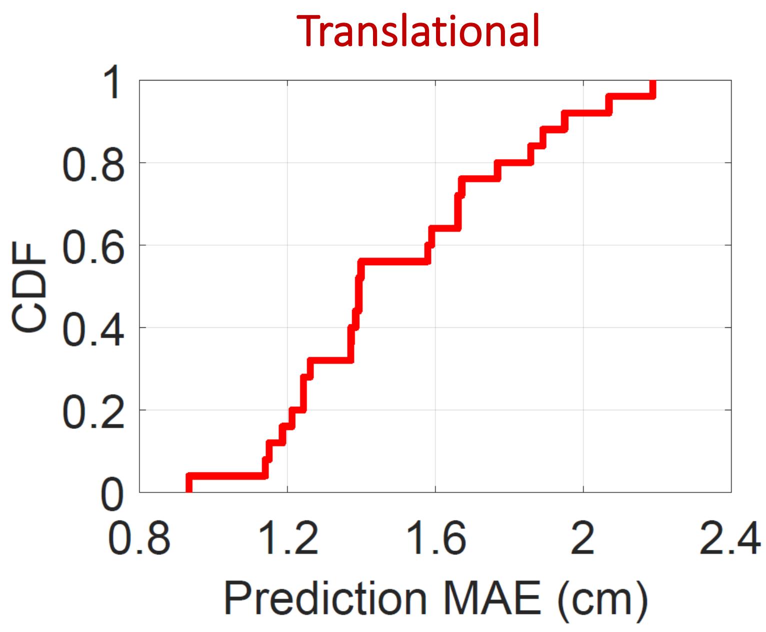


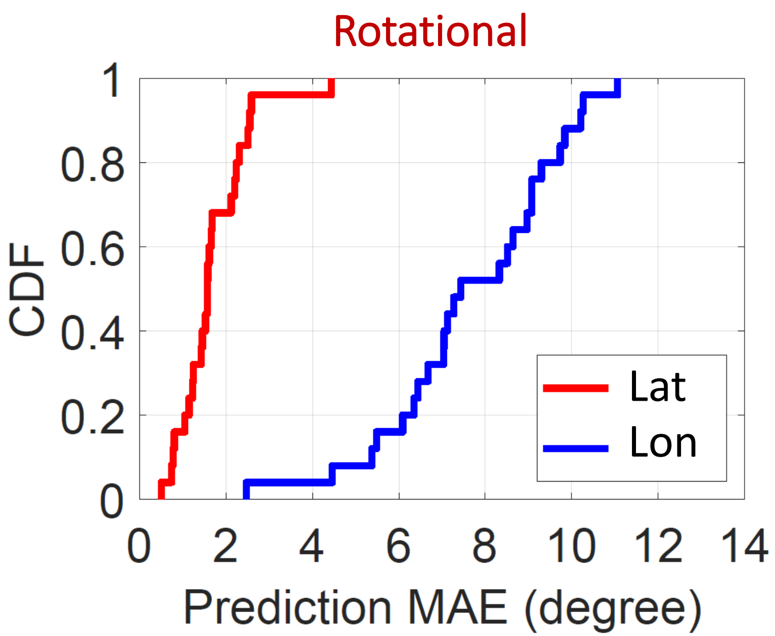
Office



Understanding VR Motion Predictability

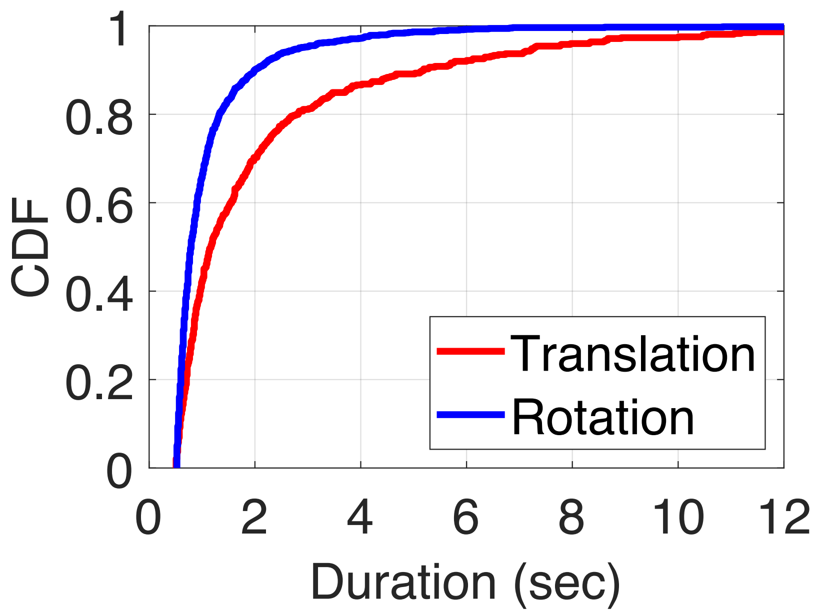
- A simple Linear Regression (LR) model (*H*=50ms, *P*=150ms)
