Vol.2014-DPS-159 No.24
Vol.2014-MBL-71 No.24
2014/5/16

iOS に対応した電力消費量測定用 Web ブラウザの開発

西村顕[†1]　杉田薫[†2]

コミュニケーションサービス全体の電力消費量の削減を目標として，従来研究においてオンラインビデオ再生時における電力消費量測定用 iOS アプリケーションを開発と評価を行った．しかし，オンラインビデオに Apple 提供の HTTP Live Streaming を使用したため，ビデオ配信しか行うことができず，コミュニケーションサービスへの適用が困難であった．そこで，マイクロブログやソーシャルネットワークサービスのようなコミュニケーションサービスで利用される Web サービスの電力消費量が測定可能な iOS 対応の Web ブラウザを開発したので報告する．

Development of A Web browser to mecasure power consumption on iOS

KEN NISHIMURA[†1]　KAORU SHUGITA[†2]

In order to decrease total power consumption for a communication service, we have developed and evaluated a Web browser to support some measurement of power consumption during playing video streaming service.But, the Web browser have been difficult to apply a communication service only to support video streaming function based on Apple HTTP Live Streaming. For this reason, we developed a new Web browser to measure power consumption of communication services such as a Web service, a microBlog and a social network services (SNS) running on iOS.

1. はじめに

1.1 研究の背景

近年，iPhone や Android などのスマートフォンが急速に普及している．スマートフォンは，次のような特徴を持っている．

(1) API 仕様の公開
(2) LCD 画面とタッチパネルによる入出力
(3) アプリケーションの開発とインストールが可能
(4) 写真，動画が撮影可能
(5) GPS 情報が取得可能

スマートフォン利用者には，Facebook や Twitter などのソーシャルネットワークサービス(SNS)や LINE や Skype などのグループウェアが急速に普及している．さらに，ネットワークが高速化したため，いつどこで何をしたという動画や画像による記録が簡単に共有することが可能になった．しかし，上記のようなサービスを利用するとスマートフォンのバッテリは急速に消費されてしまうため，多くの利用者は，充電器やモバイルバッテリーを常用している．このようなことから，現在の SNS やグループウェアは，電力消費量に配慮されていないと考えられる．また，コミュニケーションサービス全体の電力消費量の削減に関わる研究はほとんど報告されていない．そのため，コミュニケーションサービス全体の電力消費量削減する研究を行う必要がある．

1.2 研究の目的

前述の背景より，コミュニケーションサービス全体の電力消費量削減を目指し，従来研究ではコミュニケーションサービスの中でクライアントの電力消費量に関する測定用ソフトウェアを開発し，ビデオ品質による消費電力への影響について測定を行った．しかし，HTTP Live Streaming (HLS)を利用したため，放送型や配信型の形態となるため，双方向であるコミュニケーションサービスにおいて HLS を利用することは困難であると判明した．また，近年のコミュニケーションサービスは急増するプラットフォームや多彩な端末に対応するために Web ベースのシステムが一般的である．そこで，iOS 端末で Web ページを閲覧した時の電力消費量が測定可能な iOS 対応のブラウザを開発することとした．これにより，既に存在するコミュニケーションサービスにおいて消費電力が多い処理を特定し，その処理やデータ送信量の削減について検討することにした．また，ユーザの集中度によってコミュニケーションサービスの品質を変化させることや，ネットワーク仮想化を行い転送するデータを分散させる．特に本論文では，処理とデータが多いと思われるオンラインビデオに対する電力消費について測定した．

1.3 関連研究

コンピュータネットワーク分野における電力消費量削減に関する研究としては，ネットワークの自立制御による通信路の電力消費量削減に関する研究[1]，アプリケーションプログラム参加の省電力制御に必要な機能の考察[2]，オブジェクトの監視・追跡を行う無線マルチメディアセンサネ

†1 福岡工業大学大学院
Fukuoka Institute of Technology
†2 福岡工業大学
Fukuoka Institute of Technology

Vol.2014-DPS-159 No.24
Vol.2014-MBL-71 No.24
2014/5/16

ットワークの稼働時間延長および QoS 確保のためのルーティング手法[3]，多様な要求品質を持つ移動端末ユーザへのリソース効率の良いビデオ配信方式[4]，快適度の低下を最小限に抑える省エネデバイス制御手法[5]が報告されている．

