Outatime: Using Speculation to Enable Low-Latency Continuous Interaction for Mobile Cloud Gaming

Kyungmin Lee*, David Chu†, Eduardo Cuervo†, Johannes Kopf†, Yury Degtyarev^, Sergey Grizan°, Alec Wolman†, and Jason Flinn*

University of Michigan*, Microsoft Research†, St. Petersburg Polytechnic University^, Siberian Federal University°
Mobile games are popular

People like games!

**Time Spent on iOS and Android Connected Devices**

- **14%** Browser
- **86%** Apps
  - **Gaming 32%**
  - Other 2%
  - Google Browsers 5%
  - Apple Safari 7%
  - News 3%
  - Productivity 4%
  - Utilities 8%
  - Entertainment 4%
  - Facebook 17%
  - Twitter 1.5%
  - Social Messaging 9.5%
  - (other)
  - YouTube 4%

Source: Flurry Analytics, ComScore, NetMarketShare

(Flurry 2014)

Kyungmin Lee
Emerging cloud gaming platform
Cloud gaming

+ Powerful servers, more immersive games
  (advanced graphics, better AI)
+ Few client dependencies
+ Easy maintenance & upgrades
+ Piracy is much more difficult
+ Instant gratification

- Network latency
- Bandwidth
- Server multi-tenancy
Cloud gaming with no RTT

- RTT=0ms, FPS=30fps (approx. 32ms per frame)
Impact of network RTT

- RTT=128ms, FPS=30fps (approx. 32ms/frame)
When network RTT is high...

Players are sensitive to as little as 60ms RTT!
Our solution: Outatime

Predict future inputs and generate future frame

No perceived RTT delay

Display predicted frame

Perform misprediction compensation

RTT=128ms, FPS=30fps (approx. 32ms/frame)
Outatime overview

Outatime client
- Decode
- Display
- Capture/send user input

Outatime server
- Receive user input
- Run game logic
- Render frame
- Encode

Kyungmin Lee
Outatime overview

Outatime client
- Decode
- Display
- Capture/send user input

Outatime server
- Receive user input
- Run game logic
- **Perform speculation**
- **Render speculated frame**
- Encode
Outatime overview

Outatime client
- Decode
- **Handle misprediction**
- Display
- Capture/send user input

Outatime server
- Receive user input
- Run game logic
- **Perform speculation**
- Render speculated frame
- Unwind misprediction
- Encode

Kyungmin Lee
Outatime overview

Supports up to 128ms RTT!

Outatime client
- Decode
- **Handle misprediction**
- Display
- Capture/send user input

Outatime server
- Receive user input
- Run game logic
- **Perform speculation**
- Render speculated frame
- Unwind misprediction
- Encode

Kyungmin Lee
Outatime overview

Outatime client
- Decode
- **Handle misprediction**
- Display
- Capture/send user input

Outatime server
- Receive user input
- Run game logic
- **Perform speculation**
- Render speculated frame
- Unwind misprediction
- Encode

**Supports up to 128ms RTT!**
Taxonomy of input events

• Navigation
  – LOOK up/down left/right
  – MOVE forward/back left/right jump/duck

• Impulse
  – FIRE a gun
  – SWING a sword
  – CAST a spell

• Delay tolerant
  – Open in-game menu, switch weapons, ...
Speculation for navigation events

Navigation Timeline

RTT = 8 ticks (256ms)

Left

Look right 2°

Markov prediction
How accurate is it?

- Forward/Back
- Left/Right
- Up/Down
- Yaw (Horizontal view)
- Pitch (Vertical view)

(Doom 3)

Accuracy

Network Latency RTT (ms)

64 128 256

Kyungmin Lee

Network Latency RTT (ms)
Handling misprediction: Image based rendering (IBR)

Input Image + Input Depth → Output Image
Cube map along with IBR

99% error coverage (degrees) as RTT increases

BACK not shown

Microsoft Research

Kyungmin Lee
Speculation for Impulse events

**Impulse Timeline**

$\text{RTT} = 8$ ticks (256ms)

$t_0 \rightarrow t_1 \rightarrow t_2 \rightarrow t_3 \rightarrow t_4 \rightarrow t_5 \rightarrow t_6 \rightarrow t_7 \rightarrow t_8$

**Speculative Sequences**

$X$: fire

$\sim X$: no fire

[Diagram showing speculative sequences with 256 choices]

Kyungmin Lee
Time-shifting mechanism

Impulse Timeline

RTT = 8 ticks (256ms)

shift forward
shift backward
shift forward

Speculative Sequences

X: fire
~X: no fire

Kyungmin Lee
Outatime
Improving Cloud Gaming w/
Speculative Execution
Mean Opinion Score (MOS) vs. Network Latency RTT (ms)

- **Perfect**
- **Excellent**
- **Barely acceptable**
- **Annoying**
- **Would not use**

**Controls were fluid, animations worked as expected, no rendering issues on movement and no transition issues with textures**

**Frame rate comfortable; liked the smoothness**

**Felt slow. Needed to guess actions to play**

**Controls were extremely delayed**

- **Standard thick client**
- **Outatime**
- **Standard thin client**

41 participants, many self-described “expert gamers”
Out of time, there is more!

- Meeting FPS deadline (e.g., 30 or 60 fps)
  - Fast speculation
  - Fast checkpoint & restore
  - Pack depth-map with frame image
  - Fast capture & encode
- Better visual experience
  - Dealing with IBR artifacts (patching smears)
  - Kalman-filtering for shake reduction
Conclusion

• Speculation masks RTT for twitch interactions

• Key techniques
  – Navigation: Predict future state
    Gracefully handle misprediction
  – Impulse: Reduce state space exploration

• Outatime can support up to 128ms RTT
  – Better game experience than standard thin-client
Questions?