- MAE_{trans} = 1.4cm, MAE_{lat} = 1.9°, MAE_{lon} = 7.6° (**FoV 100° x 90°**)
- Predict each dimension separately





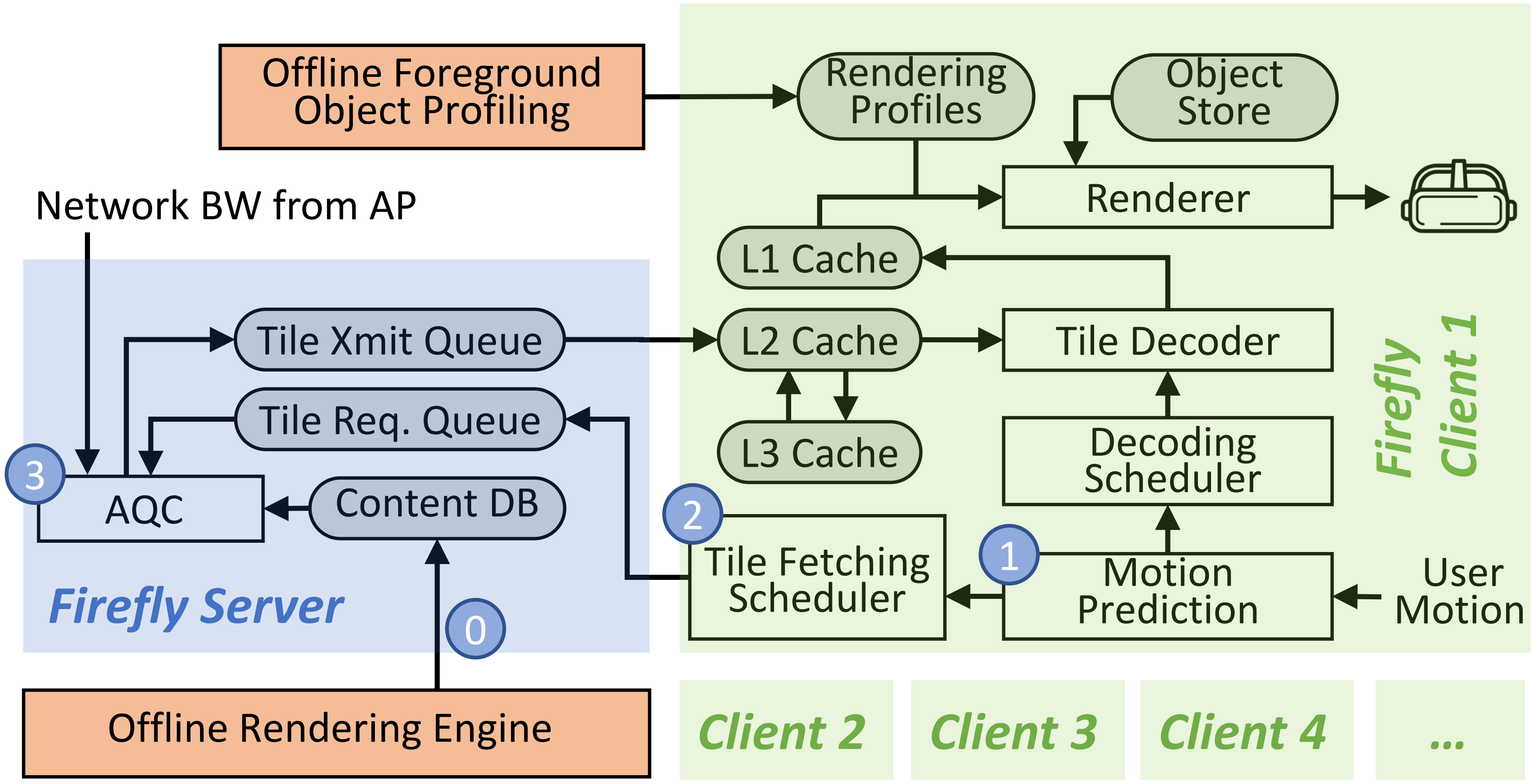
VR User Stationary Periods (SP)