以上の研究に関して，電力消費量削減関連の研究を分類に分けると，サーバ限定，通信路限定，クライアント限定の 3 つに分ける事ができるが，コミュニケーションサービス全体を対象とした電力消費量の削減については検討されていない．そこで，コミュニケーションサービス全体に関わる統合的な電力消費量削減方法について検討する必要がある．

2. iOS に対応した電力消費量測定用 Web ブラウザの開発

2.1 ビデオ品質と考慮した消費電力量の削減方法

図 1 にオンラインビデオ再生時の構成とオンラインビデオ再生時の電力消費要因を示す．サーバではフォーマットの種類や，フレームレートやパラメータの異なるビデオデータがファイルに格納され，ファイル読み込みとビデオデータの送信が実行される．クライアントではサーバからパケットに分割されたビデオデータが通信を経由して受信され，デコードと表示が実行される．これらを要素としてコミュニケーションサービス全体で電力消費する事と考えられる．このため，サーバではビデオデータの読み込み頻度とデータ量削減，ビデオデータの送信頻度とデータ量削減，通信路ではクライアントへ伝送できない場合にパケットが廃棄されてしまうので，マルチパス転送の様な複数経路の活用をする．クライアントでは，ビデオデータの受信頻度と受信データ量削減，デコード負荷の軽減，ビデオのフレーム表示頻度の削減による電力消費量への影響について検討する．具体的には，利用者がわからない程度に品質を低下させる事によって，以上の様な効果を狙う．

図 2 はビデオの品質に影響を与えるパラメータとして良く知られているフレームレートとフレームサイズを示したものである．フレームレートは，1 秒間あたりのフレーム数を示している．このパラメータは映像中の動きの滑らかさに影響するパラメータであり，これが高いと動きが滑らかに，低いと粗く見える．ビデオ再生中にフレームレートを変化させたところ，ビデオ本来のフレームサイズ以内であれば，ビデオ視聴者は気がつかないという研究報告がある[6] [7]．また，フレームレートを減少させると，フレーム数が少なくなり，ビデオデータの通信量が削減されることになる．このため，フレームレートの低減は電力消費量削減が期待できる．フレームサイズについても同様に，フレームサイズを縮小するとピクセル数が少なくなり，ビデオデータの通信量が削減されることになる．この結果，フレームサイズの縮小は電力消費量削減が期待できる．以上のことから，利用者が気づかない程度にフレームレートとフレームサイズを低減させて電力消費量を抑えた方が良い．

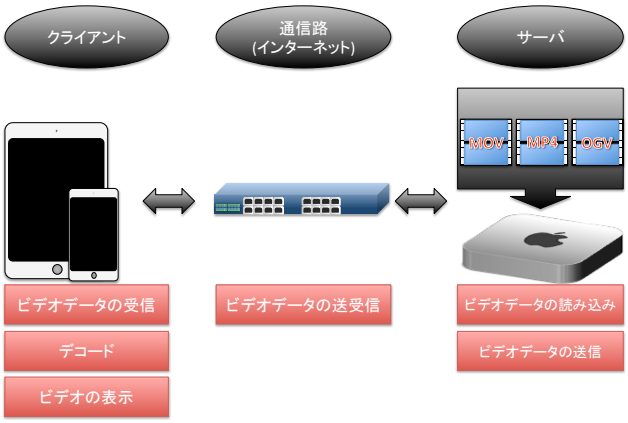

図 1 オンラインビデオ再生時での電力消費項目

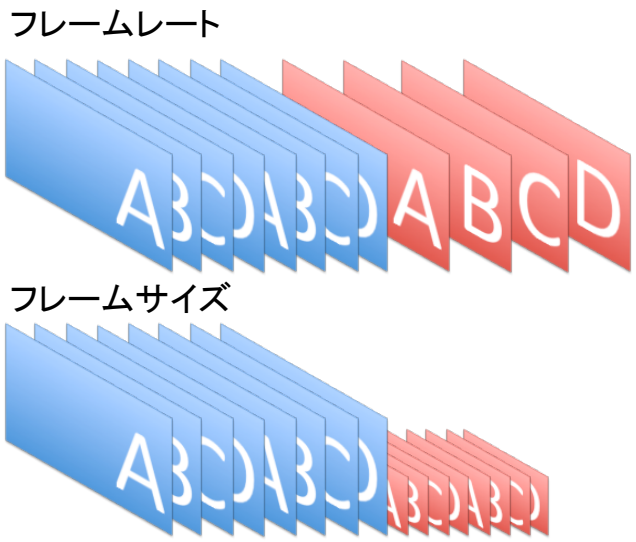

図 2 ビデオ品質パラメータ

2.2 HTTP Live Streaming

本研究では，iOS 端末を対象とした．iOS 端末上で 10 分以上オンラインビデオ再生するには Apple 社による制約によって HLS[8]を利用する必要がある．そこで，HLS を利用する．

