ARIVU: Power-Aware Middleware for Multiplayer Mobile Games

Anand Bhojan
COM2-04-26, School of Computing
www.comp.nus.edu.sg/~bhojan
banand@comp.nus.edu.sg ph: 651-67351
References


Multiplayer Game
Mobile Multiplayer Game
Complex User Demand

Data rate, Computational power

User demand is shifting
Simple to complex requirement

QoS (user experience)
Energy efficiency
Need for Power Management

Demand and Supply

“Improvements in battery technology, while steady, no longer happen at the breakneck speed of younger technology like smartphones,” says Keith Nowak of phone and tablet maker HTC.
Mobile game is one of the most rapidly growing areas in today’s consumer technology. Games alone account for more than 50% of current iPhone application downloads.

### Need for Power Management

#### Usability Perspective

<table>
<thead>
<tr>
<th>Platform (Phone)</th>
<th>Type</th>
<th>Battery Type</th>
<th>GSM talk, standby</th>
<th>WCDMA talk, standby</th>
<th>Quake III play</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symbian (Nokia X7 [52])</td>
<td>OIA</td>
<td>1200 mAh Li-Ion (BL-5K)</td>
<td>6h 30min, 450h</td>
<td>4h 30min, 450 h</td>
<td>-</td>
</tr>
<tr>
<td>Apple iOS (iPhone 4 [7])</td>
<td>VIA</td>
<td>1420 mAh Li-Po</td>
<td>14h, 300h</td>
<td>7h, 300h</td>
<td>-</td>
</tr>
<tr>
<td>Android (Google Nexus S [33])</td>
<td>OIA</td>
<td>1500 mAh Li-Ion</td>
<td>14h, 713h</td>
<td>6h 42 min, 427h</td>
<td>-</td>
</tr>
<tr>
<td>Android (HTC Desire HD [37])</td>
<td>OIA</td>
<td>1230 mAh Li-Ion</td>
<td>8h 10 min, 420h</td>
<td>5h 20 min, 490h</td>
<td>*1h 50min</td>
</tr>
</tbody>
</table>

* - To estimate this, we played Quake III Arena on a HTC Desire HD smartphone, for 5 minutes, and determined the percentage of battery power drained.
Need for Power Management
Environmental and Financial Perspective

- Mobile devices - Growing as one of the major contributor for **ICT energy consumption**.
- Mobile phone subscriptions worldwide has surpassed 5 billion
- Annual electricity consumption for a mobile phone 11 KWh per year.
- Mobile devices: **12% of total ICT energy**, which is (55 million MWh / 452.3 million MWh per year)
- Contribute to Green House Gases and Energy Budget
Energy Distribution

Measured on HTC Magic while streaming a Youtube Video
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 SERVER

POWER AWARE MIDDLEWARE

Resource Data Collector (RDC)

- Power Management API
- World Discretisation and Initialisation Engine (WDIE)
- Game State Estimation Engine (GSEE)

Resource Controller (RC)

POWER CONSUMING RESOURCES

- Wireless Interface
- Display
- CPU
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 - 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 game state
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
Power Savings On Network Interface - Basic Technique

- Put Network Interface to sleep
  - Tx = Rx (~ 248 mA); Sleep ~ 4.66 mA
- Current Schemes - Network Access Pattern and Application’s Intent.
- Games – Strict Real-time Requirements
  - When to put SLEEP mode? and How long?

![Diagram showing power consumption between active transmission/reception (Tx/Rx) and sleep modes.](chart.png)
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
Power Aware Middleware - Architecture

GAME SERVER

POWER AWARE MIDDLEWARE

GAME CLIENT

Resource Data Collector (RDC)

Resource Controller (RC)

Power Management API

World Discretisation and Initialisation Engine (WDIE)

Game State Estimation Engine (GSEE)

POWER CONSUMING RESOURCES

Wireless Interface

Display

CPU
World Discretisation and Initialization

- **Tile Based** (mostly 2D)
- **PVS based** (Area, Cluster, Portal) (3D)
Power Aware Middleware - Architecture

GAME SERVER

POWER AWARE MIDDLEWARE

- Power Management API
- World Discretisation and Initialisation Engine (WDIE)
- Game State Estimation Engine (GSEE)

Resource Collector (RDC)

RESOURCE CONTROLLERS

Resource Controller (RC)

POWER CONSUMING RESOURCES

- Wireless Interface
- Display
- CPU
Game State Estimation Engine

Various Algorithms (Requires Game Genre Based Selection)

- Macro Level (Server)
  - Distance Based
    - Single Ring
    - Dual Ring
  - Renderer’s View Based - Path Distance (Area/Cluster/Portal) [Not Covered]

- Visibility Based
  - Cell Based Visibility (2D)

- Micro Level (Server)
  - PAL
  - Game Action Prediction
  - Renderer’s View Based – PVS (Area/Cluster/Portal) [Not Covered]
Algorithm – Single Ring

Relative Velocity Based

- MACRO level scanning (Server Side)

\[ r \] - is dynamic based on the environment

\[ r = r_1 \text{ or } r_2 \]

\[ r \] 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?
  - $s_i$ => $\text{Duration}_i$
  - 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
   - If PSD = negative then exit

4) **ESD** = Minimum of all 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

Safe Area ($s$) = $r +$ (global average velocity of player $x$ 200ms).