- Within a 5-min VR session…
- 43 seconds of SP
- SP are short (~ 1s), but frequent
- Sudden movements makes prediction unavailable
  - Moving – fetch based on prediction
  - Stationary – fetch (best-effort) neighboring tiles



<u>TAKE AWAY</u>**:**
1. Users' motion profile are diverse.
2. Good predictability for continuous VR motion.
3. Need to handle sudden movements.

System Architecture



0 Offline rendering engine populates content DB

1 Lightweight motion prediction for frequent viewport updates

2 Interprets prediction results into a ranked list of tiles

3 Adaptive Quality Control (AQC)

Features

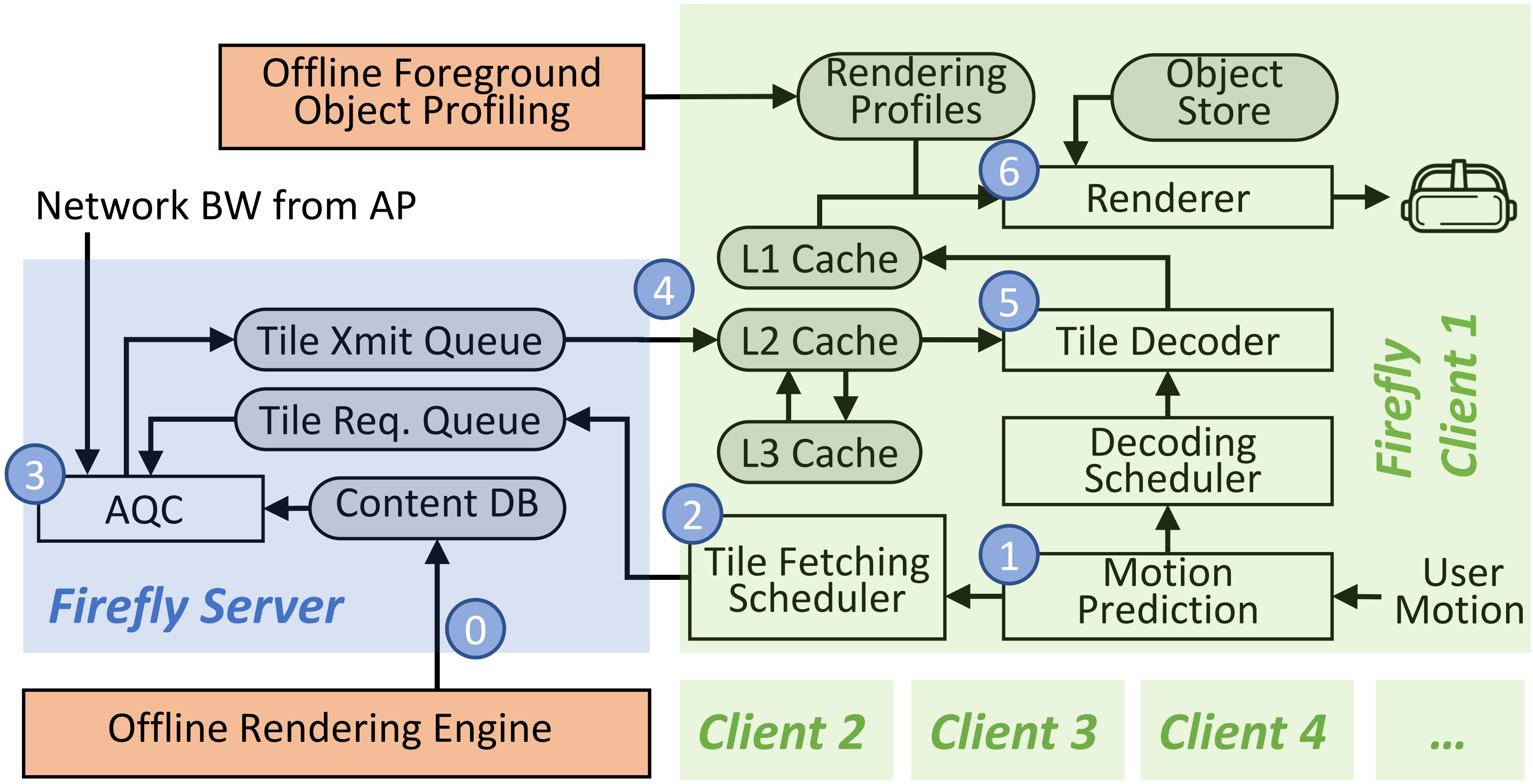
- *maximize the quality level, minimize stall and quality switch*
- *Fairness among users*
- *Fast pace*
- *Scale more users*
- *Optimization vs. heuristics*

