CS4513 Distributed Computing Systems

Games in the Cloud

Introduction



- Growth:
  - Networks – high bandwidth to the home
  - Thin clients – Remote Desktop, Google Desktop
  - Online games
- Opportunity:
  - Heavyweight, "fat" server hosting game
  - Stream game as interactive video over network
  - Played on a lightweight, thin client
- Motivation:
  - Rendering game that requires data and specialized hardware not at client
    - Sony Remote Play, OnLive
  - Augmented reality - physical world enhanced by thin, wearable computers (e.g., head-mounted displays)
  - Ease of implementation and maintenance

References

- [CCL14] W. Cai, M. Chen, and V. Leung. "Toward Gaming as a Service", *IEEE Internet Computing*, May/June, 2014 (to appear)
- [Cla09] Mark Claypool. "Motion and Scene Complexity for Streaming Video Games", In *Proceedings of the 4th ACM International Conference on the Foundations of Digital Games (FDG)*, Florida, USA, April 2009.
- [CFGS14] Mark Claypool, David Finkel, Alexander Grant and Michael Solano. "On the Performance of OnLive Thin Client Games", *Springer Multimedia Systems Journal (MMSJ),* February 2014.

Outline

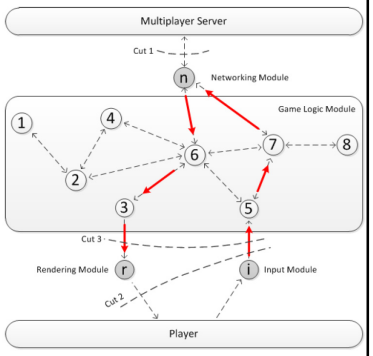
- Introduction (done)
- Games as a Service (next)
- Games as Video
- Game Video Performance

Why Games as a Service?

- Potential scalability
  - Overcome processing and storage limitations
- Cross-platform support
  - Can run games built for different platforms (e.g., Xbox and Playstation) on one device
- Piracy prevention
  - Since game code is stored in cloud, cannot be copied
- Click-to-play
  - Game can be run without installation

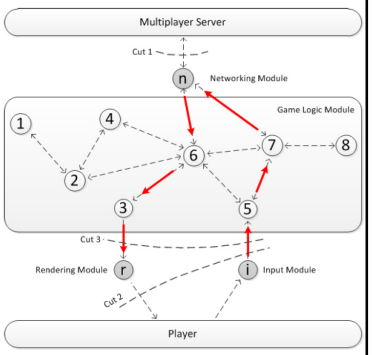
Cloud Game Modules (1 of 2)

- *Input* – receives control messages from players
- *Game logic* – manages game content
- *Networking* – exchanges data with server
- *Rendering* – renders game frames
- How do put in cloud?



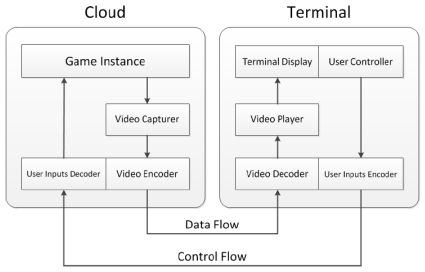
Cloud Game Modules (2 of 2)

- Cuts
  1. All game logic on player, cloud only relay information (traditional network game)
  2. Player only gets input and displays frames (remote rendering)
  3. Player gets input and renders frames (local rendering)



Remote Rendering

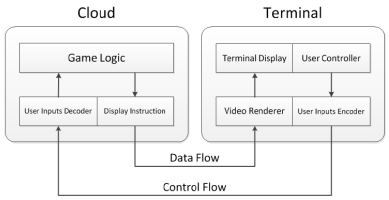
- Cloud runs full, traditional game
- Captures video ("scrape" screen) and encode
- Client only needs capability to decode and play
  - Relatively minor requirements
- Bitrate requirements can be an issue



- e.g.,
  - Onlive (commercial)
  - Gaming Anywhere (research)
  - Cloud Saucer Shoot (teaching)

Local Rendering

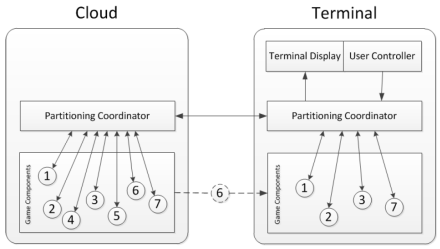
- Instead of video frames, send display instructions
  - Potentially great bitrate savings
- Challenge for instruction set: able to represent all images for all games



e.g., Browser-based games (via HTML5 and/or Javascript), [De Winter et al., NOSSDAV '06]