‘s’ grows in 200ms time steps... (upto 1sec for M³ORPGs)

Client’s tile positions are registered with the Tiles!
Visibility Based Game State

- A state is important only if an opponent can see the player or vice versa.

- **Simplest Solution** - If no opponent is visible, safe to sleep network card for some time.
Visibility Based Game State

- We need to make sure no opponent can reach the player while player NIC is sleeping.

- Advanced Solution – Predict ahead all possible movements and calculate max safe sleep time.
Visibility Based Game State

- Each player continues traveling in the direction he is traveling now, in the near future (100s of ms)

- **Further Improvement** – Use direction based weights and error tolerance factor (\( \alpha \)) for additional low risk sleep
Cell Based Visibility

- Game area is discretised into 2D grid
- Grid element size is related to MAX distance traveled in a set interval of time.
- Game provides function to check visibility between two points
- Used to pre-compute visibility between grid elements
Cell Based Visibility

Look Ahead

Map Divided into 2d grid

Obstacles

1

2

Not Visible

Visible

Not Visible

3
Cell Based Visibility

Look Ahead

Position of player

1
2
3

Not Visible
Visible
Not Visible
Cell Based Visibility

**Look Ahead**

Possible position of the player in $\Delta t$, say 200ms
Cell Based Visibility

Look Ahead

Players not visible to each other, but potentially visible in $\Delta t$
Cell Based Visibility

Look Ahead

Player 1 not visible to others in $\Delta t$, safe to sleep for $\Delta t$
Cell Based Visibility

Normalized Sum of Costs

- Consider 2 players at grids x1 and x2
- In t time steps, let the grids reachable for players 1 and 2 be $L_t^1$ and $L_t^2$ respectively.
- For each point $l_1 \in L_t^1$ check its visibility to each point in $L_t^2$
  $$v(l_1, l_2) = 1 \text{ or } 0$$

$$S_t(1, 2) = \frac{\sum_{l_1 \in L_t^1, l_2 \in L_t^2} w_{l_1} * w_{l_2} * v(l_1, l_2)}{\sum_{l_1 \in L_t^1, l_2 \in L_t^2} w_{l_1} * w_{l_2}}$$

Based on Pre-Computed Visibility Matrix
Cell Based Visibility

Direction based Weights

- Different future grid positions assigned weights based on direction
- Calculate normalized sum of product of weights for each position that is visible, for 2 players
- If sum < error tolerance (\(\alpha\)), safe to sleep
Direction based Weights

Different Sleep Times

- From a grid calculated to $\Delta t$ intervals, algorithm can easily be extended to $2\times\Delta t, 3\times\Delta t$.

- Previous steps first run on $3\times\Delta t$, down to $\Delta t$ to calculate safe sleep time.

<table>
<thead>
<tr>
<th>Potential player position in $\Delta t$</th>
<th>$\Delta t$</th>
<th>$2\times\Delta t$</th>
<th>$3\times\Delta t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(-3, 3)$</td>
<td>$(-3, 3)$</td>
<td>$(-3, 2)$</td>
<td>$(-3, 1)$</td>
</tr>
<tr>
<td>$(-3, 2)$</td>
<td>$(-3, 2)$</td>
<td>$(-3, 1)$</td>
<td>$(-3, 0)$</td>
</tr>
<tr>
<td>$(-3, 1)$</td>
<td>$(-3, 1)$</td>
<td>$(-3, 0)$</td>
<td>$(-3, -1)$</td>
</tr>
<tr>
<td>$(-3, 0)$</td>
<td>$(-3, 0)$</td>
<td>$(-3, -1)$</td>
<td>$(-3, -2)$</td>
</tr>
<tr>
<td>$(-3, -1)$</td>
<td>$(-3, -1)$</td>
<td>$(-3, -2)$</td>
<td>$(-3, -3)$</td>
</tr>
<tr>
<td>$(-3, -2)$</td>
<td>$(-3, -2)$</td>
<td>$(-3, -3)$</td>
<td></td>
</tr>
<tr>
<td>$(-3, -3)$</td>
<td>$(-3, -3)$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Cell Based Visibility
Determining Sleep Duration

For each player \( i \)

For each time slot \( t \in T \)

For each player \( j \in N \setminus i \)

compute \( S_t(i, j) \)

\( t \) is feasible if \( S_t(i, j) \leq \alpha, \forall j \)

Select the largest feasible \( t \) as user \( i \)'s sleeping interval,
otherwise user \( i \) does not sleep.