Adaptive Quality Control (AQC)

```
n: total number of users
T: total available bandwidth across all users
Q: users' current quality levels (input & output)
Tiles: users' to-be-fetched tile lists (input)
Q': local copy of Q
B: individual user's available bandwidth
λ: bandwidth usage safety margin
RESERVE: reserved bandwidth for each user
bw_util: estimate bandwidth requirement for the request
```

```
T = get_total_bw_from_AP() * λ
Q'[1..n] = Q[1..n]
B[1..n] = get_individual_bw_from_AP([1..n]) * λ
foreach user i:
    while (bw_util(Tiles[i],Q'[i])≥B[i] and Q'[i] is not lowest):
        Q'[i] = Q'[i] - 1
    T = T – min(B[i], max(RESERVE, bw_util(Tiles[i], Q'[i])))
if (T < 0):
    lru_decrease(Q'[1..n]) until (T≥0 or Q'[1..n] are lowest)
else:
    lru_increase(Q'[1..n]) until (T≈0 or Q'[1..n] are highest)
Q[1..n] = Q'[1..n]
```

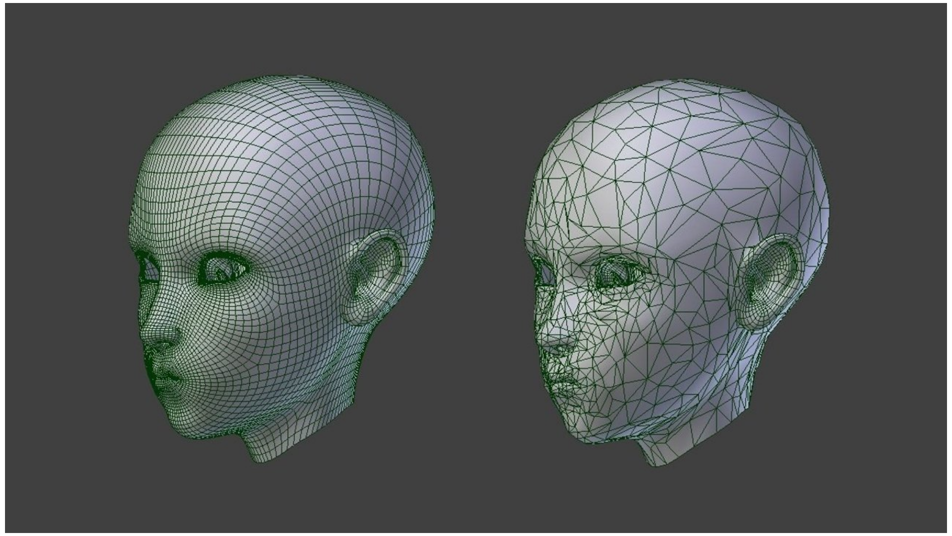
System Architecture



0. Offline rendering engine populates content DB
1. Lightweight motion prediction for frequent viewport updates
2. Interprets prediction results into a ranked list of tiles
3. Adaptive Quality Control (AQC)
4. Hierarchical cache, L3 disk, L2 main mem, L1 video mem
5. Hardware accelerated concurrent decoders, tile decoding
6. Tiles rendering, foreground object overlaying

