Konva: Power and Network Aware Framework and Protocols for Multiplayer Mobile Games

Bhojan Anand
National University of Singapore

ABSTRACT
Multiplayer mobile games are an increasingly important class of mobile application. While device features and application quality are rapidly growing, the battery technologies are not growing at the same pace. Battery lifetime is one of the key factors that hinder the usability of the mobile devices for resource-intensive applications. In this work, we design a framework for power management that adapts its behavior to the intent of the user and the game, the characteristics of the network and the network interface. Our system is being designed to reduce the overall device power usage without sacrificing the end-user game experience.

Categories and Subject Descriptors
C.2.1 [Network Architecture and Design]: Wireless Communication; C.2.2 [Network Protocols]: Applications; C.2.4 [Distributed Systems]: Client/Server; K.8.0 [General]: Games.

General Terms
Algorithms, Management, Measurement, Performance, Design, Security, Human Factors.

Keywords
Mobile games, wireless networks, power management, statistical prediction, transport protocol, gamelet.

1. SYSTEM ARCHITECTURE
Our system's architecture is depicted the Figure 1. We envision a three-tier architecture comprising of wireless game clients (cell phones), game servers (highly provisioned back-end servers), and access point proxies (used to isolate the effect of poor wireless latencies from the game player).

The consistency manager is used to maintain game server state between multiple game servers and the proxies. The network manger is used between the wireless clients and the proxies to provide the most optimized wireless connectivity for the required energy profile (proxy might choose to switch to higher latency lower power Bluetooth over 802.11g for a specific client for example). Finally, the resource manager is responsible for monitoring the current resource conditions and for deciding on the appropriate energy conservation techniques that achieve the best savings without impacting the end user experience. The resource manager will use different inputs and algorithms on the three different components. For example, the client resource manager will obtain inputs directly from the mobile phone's battery and use CPU and network throttling to achieve power savings while the server's resource manager will collaborate with the proxy to reduce the network bandwidth to resource constrained clients.

HotMobile 2010, February 22–23, 2010, Annapolis, Maryland,USA.

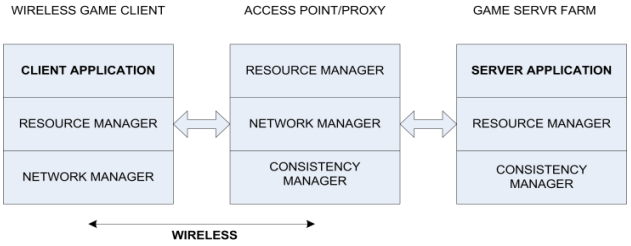

Figure 1 Top-Level System Architecture

2. CLIENT POWER MANAGEMENT
The client's resource manager, shown in Figure 2, collects and maintains data about the hardware status (*WNIC mode*, *Battery Level*, *CPU frequency*) and the client-server connectivity (*Latency*, *Estimated bandwidth*, *Connectivity*).

The resource manager computes a State Index for each game frame *'i'* using a combination of Action Data (what the player is doing), Interest Data (what the player is interacting with and his environment), Network Status and Power Status. This Index is used to determine the appropriate power conservation technique to use that best matches the current power and latency requirements. In particular, the resource manager can change the CPU frequency, display intensity, the network traffic sending rate, the type of connection (Bluetooth, WiFi, reliable, etc.), and the network interface power mode (sleep, etc.).

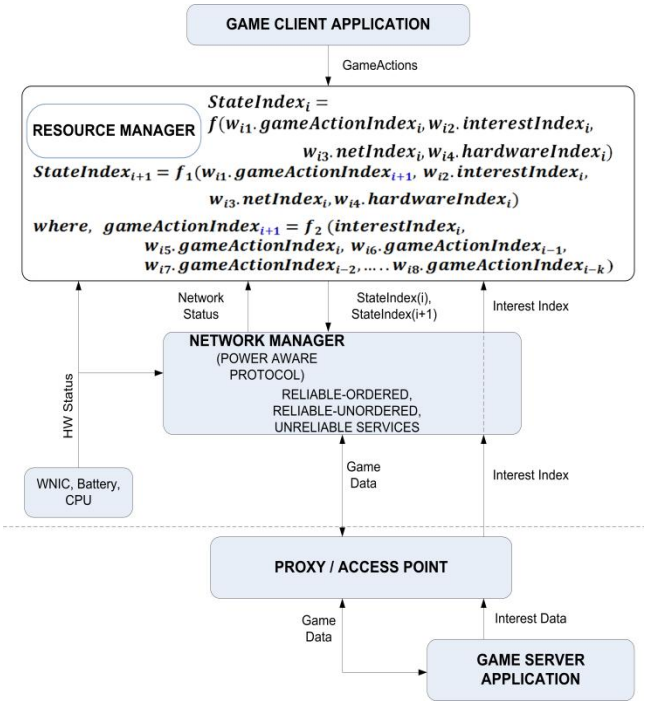

Figure 2 Information Flow for Resource Management

We use input from the game (Action Data and Interest Data) to ensure that our optimizations do not impact the end-user game experience. For example, during highly interactive game moments, we do not trigger more aggressive power saving modes. Our previous works [1] shows that, by learning the game actions (Action Data) alone we can save significant amount of power by preserving quality.