\( t \) – is potential sleep duration (PSD)

Alpha varies form 0 – 1.
0 – conservative
1 – aggressive (always sleep)
Intro to Renderer’s View/PVS Based Basics

- **3D spatial subdivision** is used for collision detection, rendering...
  - The world is split into small convex hulls (areas). From the convex hull, there is in fact a limited number of other convex hulls can be seen.

- **Potentially Visible Sets (PVS)** is a set of potentially visible areas form current area. It is map dependent and pre-computed.
  - For occlusion culling, used by renderers, pre-computed, then indexed at run-time in order to quickly obtain an estimate of the visible geometry.
  - It is symmetric

- We leverage on this in-game spatial subdivision & PVS (lower overhead costs)
Intro to Renderer’s View/PVS Based

Background: Cluster, Area and Portal

- Adjacent **areas** shared same surfaces are grouped together to form a **cluster**.
  - Eg. Room is a Cluster with areas for walking, area under table…

- The door connecting two clusters is a **cluster portal**.
  - Some modern games allow the artist to specify cluster portals.
Intro to Renderer’s View/PVS Based

PVS in one Cluster

If it is translucent object, vision can go beyond

Can be fully computed

<or>

Artist can specify Portals to simplify
Intro to Renderer’s View/PVS Based Renderer’s View of the World (PVS)

- **Safe Period:** When no other players in the current player's PVS set.
Micro Level Algorithms

Server Side

- 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 – $2 \times \frac{\pi}{3}$
  - Max turning speed – $2 \text{ rad/second}$ or $0.5 \text{ rad/200ms}$
Micro Level Algorithms

Client Side (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)
(Game Action Prediction)

Frequency of Game Actions

Actions in Quake 3
Walk, Run, Hide, Go up and down, Jump, Shooting-booming, Change the weapon, Collect Utility

Percentage
0, 5, 10, 15, 20, 25, 30
Micro Level Algorithms

Client Side (Client Only)

- GAP (Game Action Prediction) 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
  - $\text{GameAction}(i+1) = w_j \ast \text{GameAction}(i-j); \text{ for } j = 0 \text{ to } n-1$
  - $w_0 > w_1 > w_2 > w_3 > w_4 > \ldots > w_{n-1}$
  - Initial weights are $\frac{1}{2}$, $\frac{1}{4}$, $\frac{1}{8}$, $\frac{1}{16}$ and $\frac{1}{16}$ for $w_0$ to $w_4$
Power Aware Middleware - Architecture

- **GAME SERVER**
- **GAME CLIENT**

**POWER AWARE MIDDLEWARE**
- Power Management API
- World Discretisation and Initialisation Engine (WDIE)
- Game State Estimation Engine (GSEE)

**Resource Controller (RC)**

**POWER CONSUMING RESOURCES**
- Wireless Interface
- Display
- CPU
RDC 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
Implementation & Results

Quake III
FPS

Armageddon
RPG

Ryzom
MMORPG
Armageddon (RPG)
ARIVU on Armageddon

Built our own Android game, an isometric Mobile Multiplayer RPG called Armageddon. (FPS mode can be simulated)

Algo: Single Ring (Macro), View angle (Micro).

<table>
<thead>
<tr>
<th>Metric</th>
<th>MW Enabled</th>
<th>MW Disabled</th>
<th>Overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current Drawn by CPU(mA)</td>
<td>213.39</td>
<td>207.29</td>
<td>6.1</td>
</tr>
<tr>
<td>Memory Used(KB)</td>
<td>2294.6</td>
<td>2293.1</td>
<td>1.5</td>
</tr>
</tbody>
</table>
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.
Results (RPG)

Energy saving

- 2 players
- 4 players

Estimated Avg Player Velocity

0% 5% 10% 15% 20% 25% 30% 35% 40%

500 300 100
Results (RPG)

Error rate

Estimated Avg Player Velocity

- 2 players
- 4 players
Results (RPG)

Energy Composition (4 players)

Contribution (%)

- 100%
- 90%
- 80%
- 70%
- 60%
- 50%
- 40%
- 30%
- 20%
- 10%
- 0%

Estimated-Avg-Player-Velocity

- 500
- 300
- 100

Micro

Macro
Results (Simulated FPS with More Players)

![Graph showing the drop rate of important packets with increasing number of players.](image)

- EAPV: 300

The graph depicts the drop rate of important packets in FPS games as the number of players increases. The energy saved is shown on the y-axis, and the number of players is shown on the x-axis. As the number of players increases, the energy saved decreases linearly, indicating a decrease in performance or reliability of the game.
Results (Simulated FPS) with more players

