Android 端末の無線 LAN 通信性能の評価

三木香央理　(指導教員：小口 正人)

1　はじめに

近年, スマートフォン市場の成長に伴い, 携帯端末で動作する組込み機器のソフトウェアプラットホームとして Google 社開発の Android[1] が注目されている. アプリケーション開発や柔軟な拡張性において注目度の高い Android 携帯に対し, 本研究ではそのサービス提供を可能にしたシステムプラットホームとしての Android に焦点を当て, 特にそのネットワークおよびネットワークコンピューティング能力について評価する. 特に Android 携帯の無線 LAN の通信能力について解析し, その設定に手を加えることで, より高性能な通信を目指す.

2　Android

Android のアーキテクチャを図 1 に示す. Android は Linux2.6 カーネルを用いて構築されており, この OS に各種コンポーネントを追加し Android というプラットフォームを構成している. 市場に複数ある Linux パッケージでも Android が構成できるようになっている.

また,Linux カーネルの上に Android 独自のアプリケーション実行環境である Android Runtime を実装し,Dalvik と呼ばれる独自の仮想マシンを搭載している. これは Java の仮想マシン（JVM）に相当する.

その上にアプリケーション・フレームワーク, アプリケーションが乗る形態であるため, アプリケーションは Dalvik にあわせて開発すればよく, ポータビリティが高い. このように, これまでの携帯端末用開発ツールとは異なり, オープンソースでありキャリア間の制約がない. そのためユーザのカスタマイズ自由度が高く, アプリケーション開発の負荷が軽減され, 他キャリア, 他機種への柔軟な拡張性があるといえる.

一方, 通信は Linux カーネルの中のプロトコルスタックを用いて実行されているため, この TCP 実装部分などで性能が決まってくると考えられる. そのため, 本研究ではカーネル中のトランスポート層実装に焦点を当て評価を行う.

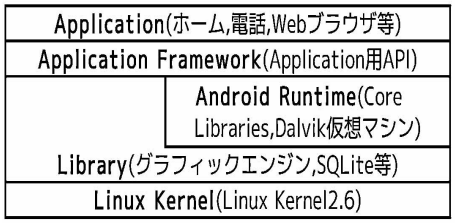

図 1: Android のアーキテクチャ

3　研究概要

本研究では, まず AP の近くにいるユーザ同士の通信を想定し,Wi-Fi 環境における同一無線 LAN 内の通信を評価した. 次に Wi-Fi を用いず,Bluetooth を用いたアドホック接続の通信を評価し, 最後に遠隔地に存在するモバイルクラウドを提供するサーバへ携帯端末からアクセスする通信の計 3 パターンの無線通信のケースを想定しその性能を評価した. スループット測定は iperf-2.0.4[2] をクロスコンパイルし,Android 実機に送り込んで実行した.

3.1　実験手順

表 1,2 および図 2〜4 に本研究の実験環境の一部を示す.

Ubuntu	OS	Ubuntu8.10(Linux2.6.27-7)
	CPU	iIntel(R) Pentium(R) M processor 1.73GHz
	Main Memory	512MB
WindowsXP	OS	Microsoft Windows XP
	CPU	Intel(R) Pentium(R) 4 CPU 3.00GHz
	Main Memory	512MB
	Bluetooth	Bluetooth Ver2.1+EDR
mobilegw	OS	Fedora Core release 3
	CPU	Intel(R) Pentium(R) 4 CPU 3.00GHz
	Main Memory	1GB

表 1: 実験環境

Android1	Model number	HT-03A
	Firmware version	1.5
	Baseband version	6.2.50S.20.17H_2.22.19.26I
	Mod version	docomo standard
	Build number	CDB72
	CPU	Qualcomm MSM7201A 528MHz
Android2	Model number	T-Mobile myTouch 3G
	Firmware version	1.6
	Baseband version	2.6.29.6-cm42 shade@toyxygene
	Mod version	CyanogenMod-4.2.3.1
	Build number	DRC92
	CPU	Qualcomm MSM7201A 528MHz
	Bluetooth	Bluetooth Ver2.0+EDR