Potential Distribution of Computing

- Partitioning coordinator if/when to migrate functionality (e.g., reduce cloud load and/or when terminal has greater capabilities)
  - Remote and Local rendering cases (above) are really just special cases
- Challenge: how to do so in general, how to synchronize if both cloud + terminal have module (e.g., "6")



e.g., [Cai et al., CloudCom 2013]

Outline

- Introduction (done)
- Games as a Service (done)
- Games as Video (next)
- Game Video Performance

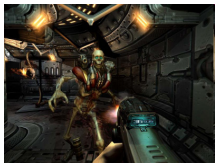
Application Streams vs. Game Streams

- Traditional thin client applications (e.g., x-term, remote login shell):
  - Relatively casual interaction
    - e.g., typing or mouse clicking
  - Infrequent display updates
    - e.g., character updates or scrolling text
- Computer games:
  - Intense interaction
    - e.g., avatar movement and shooting
  - Frequently changing displays
    - e.g., 360 degree panning

Games as Streaming Video

- High bandwidth – push limits of graphics
  - Need efficient compression
- Adapting traditional video to network → motion and scene complexity crucial to maximize quality
  - High motion needs quality scaling
  - Low motion needs temporal scaling
  - Complex scenes limit ability to quality scale
  - Getting it "right" improves perceived quality by as much as 50%
- To more effectively stream games as video, need:
  1. Standard measures of motion and scene complexity
  2. Streaming game videos as benchmarks
  3. Understanding how current thin tech is limited

Game Perspectives



First Person Linear

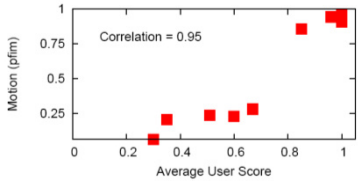
Third Person Linear

Omnipresent

Third Person Isometric

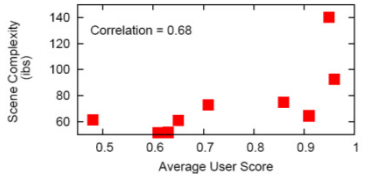
Motion

- 9 Videos varying motion/scene complexity
- Divide frame into 16 blocks
- User rated amount of motion (0, ¼, ½, ¾, 1)
- Results:
  - MPEG vector [12]: 0.51
  - PMES [9]: 0.70
  - Interpolated macroblocks [13]: 0.63
- Our measure:
  - Percentage of Forward/backward or Intracoded Macroblocks (*PFIM*) 0.95



Scene Complexity

- Same 9 Videos varying motion/scene complexity
- Divide frame into 16 blocks
- User rated complexity (0, ¼, ½, ¾, 1)
- Our measure:
  - Intracoded Block Size (*IBS*) 0.68



Select Games

Perspective	Game
First	Battlefield 1942, Battlefield 2, Battlefield Vietnam, Doom 3, Medal of Honor Allied Assault, Quake III Arena, Star Wars Battlefront
Third (Lin)	Fahrenheit, Guild Wars, Harry Potter Chamber of Secrets, The Incredibles, The Wonderful End of the World
Third (Iso)	Diablo II, Evil, Galactic Mail, Koalabr8 Lazarus, Pyramid Panic, Rainbow Reef, Wingman Sam
Omnipresent	Age of Empires 3, Age of Mythology, Battle for Middle Earth 2, Command and Conquer 3, Command and Conquer Generals, Company of Heroes, Star Wars Galactic Battlegrounds, Stronghold 2, Warcraft III

Capture Game Videos

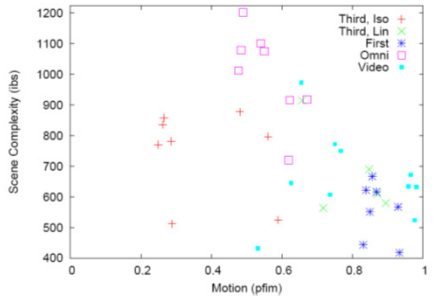
- FRAPS (Direct X or OpenGL), 30 f/s
- PC Intel P4, 4.0 GHz, 512 MB RAM, nVidia Geforce 6800GT 256
  - After: MPEG compress using Berkeley MPEG Tools
- Resolution: *800x600 pixels*
- Length: *30 seconds*

Select Videos

- Widely used by multimedia community
- Range of motion and scene complexity
- Each 10 seconds long