HLS とは，通常の Web サーバから HTTP を使ってオーディオとビデオを送信し，iPhone，iPad，iPod touch，Apple TV などの iOS 端末，および MacOSX で再生するための機能であり，ライブ放送と蓄積型のコンテンツのいずれにも対応している．また，ビットレートの異なる複数の代替ストリームをサポートし，クライアントソフトウェアはネットワーク帯域幅の変更に応じて，インテリジェントにストリームを切り替える．また，HTTPS を使用したメディアの

Vol.2014-DPS-159 No.24
Vol.2014-MBL-71 No.24
2014/5/16

暗号化とユーザ認証にも対応している．なお，HLS を図にしたものを図 3 に示す．

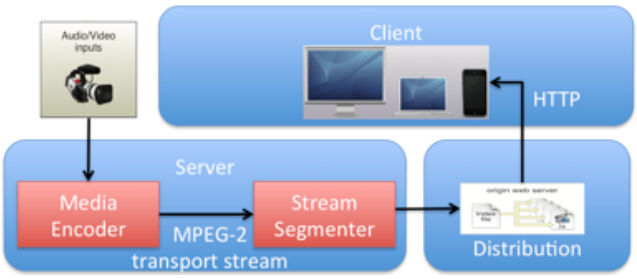

図 3 HLS によるコンテンツ配信

3. 実装

3.1 開発・実験環境

表 1 に，開発時に使用した機材の性能表を示す．本ブラウザはより，Macmini に Xcode5.0.2 をインストールして開発を行った．開発したアプリケーションのデバッグとテストには表 1 左に記載している iPadmini で行った．開発したアプリケーションのデバッグやテスト，実験の環境については図 4 に示す．研究室内 LAN と Macmini 間と研究室内 LAN から TimeCapsule 間は共に 1Gbps でリンクしている．TimeCapsule から iOS 端末などのクライアントに対しては，IEEE 802.11g では 54Mbps，IEEE 802.11n では 65Mbps - 600Mbps でリンクしている．

次に，実験に使用したビデオデータの詳細を説明する．撮影機材として、Canon iVIS HFM51 を使用した．撮影条件は，フレームレート 60fps，研究室の窓から外を，128GB の SD カード容量が一杯になるまで撮影した．音声は編集でカットしている．

表 1 使用した機材の性能表

構成要素	iPadmini(第1世代 2012年モデル)	Macmini(2013年モデル)
CPU	AppleA5 2コア 1.0GHz	Intel Core i7 2.6GHz
GPU	PowerVR SGX 543MP2 2コア	Intel HD Graphics 4000
RAM	512MB (LPDDR2)	16GB (DDR3)
NIC	IEEE802.11 a/b/g/n	10/100/1000BASE-Tギガビット Ethernet
Power	4400mAh	AC100V

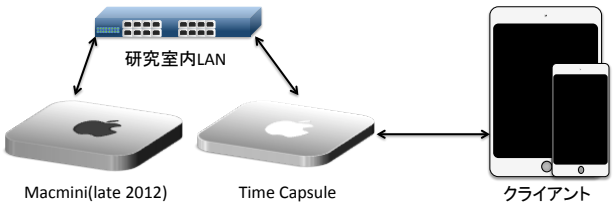

図 4 実験環境

3.2 実装したアプリケーション

実装した機能は，Web ページを表示する機能と一定間隔でバッテリ残量を 2 つの API で取得して CSV ファイルに書き込む機能である．

Web ページを表示する機能は，UIWebView という Apple が提供している Xcode に含まれるものを使用した．URL を NSURL 型で渡すだけで簡単に Web ページを表示してくれるモジュールである．また，ブラウザとして「戻る」，「進む」，「読み込み中止」，「更新」といった操作機能もボタンを配置して関連付けるだけで実装ができる．

次にバッテリ残量取得機能は，5%区切りでバッテリ残量の取得が可能な Apple 提供する API と 3%の誤差でバッテリ残量の取得が可能な iOS 端末自体に存在するダイナミックライブラリを使用して実装した．