- Graph showing the relationship between the number of players and error rate.
- EAPV: 300

Axes:
- Y-axis: Error rate (0% to 7%)
- X-axis: Number of Players (0 to 10)
Ryzom (MMORPG - OpenSource)
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)

Micro
Macro
Quake III (FPS)
Implementation in Quake III

- Performance depends on map size and number of players

- On an average map, we are able to save up to 25% of network power with little noticeable impact to the game

- User study conducted to study impact

- Any artifacts only manifest when the player first comes into vision
Implementation in Quake III

- Modes (Algorithm Variations):
  - Static 200
  - Static 400
  - Static 600
  - Dynamic

<table>
<thead>
<tr>
<th></th>
<th>(-3,3)</th>
<th>(-2,3)</th>
<th>(-1,3)</th>
<th>(0,3)</th>
<th>(1,3)</th>
<th>(2,3)</th>
<th>(3,3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(-3,2)</td>
<td>(-2,2)</td>
<td>(-1,2)</td>
<td>(0,2)</td>
<td>(1,2)</td>
<td>(2,2)</td>
<td>(3,2)</td>
<td></td>
</tr>
<tr>
<td>(-3,1)</td>
<td>(-2,1)</td>
<td>(-1,1)</td>
<td>(0,1)</td>
<td>(1,1)</td>
<td>(2,1)</td>
<td>(3,1)</td>
<td></td>
</tr>
<tr>
<td>(-3,0)</td>
<td>(-2,0)</td>
<td>(-1,0)</td>
<td>(0,0)</td>
<td>(1,0)</td>
<td>(2,0)</td>
<td>(3,0)</td>
<td></td>
</tr>
<tr>
<td>(-3,-1)</td>
<td>(-2,-1)</td>
<td>(-1,-1)</td>
<td>(0,-1)</td>
<td>(1,-1)</td>
<td>(2,-1)</td>
<td>(3,-1)</td>
<td></td>
</tr>
<tr>
<td>(-3,-2)</td>
<td>(-2,-2)</td>
<td>(-1,-2)</td>
<td>(0,-2)</td>
<td>(1,-2)</td>
<td>(2,-2)</td>
<td>(3,-2)</td>
<td></td>
</tr>
<tr>
<td>(-3,-3)</td>
<td>(-2,-3)</td>
<td>(-1,-3)</td>
<td>(0,-3)</td>
<td>(1,-3)</td>
<td>(2,-3)</td>
<td>(3,-3)</td>
<td></td>
</tr>
</tbody>
</table>
Baseline Results

![Graph showing Power Saved (%) vs. Static Sleep Time (ms) for different number of players.]

- **Power Saved (%)**
  - 600 ms: 8 Players, 4 Players, 2 Players
  - 400 ms: 8 Players, 4 Players, 2 Players
  - 200 ms: 8 Players, 4 Players, 2 Players

- **Accuracy (%)**
  - 600 ms: 8 Players, 4 Players, 2 Players
  - 400 ms: 8 Players, 4 Players, 2 Players
  - 200 ms: 8 Players, 4 Players, 2 Players
Results over 3G

Average Accuracy (%) vs. Average Power Saved (%)

- Dynamic
- Static 200
- Static 400
- Static 600
Results over Wifi

![Graph showing average accuracy vs average power saved over WiFi for different static values. The graph compares dynamic and static settings, with lines for Static 200, Static 400, and Static 600. The y-axis represents average accuracy in percentages, and the x-axis represents average power saved in percentages.]
Effect Of Density & Dynamic Algorithm

Average Power Saved (%)

Relative Player Density

- Alpha = 0.2
- Alpha = 0.15
- Alpha = 0.1
- Alpha = 0.05
- Alpha = 0
Effect Of Density & Dynamic Algorithm

Average Power Saved (%) vs Relative Player Density

- Dynamic
- Static 200
- Static 400
- Static 600
Results - Visibility

Effect of Different sleep intervals

![Graph showing the effect of different sleep intervals on visibility. The graph plots Average Accuracy (%) against Average Power Saved (%). The x-axis ranges from 14 to 32, and the y-axis ranges from 95 to 60.]
Actual Measurements

![Graph showing actual measurements and simulations of average accuracy vs. average power saving.](image-url)
User Study Results - Game Quality Impact

Unnoticeable
Barely Noticeable
Satisfactory
Noticeable
Very Noticeable

Alpha
Intro to Renderer’s View/PVS Based Implementation in Quake III

- **MACRO** - Single Ring (Dmax), Cluster Level, Path Distance
- **MICRO** - Area Level Visibility
Intro to Renderer’s View/PVS Based Implementation in Quake III

Macro Algorithm

- A BFS is carried on the clusters, using the BSP tree
Intro to Renderer’s View/PVS Based Implementation in Quake III

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

THANKS FOR YOUR ATTENTION

Questions?