表 2: 実験環境 (Android 側)

3.2　測定環境

図 2: Wi-Fi を用いた AP 経由の通信

図 3: Bluetooth を用いた AP を経由しない通信

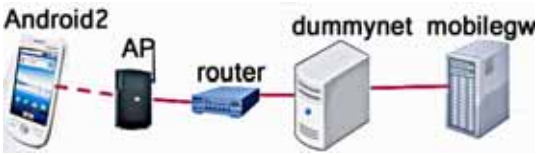

図 4: 高遅延環境におけるサーバと Android の通信

4　実験概要

4.1　Wi-Fi および Bluetooth を用いた通信

Wi-Fi 環境における同一無線 LAN 内の通信においては,Ubuntu から Android への通信の平均スループットは 9.83(Mbps), 逆方向は 8.56(Mbps) と,Android の方が送信性能が低いことがわかった. また UDP 通信は,Ubuntu から Android への平均スループットが 8.85(Mbps), 逆方向は 6.47(Mbps) となった. この結果は buffer size にスループットが依存しなかった. また Android から Android への通信の場合,TCP 通信の平均スループットは 4.62(Mbps),UDP 通信は 5.88(Mbps)

という結果になった.

Bluetooth による通信も,WindowsXP から Android への通信の平均スループットは 1.56(Mbps), 逆方向は 1.44(Mbps) と Android の方が送信性能が低いことがわかった. しかし,Wi-Fi を用いた AP 経由による通信の結果よりも, スループットの差が小さいことがわかった. また TCP 通信においてスループットは window size に依存しないことがわかった. Bluetooth を用いた UDP 通信のスループットの値は TCP 通信とほぼ同じ結果になった. さらに UDP 通信では Android から WindowsXP に送る場合 buffer size に結果は依存しないが WindowsXP から Android に送る場合, 受け手である Android の buffer size が大きい方がスループットが高くなることがわかった.

4.2 モバイルクラウドを利用する通信

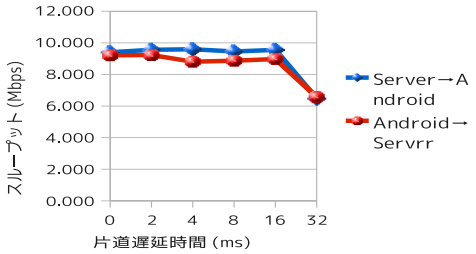

図 5: 遅延環境におけるサーバと Android の TCP 通信

遅延装置を使って, 片道遅延時間 0-32ms の遅延環境を作り,Android と mobilegw 間のスループットを測定した. 結果は図 5 に示す通り, 片道遅延時間が 16ms までは安定したスループットが得られるが, それ以上になると大幅にスループットが低下してしまうことがわかった. UDP における通信では mobilegw から Android への平均スループットが 7.92(Mbps), 逆方向が 6.00(Mbps) という結果になった. UDP 通信では片道遅延時間が 32ms になってもスループットは安定していて低下しなかった.

4.3 TCP チューニング

次にモバイルクラウドを利用するケースにおいて TCP の輻輳制御アルゴリズムを変えてみることでその性能の違いを測定した. 本研究で用いた輻輳制御アルゴリズムは,bic,cubic,htcp,reno,westwood の 5 種類である. TCP チューニングを行うためにサーバの OS を Fedora release 9 にバージョンアップした.