3.3 モジュール構成

図 5 に開発したアプリケーションのモジュール構成図を示す．モジュール構成図は，開発言語が Objective-C のため，MVC モデルでモジュール図を作成した．開発したアプリケーションは，ビューが 2 つあるが，図 5 ではビューを簡略化している．

開発したアプリケーションのモジュール構成としては，大まかに分けると，設定を管理するビューと，Web ページを表示しながらバックグラウンドでバッテリ残量を測定するビューに分けられる．設定を管理するビューでは，表示する URL と実験時の画面輝度の設定が可能である．また，これまでの実験概要が一目見て解るようにするため，簡単な概要を集計して表示する機能も実装した．この機能は，実験以外の状態で行うため，測定結果に影響は出ない．次に，Web ページを表示しながらバックグラウンドでバッテリ残量を測定するビューでは，前の画面で設定された URL を読み込んで表示する．画面左上には公式の API と非公式の API で取得したバッテリ残量を表示する．さらに，画面右上には Workload を表示しており上から使用メモリの差分，消費メモリ量，1 秒間での CPU 使用時間，View の数となっている．画面中央には，UIWebView を設置し，これに NSURL 型で URL を渡すと指定した URL を読み込んで表示することが出来る．また，Web ブラウザのため「戻る」，「進む」，「読み込み中止」，「再読み込み」といった機能もついており，画面下にボタンとして実装してある．実験をやり直したい時には強制終了ボタンを押すことで実験を終了することが出来る．なお，バックグラウンドでは一定間隔で，日時，バッテリ残量，CPU 使用時間，その時に表示していた URL，画面輝度を CSV ファイルへ書き出している．この CSV ファイルは，iTunes を利用してファイル共有を行えるよう設定をしているため，ディベロッパではなくても取り出すことが可能となっており，アプリ単体でのテストも可能である．なお，図 6 と図 7 に開発したアプリケーションのスクリーンショットを示す．図 6 が設定を管理する画面であり，図 7 が Web ページを表示しながらバッ

Vol.2014-DPS-159 No.24
Vol.2014-MBL-71 No.24
2014/5/16

クグラウンドでバッテリ残量を測定する画面である．

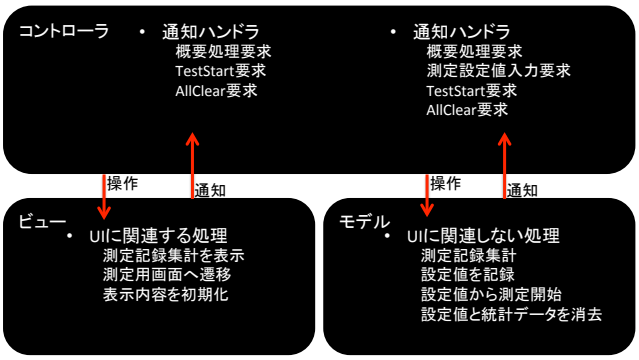

図 5 モジュール構成図

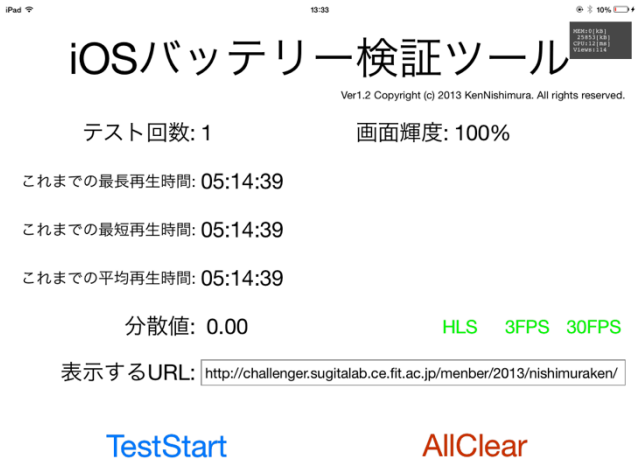

図 6 Battery Logger Top View

図 7 Experimentation View

4. 予備実験

4.1 予備実験手順

(1) iOS 端末を iTunes へ接続し復元．(バージョンを同じにするためにリストア用 IPSW ファイルを使用する)
(2) Xcode と接続して開発したアプリケーションをインストール
(3) iOS 端末をバッテリ残量が 100%になるまで充電．
(4) 外部電源との接続解除
(5) アプリケーションの起動
(6) URL と画面輝度を設定して TestStart を押す
(7) 実験対象を開いてはじめの画面に戻るまで閲覧
(8) iTunes へ接続し，CSV ファイルを取り出す