Dynamic Foreground Objects

- Other users' avatars, control panel, *etc.*
- Foreground objects are rendered locally real-time
  - Pre-render not feasible
  - Less rendering compared with complex backgrounds
  - Latency sensitive
- Adaptive object rendering
  - Prepare lower quality by mesh simplification
  - Dynamic decision

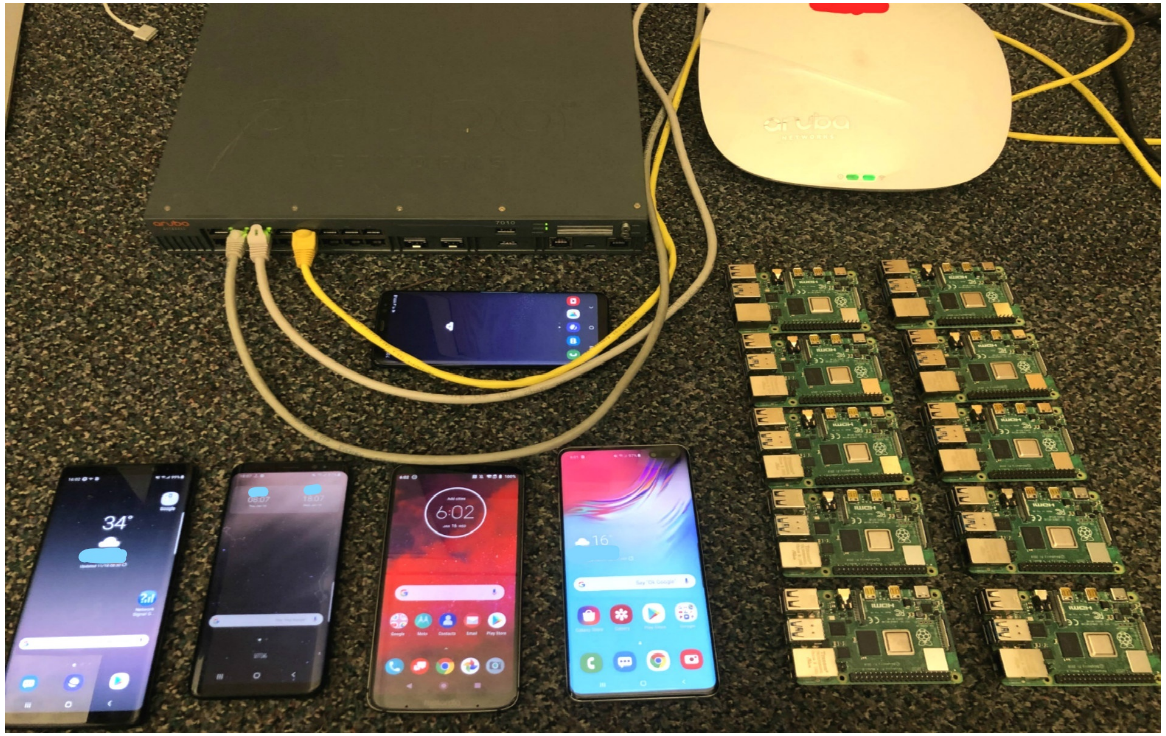


Outline

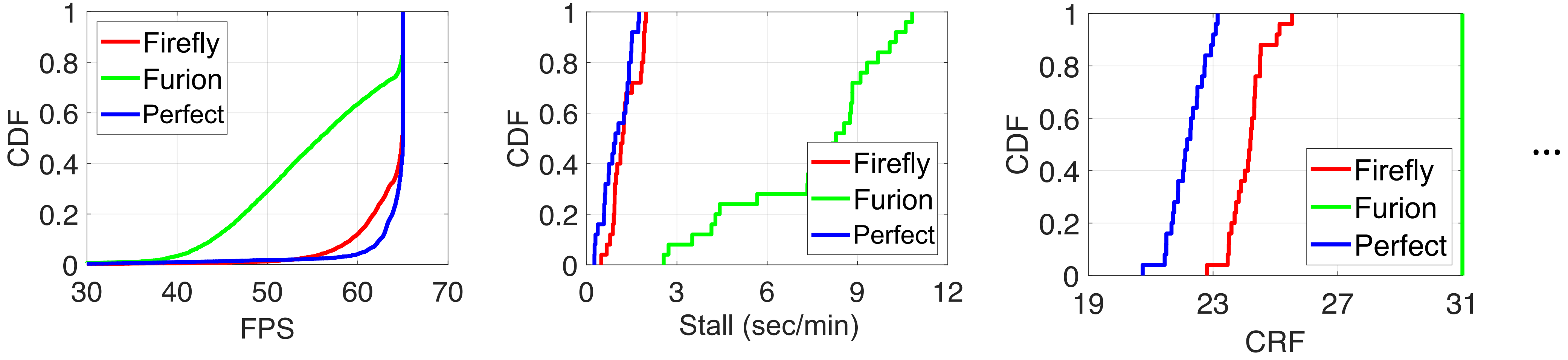
- Overview
- Firefly System Components
- Evaluation
- Summary