2.1 LEARNING GAME ACTIONS

The client's resource manager uses the current game state to trigger specific actions. We obtain these states by augmenting the game API as this is easier and more accurate than sniffing the game packets indirectly. We have developed this API extension to be easy to add to existing game engines. For example, for RPG games the following functions are defined to learn about the overall game state.

Action Data @ Game Client: *setPlayerAction(int Action); setFrameValidityThreshold(int fvThreshold);*

Interest Data @ Game Server: *setPlayerLocation(Boolean Hostility, int Type); setProximity(int Number, int Distance); setInteraction(Boolean Intract);setProximityInAngle(int Number, int Distance); setViewField(int Type); setExtrapolationThreshold(int drThreshold);*

We define a set of common action for each genre of game after studying several games in the same genre. For RPG games the following actions are defined: *Action.IDLE, Action.ATTACK, Action.MOVING, Action.MENU_ACCESS, Action.DEAD, Action.CHAT, Action.TRADING* and *Action.ITEM_INTERACTION.*

2.2 LEARNING INTEREST DATA

The game server will compute the Interest Data for each client connected and sends a single Interest Index value to the client's resource manager for making power management decisions. The Interest Data [2] is computed based on the following parameters: *Proximity to other players and AI characters; Location of the player; Player's interaction with other players and environmental objects; Player's viewing angle and view field.*

These parameter values can be directly sent to the client but, it will result in security/cheating risk. Encrypting these data will create additional computational overhead to the client which in turn will defeat the purpose (saving power). Furthermore, evaluating these parameters and analyzing the environment at the server side will reduce the computations required at client side.

2.3 POWER AWARE GAME TRANSPORT PROTOCOL (PAGTP)

Online games require multiple streams of different types in single association (using one socket pair) to each client: reliable ordered streams, reliable unordered streams, partially reliable streams and unreliable streams. After initial evaluations of current transmission protocols – TCP [5], UDP [4], DCCP based [8] and SCTP[6], SCTP and its variation PR-SCTP [7] looks more suitable for online games. Our PAGTP acquires knowledge about the Game State and Hardware State from the Resource Manager and manages the transport queues according to the current power saving mode. PAGTP can be implemented on top of SCTP. However, for the following reasons we have implemented it as a separate transport protocol suitable for resource constrained mobile environments.

- SCTP is basically a connection oriented protocol and offers unreliable service with unnecessary additional overheads. For eg., a typical packet which contains data and acknowledgement, takes at least 44 bytes overhead for SCTP headers. Since most game packets are less than 32 bytes, more than half of the packet contains non-data and this makes SCTP very inefficient. Most of the traffic a game generates is for unreliable delivery and only a few for reliable delivery. SCTP is only suitable for traffic which needs mostly reliable delivery with few packets for unreliable delivery.
- SCTP do not support intermittent connection failures which are common in mobile environments. Custom modifications can be made but the potential amount of change and effort may warrant it impractical.
- If PR-SCTP cannot send a packet before its lifetime expires, it is simply dropped. This is required behavior for online games. However, if the packet has been sent but not yet acknowledged, it will still be re-sent even if the retry time exceeds the lifetime. This is unnecessary for time-sensitive packets.
- For priority processing, pSCTP assigns each stream a priority and SCTP sends Heartbeat chunks to periodically probe an idle stream. Since high priority game packets are usually sparse and infrequent, this introduces needless network traffic. Our PAGTP uses game state aware priority processing hence it improves quality of the game or player satisfaction index.
- Games need multihoming for automatic transport layer level redundancy and load sharing. SCTP's multihoming is only for redundancy.

PAGTP also supports split connection: client-proxy and proxy-server. The proxy is partially aware of the game state. It gets the Interest Data and State Index to know the game state and optimize the traffic for wireless clients for various power saving profiles.

3. CONCLUSION & EXTENSIONS

Design of the game action learning technique, the resource manager algorithms and PAGTP are our primary interest for discussion in consortium. We also present about, 1. additional architectural components and, 2. making our proxy extensively game state aware by running 'gamelets' (part of the game server in a distributed fashion) in proxy to optimize traffic and hide latency for wireless clients.

4. REFERENCES

[1] Anand, B., Ananda, A. L., Chan, M. C., Long, L. T., and Balan, R. K., "Game action based power management for multiplayer online game", *MobiHeld'09, ACM SIGCOMM workshop.*

[2] Ashwin Bharambe et.al., "Donnybrook: Enabling Large-Scale, High-Speed, Peer-to-Peer Games", *SIGCOMM'08.*

[3] Chen, K.T., Huang, C.Y., Huang, P., and Lei, C.L., "An Empirical Evaluation of TCP Performance in Online Games", June 2006.

[4] RFC 768, "User Datagram Protocol", August 1980.

[5] RFC 793, "Transmission Control Protocol", September 1981.

[6] RFC 2960, "Stream Control Transmission Protocol", October 2000.

[7] RFC 3758, "Stream Control Transmission Protocol, Partial Reliability Extension", May 2004.

[8] RFC 4340, "Datagram Congestion Control Protocol", 2006