実験が失敗した場合は(3)からやり直す．また，実験を途中でやめたい場合は，強制終了ボタンを押すことではじめの画面に戻れる．AllClear ボタンはデータがすべて初期化されるため使用には注意する．CSV ファイルには下記のデータが保存される．表 2 に CSV ファイル例を示す．実際のファイルでは，(a)，(b)，(c)順で出力される．フォーマットの設定として，Date は本体から取得した日付，Time は本体から取得した時間，PublicAPI は公式の API で取得したバッテリ残量，PrivateAPI は非公式の API で取得したバッテリ残量，RSS は実メモリ量，VM は仮想メモリ量，CPUTimePerSec は 1 秒間あたりの CPU 使用時間，TargetURL は UIWebView が表示している URL，SetBrightnes は画面輝度を 0.0-1.0 の範囲で示している．(c)の図では URL の一部表示となっているが，UIWebView から取得した URL がすべて入る．また，グラフ化するときは，Time と PublicAPI の間にセルを 2 つ追加する．1 つ目のセルでは Date と Time を加算させることによって日時を求める．2 つ目のセルでは上のセルと減算して 24 倍することにより 1 時間を 1.0 とした数値で経過時間を求める．このセルがグラフでは横軸となる．CPUTimePerSec の隣にも 1 つセルを追加する．ここでは，CPUTimePerSec の値を 1000 で除算し 100 倍することによって CPU 使用率を求めている．以上の操作によって結果の集計を行った．

4.2 予備実験対象

本論文では，次に挙げるパラメータで実験を行った．

実験(1) 30fps の動画を画面輝度 100%
実験(2) 12fps の動画で画面輝度 100%
実験(3) 30fps の動画で画面輝度 50%
実験(4) 12fps の動画で画面輝度 50%

なお，ここで使用した動画の詳細については表 3 に示す．また，動画は Apple 社の FinalCutProX で編集しており Compressor で HLS 向けのエンコードを行っている．HLS 向けの設定でフレームレートの一定化を行っており，フレームレート以外の設定は変化させていない．なお，フレー

ムサイズを変更する場合は，1 つの HLS 形式のプレイリストファイルである m3u8 ファイルを指定して再生することによってフレームサイズの一定化を行っている．

4.3 予備実験結果

図 8 に実験結果を示す．実験(1)は 5.24 時間再生，実験(2)は 5.32 時間再生，実験(3)は 5.67 時間再生，実験(4)では 8.97 時間再生できた．よって，画面輝度を 50%にすることで，再生可能時間が 7.58%増加したことがわかった．

また，表 3 にグラフから Excel の線形近似機能を使用して，近似曲線を引き，その数式をまとめたものを示す．さらに，関係式を下記に挙げる．

$$f_b(t) = -at + b \cdots (式1)$$

ここで，式 1 の各変数は下記を表している．

a:バッテリ消耗率[%]

b:初期バッテリ残量[%]

t:時間[h]

表 2 出力される CSV ファイル

(a) 出力される CSV ファイル

Date	Time	PublicAPI[%]	PrivateAPI[%]
2014/4/2	10:11:03	100	98
2014/4/2	10:11:13	100	98
2014/4/2	10:11:23	100	98
2014/4/2	10:11:33	100	98

(b) 出力される CSV ファイル

RSS[Byte]	VM[Byte]	CPUTimePerSec[ms]
25661440	537272320	7
30523392	561045504	76
30666752	558866432	28
30658560	557957120	22

(c) 出力される CSV ファイル

TargetURL	SetBrightnes
about:blank	0.5
http://challeng	0.5
http://challeng	0.5
http://challeng	0.5

表 3 動画詳細

撮影機材		Canon iVIS HFM51
撮影条件	場所	研究室
	撮影時間	12時間
	音声	なし
	フレームレート	60fps

表 4 の(a)，(b)より，30fps の動画で画面輝度を 50%削減すると，PublicAPI は 1 時間あたり 8.45%の電力消費量削減，PrivateAPI は 1 時間あたり 5.86%の削減となった．同様に(c)，(d)より，PublicAPI は 1 時間あたり 41.07%の電力消費量削減，PrivateAPI は 1 時間あたり 42.27%の電力消費量削減となった．しかし，実験(1)-(3)と実験(4)のグラフをでは処理時間に大きな差が生じているため，この原因を究明する必要がある．