Implementation and Evaluation Setup

- Offline rending engine: Unity API and ffmpeg, C#/Python (LoC 1,500)
- Client: Android SDK (LoC 14,900)
  - Tile decoding: Android MediaCodec
  - Projection/rendering: OpenGL ES
  - L1 cache: OpenGL frame buffer object (FBO)
- Server: Python (LoC 1,000)
- “Replayable” experiment (15 devices)
  - SGS8 x 2, SGN8, MOTO Z3, SGS 10
  - Raspberry Pi_4 x 10
  - Server colocates with AP in a VR lab
  - Clients randomly distributed



Overall Performance Comparison



Firefly vs. multi-user Furion vs. Perfect

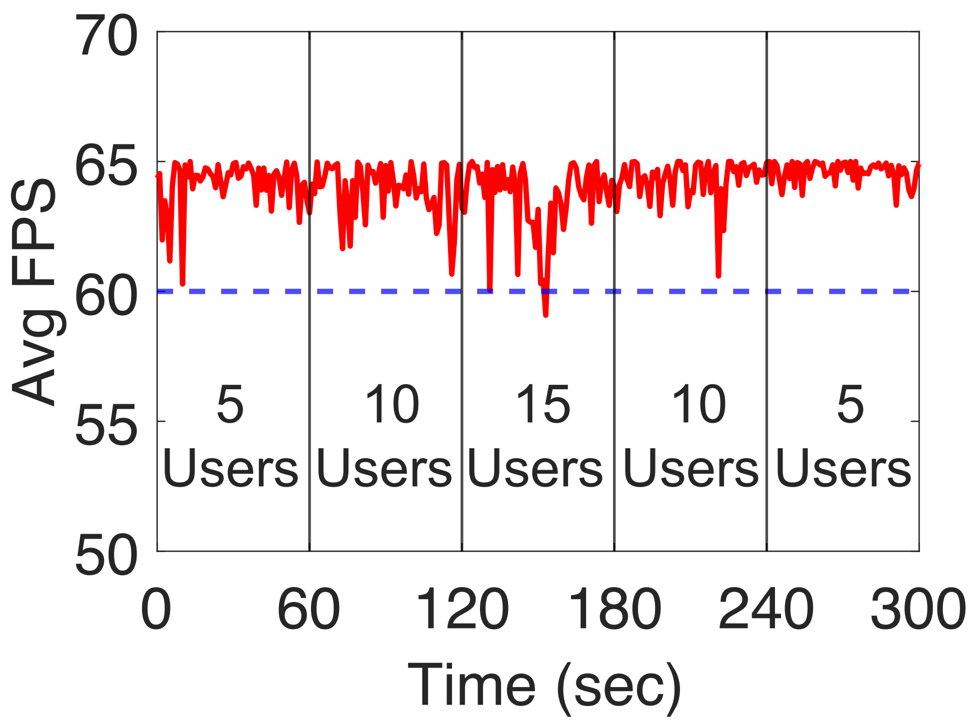
- *FPS, stall, content quality, motion-to-photon delay, inter-frame quality variation, intra-frame quality variation, fairness*
- Overall, Firefly achieves good performance across all metrics
- 90%/99% of the time FPS ≥ 60/50
- Stall = 1.2 sec/min
- Bandwidth consumption (15 users) < 200 Mbps
- Quad HD (2560 x 1440) with average CRF = 23.8