この OS のデフォルト輻輳制御アルゴリズムは cubic である. bic は RTT 公平性に着目し積極的に window size を増加させるアルゴリズムである. cubic は bic の改良版で window size の増加が緩やかである. htcp はロスベースの高速 TCP であり, RTT や帯域幅が極端に大きい広域・高速ネットワーク上では, その帯域幅に合わせて効率的な振る舞いをする. reno は輻輳が発生したときの輻輳 window size の 2 分の 1 の値を ssthresh に設定し, 後に輻輳が発生した場合は輻輳 window size を ssthresh から輻輳回避段階に入るアルゴリズムである. これにより過剰な輻輳 window size の減少を避けられるため高速な転送を可能としている. westwood は reno に比べ, 太い帯域で且つ漏れやパケットロスに強いアルゴリズムである.

図 6 にパケットロスのない環境でサーバに各アルゴリズムを適応した測定結果を示す.5 種類の輻輳制御アルゴリズムに大差は見られなかった. 0.2 %のパケットロスを加えて測定した結果は遅延時間が大きくなると多少の差が見られた.

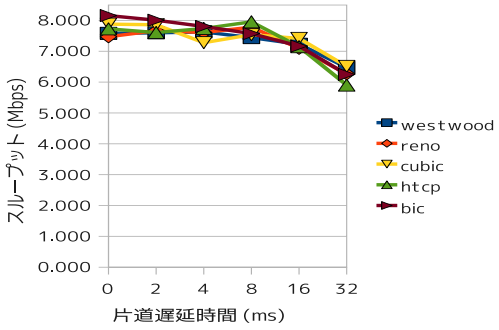

図 6: 遅延環境におけるサーバと Android の TCP 通信

また Android 側で適用可能な輻輳制御アルゴリズムは westwood と reno の 2 種類であるが, こちらも性能に大差は見られなかった. 実験から Android の性能は輻輳制御アルゴリズムよりも周りの環境の問題に強く影響されることがわかった. これを次章に示す.

5 干渉の影響

本研究では測定時に Android とアクセスポイント間を Wi-Fi 接続にて測定している. Wi-Fi で使われている 2.4GHz の周波数帯は混雑しており干渉の影響を受けやすいため近くに強い Wi-Fi の電波があるとスループットが 0.5〜0.8 倍まで低下してしまう.

また Wi-Fi は Bluetooth による影響を受けやすく, 半径 10m 以内に Bluetooth 搭載機器が稼働していた場合, スループットは半分程度まで減少してしまう [3].

図 7 に干渉時の TCP チューニングの結果を示す.

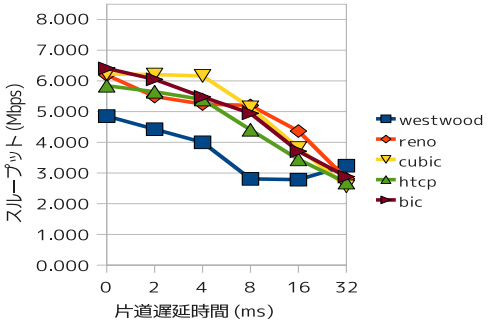

図 7: 干渉遅延環境におけるサーバ Android 間 TCP 通信

6 まとめと今後の課題

Android における様々な通信ケースを想定し, それらのスループット測定を行った. 結果からそれぞれの通信ケースにおけるスループットが window size に依存するときとしないときがあることなど, それぞれの通信ケースの特徴がわかった. 更に TCP の輻輳制御アルゴリズムを適用させた実験を行ったが,Android は輻輳制御アルゴリズムよりも特に送出時に Wi-Fi の干渉に強く影響されることがわかった. 今後は Android のカーネルの書換え等により, 干渉が発生する環境においても TCP 通信における性能向上を目指す.

参考文献

[1] Android:http://www.google.co.jp/mobile/android
[2] Iperf:http://downloads.sourceforge.net/project/iperf
[3] http://ednjapan.rbi-j.com/content/issue/2005/11/content03.html
[4] 三木香央理, 小口正人:"Android 端末の様々な無線 LAN 環境における通信性能の考察", DEIM2010,2010 年 3 月発表予定