3G-03

クライアントサーバ型実アプリケーションにおける高速動作観察環境

福田翔貴† 栗原駿† 濱中真太郎† 小口正人‡ 山口実靖†

工学院大学 工学研究科 電気・電子工学専攻† お茶の水女子大学 理学部 情報科学科‡

1 はじめに

Android OS はスマートフォンやタブレット向け OS として高いシェアを誇り，同プラットフォーム向けに製作されるアプリケーションも増加傾向にある．これに伴いアプリケーションの動作観察は重要な事項の一つとなっているが，実際にアプリケーションを動作させる動的動作解析には時間がかかることが問題となっている．これを解決するために，我々は既存研究[1]にてアプリケーションシステムが認識する時間の速度を加速することにより動作観察に要する時間を短縮可能であることを示した．本稿では，クライアントサーバ型かつ実際に配布されているアプリケーションを用いて本環境の評価を行う．

2 システム内時間の加速

動的解析によるアプリケーションの動作観察時間の短縮を実現する手法として，過去に我々は端末内時間を加速する手法[1]を提案した．Android OS や Linux OS では Linux カーネルを採用しており，この Linux カーネルの時刻管理実装を書き換えることにより，システム内のプロセスが認識する時間の流れの速度を高めることができる．

Linux カーネルにおける時刻の更新は，カーネルファイルの time/timekeeping.c 内の `timekeeping_get_ns` 関数にて行われており，同関数内では `cycle_t` 型の変数 `cycle_now` および `cycle_delta` が宣言されている．`cycle_now` には取得したクロックソースが格納され，クロックソースの増分の計算結果が `cycle_delta` に格納されている．クロックソースの増分はシステム内時間に影響するため，本手法においては，このクロックソースの増分を制御することにより，端末内時間の加速を実現している．

我々は当該文献にて，端末内時計を加速する本手法を Android 端末および Linux サーバ端末に適用し評価を行った．その結果，評価用のスタンドアローン型アプリケーションおよびサーバクライアント型アプリケーションにおいて，前述の手法により，端末内時間加速環境においても通常状態とほぼ同様の結果を得ることができることを示した．ただし，当該研究においては，実際のユーザが入手可能なアプリケーションとしては簡易なメールアプリケーションのみの評価にとどまっており，動作もメールを人工的に送信する形式となっている．よって，より多様かつ現実的なアプリケーションを用いた評価が必要であると考えられる．

3 評価

前章の課題を受けて，本稿では評価対象を一般に公開され一般に入手可能なゲームアプリケーションとし，ゲームサーバ環境をユーザが構築可能なものを選定し評価を行う．

観察対象は既存研究[2]にて重要性が示された Alarm セット，WakeLock 実行に加えて通信を計測するため UDP パケットを観察した．計測はクライアントをサーバに接続した上でゲームプレイ画面にし，無操作状態にて端末内時間で 15 分

Accelerated Monitoring Environment for Practical Client-Server Application

†Shoki Fukuda, Shun Kurihara, Shintaro Hamanaka, Saneyasu Yamaguchi, Electrical Engineering and Electronics, Kogakuin University

‡Masato Oguti, Department of Information Sciences, Ochanomizu University

間行った．また，端末内時間加速状態における加速倍率は 2 倍とし，通常状態（1 倍速）と観察結果を比較した．

3.1 測定環境

測定環境を表 1 および表 2 に示す．

表 1　測定環境（Android）

Device Name	Nexus7 (2013)
OS	Android 5.1.1 (aosp) with modified linux kernel 3.4.0

表 2　測定環境（Linux）

Device Name	ASUS X200MA-B
OS	CentOS 7.2.1511 with modified linux kernel 3.19.0

3.2 評価結果

評価結果を図 1 および図 2 に示す．図 1 では Alarm セット，WakeLock 実行，UDP パケットの数を通常状態と加速状態で比較している．図より Alarm セット，WakeLock 実行回数が UDP パケット数よりも大幅に少ないことが分かるが，これは評価対象アプリケーションが Alarm および WakeLock を使用しなかったためである．一方で UDP 通信は非常に頻繁に行っていることが確認できる．この UDP 通信間隔をヒストグラムで表し，通常状態と加速状態で比較したものが図 2 である．図 1 および図 2 より，UDP 通信において通常状態と加速状態を比較してみるとわずかながら誤差は見受けられるものの，ほぼ同様の通信回数や通信間隔を観察できており，本手法は有効に動作していることが分かる．

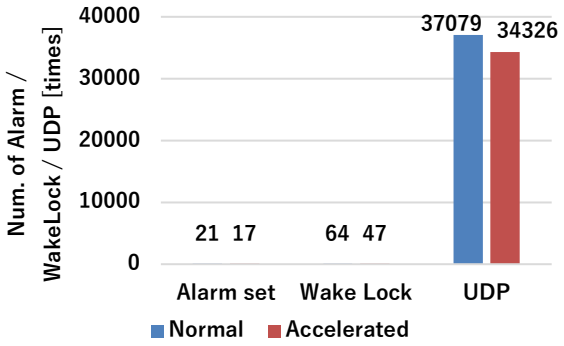

図 1　Alarm，WakeLock，UDP 数の比較

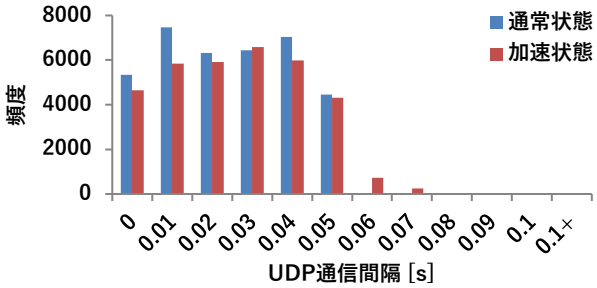

図 2　UDP 通信間隔の比較

4　おわりに

本稿では，Android OS 向けアプリケーションの動作観察が重要である反面，アプリケーション動作観察に長い時間を要する問題があることを紹介した．この問題に対して Android OS と Linux OS の双方の端末内時間を加速することによりアプリケーション観察時間を短縮出来る手法があり，本稿ではこの手法を一般に入手可能なゲームアプリケーションに適用して評価を行ったところ，UDP パケットの観察において加速環境においても通常環境と同様の結果を得ることができ，本手法が有効に機能することが確認された．

今後はより多くのアプリケーションを用いての評価や本手法の適用可能範囲の調査などを行っていく予定である．

謝辞

本研究は JSPS 科研費 25280022，26730040，15H02696 の助成を受けたものである．

本研究は，JST，CREST の支援を受けたものである．

参考文献

[1] Shoki Fukuda, Shun Kurihara, Shintaro Hamanaka, Masato Oguchi and Saneyasu Yamaguchi. “Accelerated Test for Applications with Client Application and Server Software”, 2017 The International Conference on Ubiquitous Information Management and Communication (IMCOM2017), 2017

[2] 栗原 駿, 福田翔貴, 小柳文乃, 小口正人, 山口実靖 “Alarm の観察による無操作状態携帯端末の消費電力の増加の原因となるアプリケーションの推定”, 信学技報, vol. 115, no. 230, DE2015-23, pp. 17-22, 2015 年 9 月．