Micro Benchmarks

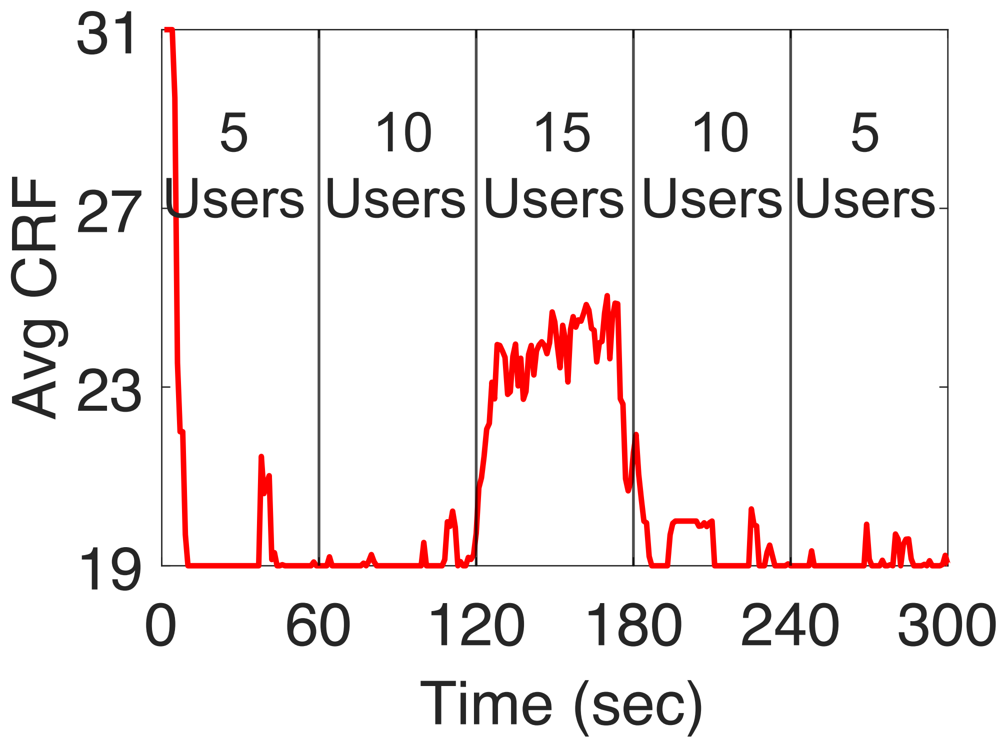
- Impact of AQC
- Impact of Bandwidth Reservation for stationary periods
- Impact of different viewport prediction strategy
- Impact of adaptive object quality selection
- …