Video	Description
Coastguard	Panning of a moving coastguard ship
Container	A container ship sailing slowly
Foreman	A close up of a talking head
Hall	An office hallway with some people
Mobile	Panning of moving toys
News	Two news reporters talking
Paris	Two people talking with gestures
Silent	A person demonstrating sign language
Vectra	Panning of a moving car

Motion and Scene Complexity



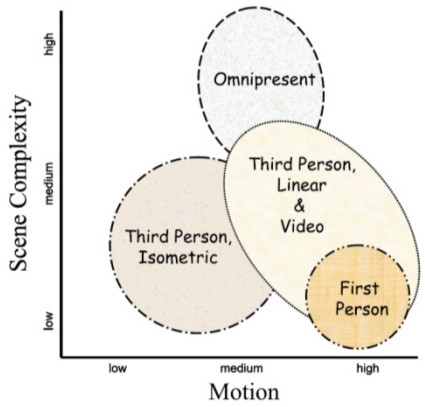
MOTION

- Games from .20 to .95
  - First highest → panning
  - Third iso → lowest (except side scroll)
  - Omin → all medium
- Videos all .70 to ~1

SCENE COMPLEXITY

- Games vary considerably across all genres
  - First least (may value responsiveness)
  - Omni most (lots of detail for game play)
  - Third medium
- Videos vary low to high but a bit less than highest omni

Motion and Scene Complexity - Summary



Outline

- Introduction (done)
- Games as a Service (done)
- Games as Video (done)
- Game Video Performance (next)

What is OnLive and Why is it Important?

- Gaming in the cloud
- Thin client, no special hw requirements
  - PC, Mac, OnLive mini-console
- Game video streamed to client
- Importance:
  - Allows playing AAA games on simple devices
  - Provide access to legacy games on next-gen consoles without hardware compatibility

Goal of Study

- How does the magic of OnLive work?
  - "black box"
- Study network traffic *turbulence* of games on OnLive
  - Packet size
  - Inter-packet time
  - Overall bitrate up and down
  - Performance during loss & latency
- Controlled variation of network parameters
- Different genres of games

Unreal Tournament III (2007)

First-person shooter

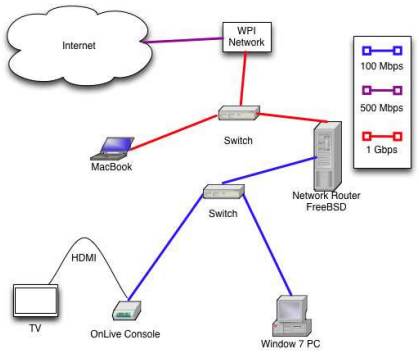
Batman: Arkham Asylum (2009)

Third-person action-adventure

Grand Ages: Rome

Real-time strategy, omnipresent (2009)

Experimental Set-Up

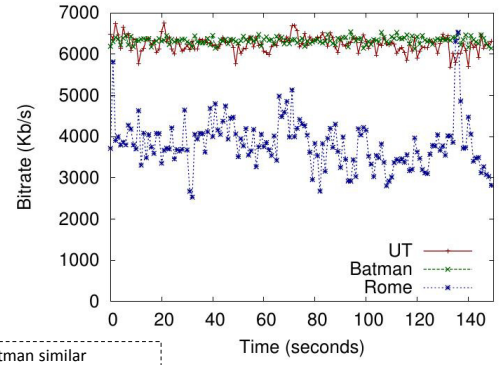


Design of Experiments

- All traffic measured UDP
- Varied capacity, loss and latency
- Parameters:
  - Capacity (down:up) 5:1 Mb/s, 10:2 Mb/s, and unrestricted
  - Latency (round-trip) 0, 40, and 70 ms
  - Loss (downstream) 0%, 1%, and 1.5%
  - Iterations: 2.5 minute game runs, 3 iterations for each experiment, following longer pilot studies

Downstream Bitrate

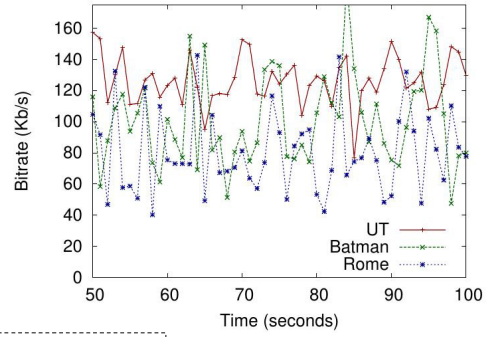
unrestricted



UT, Batman similar
Rome less dynamic display

Upstream Bitrate

unrestricted

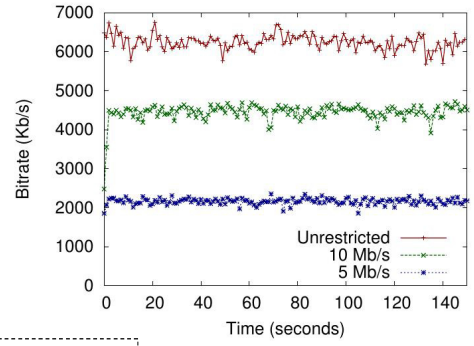


Upstream similar
Much less than downstream!

