ARIVU: Power-Aware Middleware for Multiplayer Mobile Games

Bhojan Anand‡, Karthik Thirugnanam†, Le Thanh Long‡, Duc-Dung Pham‡, Akhihebbal L. Ananda‡, Rajesh Krishna Balan†, and Mun Choon Chan‡

‡National University of Singapore and †Singapore Management University
Multiplayer Game
Mobile Multiplayer Game
Complex User Demand

Data rate, Computational power

QoS (user experience)
Energy efficiency

User demand is shifting
Simple to complex requirement
• Compute power, BW ... are growing faster to satisfy the Performance and QoS requirement, but BATTERY TECHNOLOGY is not in same pace!

• Gap between the DEMAND and SUPPLY is increasing!
Energy Distribution

**Major power consuming components**
- Display
- Network
- CPU
Middleware for Multiplayer Mobile Games

- Design Objectives:
  - Should be power aware
    - Power is limited resource, its use should be reduced.
  - Should be network aware
    - Wireless networks are unstable with high jitter, should tolerate it.
  - Should Scale
    - Infrastructure should scale to Massive levels
  - Should preserve quality of game play
  - Should work for most of the game types, FPS, MMOG...
Power Aware Middleware - Architecture

GAME CLIENT

POWER-AWARE MIDDLEWARE

Resource Data Collector

Set of Algorithms

Game Action Prediction Engine

Resource Controller

POWER CONSUMING RESOURCES

Wireless Interface

Processor

Display

GAME SERVER
Power Aware - Wireless Interface

• Most obvious policy - 'intermittent SLEEP" - Sleep whenever you can.

• State of the Art Schemes are based on Network Access Pattern and Application’s Intent. But, for Games:
  
  • Network Access Pattern : Continuous, every 20 - 50 ms
    » NIC has to be in RECEIVE mode whenever it is not transmitting
  
  • Game’s Intent : Tx, Rx regularly. Strict real-time requirement

- When to put SLEEP mode ??? and How long ???
One possible solution...

- SLEEP (for short periods) when game state/activity is not important
  - No loss of Important Packets
  - Missing game state can be extrapolated (DR, ...)

- Implemented both on server and client side, to get maximum possible information
  - More information => high power saving with minimum loss to quality

- Two level algorithms,
  - MACRO (Server Side) - yields longer sleep durations
  - MICRO (Server and Client Side) - yields shorter sleep durations
Algorithm - Single Ring (Relative Velocity Based)

MACRO level scanning (Server Side)

$r$ - is dynamic based on the environment

$r = r_1$ or $r_2$

$r$ is the “vision range” of the player. How far the player can see clearly with higher LOD?
Algorithm - Single Ring

MACRO level scanning

• No entities in Vision Range => Non-critical state.
• Desired state for WNIC SLEEP

• How long?
  • $s_i \Rightarrow \text{Duration}_i$
  • Smallest of ($\text{Duration}_i$)
Algorithm - Single Ring

1) For each <interactive entity>
   • Calculate EuclideanDistance\(^2\) to the entity
   • Record history of Square of Proximities

2) Select nearest ‘n’ entities

3) For each <nearest interactive entity>
   • Compute relative velocity (bi-directional) using history of Square of Proximities recorded in step 1
   • PSD = (currentProximity - Vision Range) / relative Velocity

4) ESD = Minimum of all positive PSDs
   • Constraint (minSLEEP < PSD < maxSLEEP)
Algorithm – Single Ring

• Can it scale?
  - Interaction Recency
    • RC maintains list of recently interacted entities for each client in Most Recent Interaction Table (MRIT) of size $m \times p$,
      - where $m$ is number of clients and $p$ is number of interactive entities a client is interested in.
  - Dual Ring
    • Games with high player density
Algorithm - Dual Ring (Incremental Lookahead)

Vision Range \( (r) \) - is dynamic based on the environment

\[
\text{SafeArea} \ (s) = r + (\text{global average velocity of player} \times 200\text{ms})
\]

's' grows in 200ms time steps... (upto 1sec for M^3ORPGs)

Client's tile positions are registered with the Tiles!
Micro Level Algorithms

- When there are interactive entities inside ‘vision range’
- Multiple Schemes – based on availability of data
- Server side
  - field of view of a character is around $2\pi/3$
  - Max turning speed - $2\text{rad/second}$ or $0.5\text{rad/200ms}$
Micro Level Algorithms

- **Client Side (Client Only)**
  - PAL (Player Activity Level) prediction
    - Def: No of key_press and mouse events per second
    - Correlation between PAL and Game State
    - State is critical if PAL > threshold
    - PAL_threshold is set based on the player’s expertise level
      - Can be measured using the data from game or externally as an independent tool
  - Prediction is based on weighted historical data, with high weight for most recent data
Micro Level Algorithms

- Client Side (Client Only)
  - GAPE (Game Action Prediction)

Frequency of Game Actions

Actions in Quake 3

<table>
<thead>
<tr>
<th>Actions in Quake 3</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Walk</td>
<td>27</td>
</tr>
<tr>
<td>Run</td>
<td>12</td>
</tr>
<tr>
<td>Hide</td>
<td>19</td>
</tr>
<tr>
<td>Go up and down</td>
<td>13</td>
</tr>
<tr>
<td>Jump</td>
<td>10</td>
</tr>
<tr>
<td>Shooting-booming</td>
<td>22</td>
</tr>
<tr>
<td>Change the weapon</td>
<td>2</td>
</tr>
<tr>
<td>Collect Utility</td>
<td>2</td>
</tr>
</tbody>
</table>
Micro Level Algorithms