Case Study - Adaptiveness to Number of Users

Average FPS



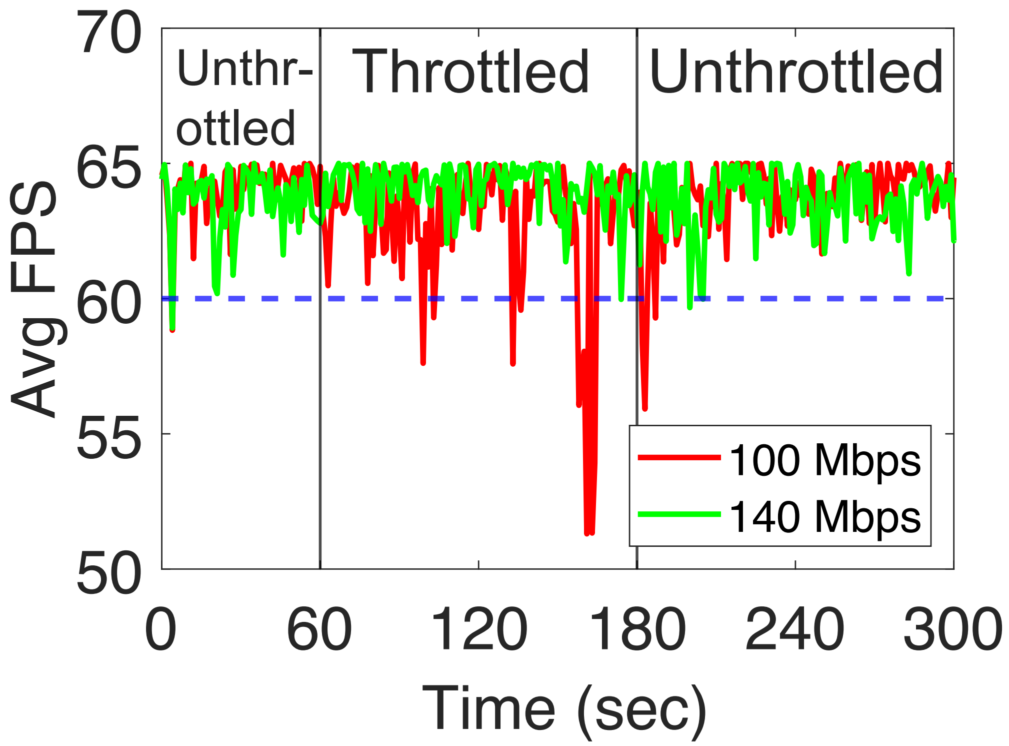
Content Quality



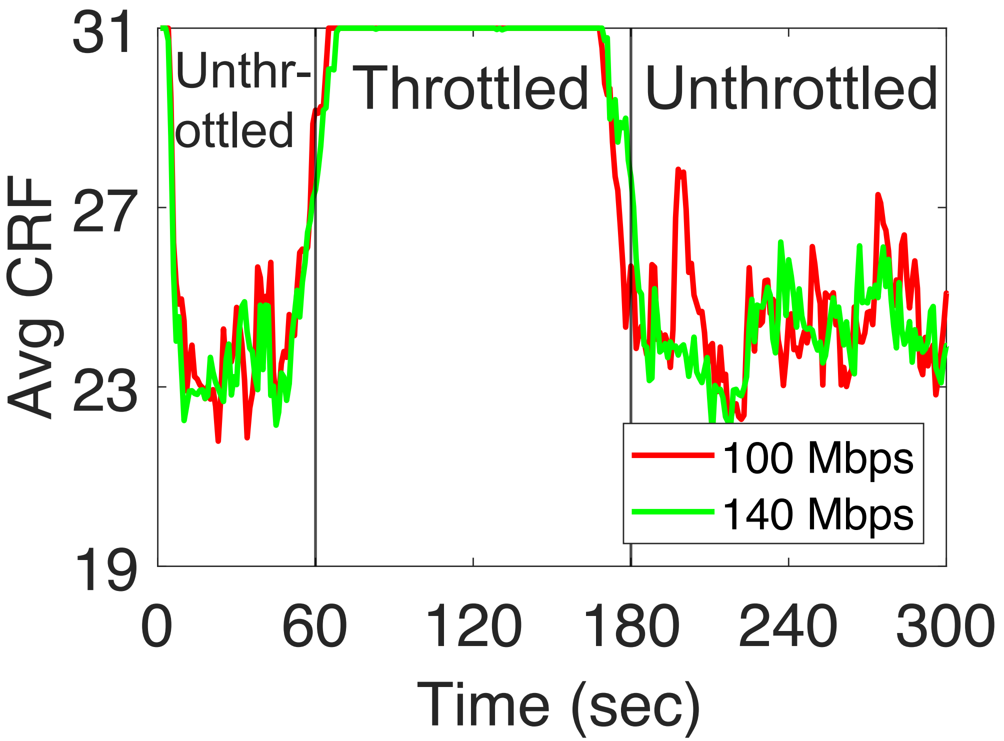
The global available bandwidth is throttled at 200 Mbps

Case Study - Adaptiveness to Available Bandwidth

Average FPS



Content Quality



15 concurrent users

Energy Usage and Thermal Characteristics

- After 25 mins of Firefly client usage, a fully charged smartphone
  - Battery left: 92% ~ 96%
  - GPU temperature < 50°C

Summary

- Firefly supports 15 VR users at 60 FPS using commodity smartphones and a single AP/server.
- Our design makes judicious decisions on
  - partitioning the workload (offline vs. runtime, client vs. server)
  - making the system adaptive to the available network/computation resources
  - handling VR users' fast-paced motion
- Core concepts of Firefly are applicable to other multi-user scenarios (AR/MR)