Only UT for subsequent analysis

Downstream Bitrate

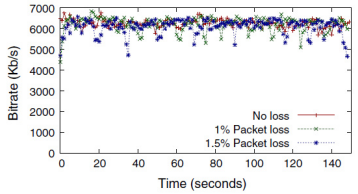
Capacity restriction

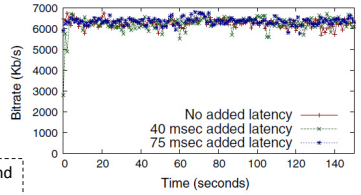


Bitrate responds to capacity

Downstream Bitrate

Loss and Latency



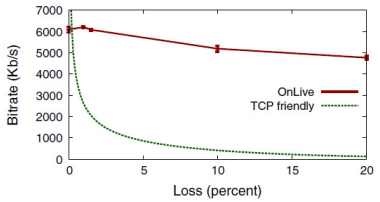


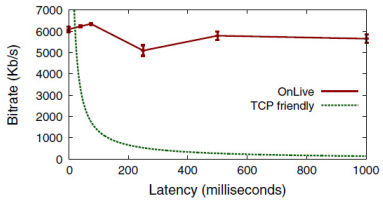
Bitrate does not respond to loss and latency

Downstream Bitrate

TCP Friendly

$$T \leq \frac{1.5\sqrt{2/3} \times s}{R \times \sqrt{p}}$$

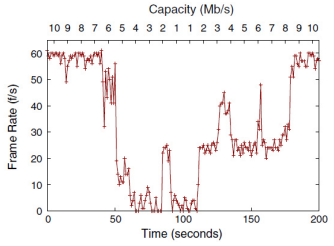


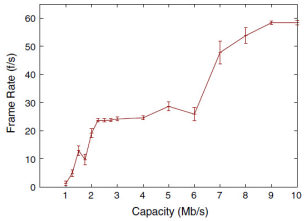


OnLive not TCP Friendly

Downstream Bitrate

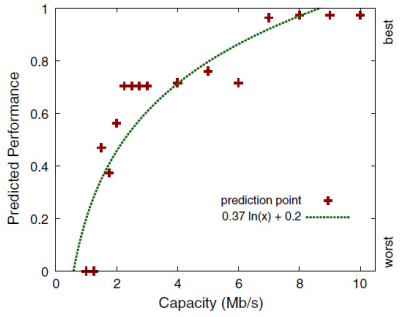
Capacity restriction





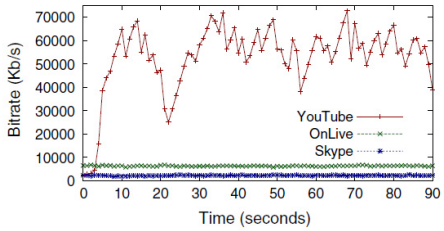
Capacity affects frame rate
(OnLive recommends 5 Mb/s, but accepts 2 Mb/s)

Predicted Player Performance



(Model based on FPS data with restricted frame rates)
Capacity affects performance

Downstream Bitrate
vs. Other Applications



Skype about ½ bitrate (but about ½ resolution)
YouTube expands to use all available (over 90 seconds)

Turbulence Summary

Application	Bitrate (kb/s)	Pkt size (bytes)	Inter-Pkt (ms)
Traditional game	67	75	45
Virtual environment	775	1,027	9
Live video	2,222	1,314	0.1
Thin Game	6,247	1,203	0.7
Pre-recorded Video	43,914	1,514	0.1

Summary

- Games as service new model for cloud computing
  - Choices on distribution of rendering and computation
- Cloud games are like video, but different
  - Wider range of motion and scene complexity
- OnLive
  - Like video conference down, traditional games up
  - Bitrate responds to capacity, but not loss or latency
    - Not TCP-Friendly
  - Best for players above 5 Mb/s, with 2 Mb/s minimum
    - Lower capacities affect player performance