- **Client Side (Client Only)**
  - GAP (Game Action Prediction) by GAP Engine
    - There is a correlation between the game action and game state.
    - ARIVU currently captures: Idle, Attacking, Moving, Accessing Menu, Dead, Chat, Trading, Item Interaction, Interacting with other avatars, Interaction with NPC
    - Prediction is based on past history of actions
      - \( \text{GameAction}(i+1) = w_j \times \text{GameAction}(i-j) \); for \( j = 0 \) to \( n-1 \)
      - \( w_0 > w_1 > w_2 > w_3 > w_4 > \ldots > w_{n-1} \)
      - Initial weights are \( 1/2, 1/4, 1/8, 1/16 \) and \( 1/16 \) for \( w_0 \) to \( w_4 \)
RM collects the following through API ...

- **Server Side:**
  - Game map info [size and shape]
  - Positions of entities
  - Entity interactions
  - Game environment
  - Game player expertise level
  - Game genre (To select appropriate MACRO / MICRO algorithm)

- **Client Side:**
  - Key press and mouse events (interactions)
  - Game actions of players
Building and Testing ARIVU

- Built our own Android game, an isometric Mobile Multiplayer RPG called Armageddon.
Building ARIVU

• Built in 3 different variants
  • Full
    - Uses both server and client side approaches
  • Secured
    - Uses only client side approaches (GAPE and PAL)
  • Black Box
    - External tool, independent of the GAME (only PAL)
Evaluation

- The effective “vision range” for friendly environment is 125 pixels and hostile area is 250 pixels.

- All the variants are tested with 6 human players and 3-12 bots. Interaction recency is used to boost the scalability (instead of Dual Ring).

- A packet is important for a client if, when the packet is transmitted, there is at least one interactive object within its vision range with which there is at least one interaction.
Testing ARIVU

Percentage of Power Saved (RPG game)
Testing ARIVU

Drop Rate of Important Packets (RPG game)
Testing ARIVU

Percentage of Power Saved (FPS game)
Testing ARIVU

Drop Rate of Important Packets (FPS games)
Q & A
Work Since then – Implementation in commercial games

- Implemented in ioquake3 (FPS), RyZOM (MMORPG) and evaluated by informal study and detailed simulation.

- Sources of information reduced.
  - Purely Server side – Macro and Micro algorithms

- Only Full-mode Variant (without client side data)
Implementation in Ryzom

- **MACRO**: Dual Ring Approach
  - With 200, 300, 400, 500 ..., 1000ms time steps

- **MICRO**: Viewing Angle
Results (for 40 players)

Moving Speed of the Players (set to high->low)

600 is Most conservative on Quality
Results (for 40 players)

Sleep Composition (moving speed)

Global Average Moving Speed of the Players
Implementation in Quake 3

• Area, Cluster, Area Portal and Cluster Portal
  - An area is convex hull with either visible or invisible side planes (portals).
    • Movement from area to area is possible only through portals
  - Cluster is a group of connected Areas sharing bounding faces and reach-abilities
    • Movement from cluster to cluster is possible only through portals
• Pre-computed and Stored in Binary Space Partition (BSP) tree - leaves are Areas
Implementation in Quake 3

- **MACRO** - Single Ring (Dmax), Cluster Level
- **MICRO** - Area Level

Implementation in Quake 3

Macro Algorithm
- A BFS is carried on the clusters, using the BSP tree

Implementation in Quake 3

- Micro Scanning - Pre-computes “Potentially Visible Set” and Stores in BSP tree

Line starts from portal points available in BSP
Implementation in Quake 3

Line-plane intersection problem

Area Portal sequence and sightline
Results (5 players)

variation of fixed sleep time

percentage

Movement Speed

energy
Results (5 players)

error composition

- macro
- micro
Current work - Display power

- Attempting to save power from LCD display backlight
- Implemented in ioquake3
- Backlight power wasted when a dark image is displayed
  - Can be saved if image is brightened and backlight intensity reduced
  - Must be done intelligently to maximize power saved, and minimize quality loss
Multiplayer Games

• Resource Consumption

• Characteristics – FPS Games (Quake, Call of Duty 4, Counter-Strike...)
  - Highly interactive
  - 40-80 fps
  - 20-30 pps client to server (in regular interval, cannot burst!)
  - 40-60 pps server to client (in regular interval, cannot burst!)
  - Latency more than 150ms makes the game play annoying
  - Latency more than 300ms makes the game not playable. Most likely the server will disconnect
Multiplayer Games

• Characteristics
  - If RPG/Strategy game (WoW, Linage 2)
    • Combination of high & low interactive areas
    • 40-60 fps
    • ~10 pps client to server
    • 15-40 pps server to client
    • Can tolerate latency 400 – 600 ms, depends on the
  
  In this project, we used FPS game Quake III. Results
  will be better for slow speed games like RPG games.
Complexities

• Real-time AV streaming.
  • Can pre-fetch/buffer for smooth playback and energy management
  • Strong relation between successive frames, easy to interpolate
  • Not highly interactive

• Games
  • Strict real-time constraints
  • No much use of buffering and interpolating
  • Highly interactive