表 4 近似値

(a) 30fps PublicAPI

30fps PublicAPI		
画面輝度[%]	a	b
100	18.67	106.04
50	17.09	106.76
バッテリ消費削減率[%]	8.45	-

(b) 30fps PrivateAPI

30fps PrivateAPI		
画面輝度[%]	a	b
100	17.76	101.37
50	16.72	104.81
バッテリ消費削減率[%]	5.86	-

(c) 12fps PublicAPI

12fps PublicAPI		
画面輝度[%]	a	b
100	18.48	106.15
50	10.89	104.93
バッテリ消費削減率[%]	41.07	-

(d) 12fps PrivateAPI

12fps PrivateAPI		
画面輝度[%]	a	b
100	17.56	101.45
50	10.31	100.07
バッテリ消費削減率[%]	41.27	-

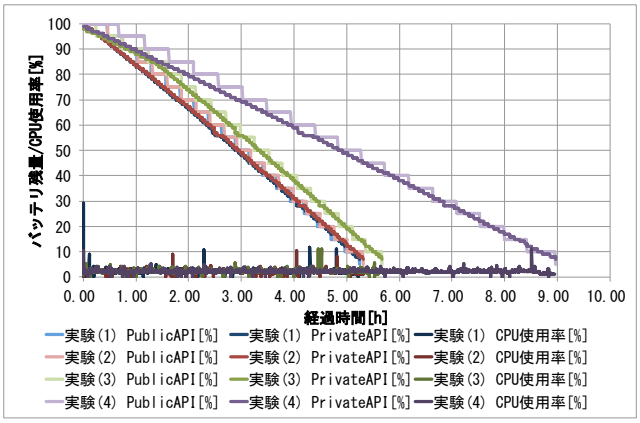

図 8 実験結果

5. まとめ

本論文では，iOS 端末で Web ページを閲覧した時の電力消費量を測定できるアプリケーションを開発した．予備実験を行ったところ，電力消費量が取得することが可能になった. バッテリ取得には 2 つの異なる API を使用している．今後は，実験方法やパラメータなどの再検討し，本実験を行う予定である．iOS 端末自体には Wi-Fi や Bluetooth をはじめとする通信モジュールやセンサーが付いており，それぞれがどの程度電力消費量に影響を及ぼすのか予想が困難であり，実験対象も動画だけではなく異なるデータでの比較なども行い，電力消費量に影響を与える要素を今後明らかにしたい.

参考文献

1) 白鳥則郎, 稲葉勉, 中村直毅, 菅沼拓夫, "災害に強いグリーン指向ネバーダイ・ネットワーク", 情報処理学会論文誌, Vol.53, No.7, 1821-1831, July 2012
2) 古市 実裕,相原 達:アプリケーションプログラム参加の省電力制御に必要な機能の考察, IPSJ OS/DPS 合同研究会, Vol.98, No.15, pp.135-140, (1998)
3) 藤本恭平, 安本慶一, 孫為華, 山内由紀子, 伊藤実, “オブジェクトの監視・追跡を行う無線マルチメディアセンサネットワークの稼働時間延長および QoS 確保のためのルーティング手法,” DICOMO2011 シンポジウム論文集, pp1156-1166, 2011.7.6.
4) 山岡修一, 孫タオ, 玉井森彦, 柴田直樹, 伊藤実, “多様な要求品質を持つ移動端末ユーザへのリソース効率の良いビデオ配信方式(QoS)”, 情報処理学会研究報告. MBL, [モバイルコンピューティングとユビキタス通信研究会研究報告] 2005(113), 77-84, 2005-11-17
5) 安本 慶一, 小倉 和也, 山本 眞也, 伊藤 実, ” 快適度の低下を最小限に抑える省エネデバイス制御手法”, 第 149 回 マルチメディア通信と分散処理(DPS)研究会, 情報処理学会研究報告 Vol.2011-DPS-149 No.9, pp. 1-8, (November 2011).
6) Kaoru Sugita, Masao Yokota, “Considerations of Effectiveness of Media Switching and QoS Functions”, Proc.of the Seventh International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), pp. 241-244,Nov. 2012.
7) Kaoru Sugita, Masao Yokota, “Building Efficient Questionnaire Survey of Video Contents with Different QoS Parameters”, Proc.of the International Conference on Network-Based Information Systems (NBIS’2013), pp.323-326, Sep.2013.
8) HTTP Live Streaming の概要 https://developer.apple.com/jp/devcenter/ios/library/documentation/StreamingMediaGuide.pdf (2013)