

Android 端末における往復遅延時間を考慮した通信制御ミドルウェアの改良

早川 愛 (指導教員：小口 正人)

1 はじめに

近年スマートフォンの爆発的な普及と高機能化に伴い、その大容量高速通信に対する需要が高まっている。しかしスマートフォンのようなモバイル端末は、低帯域でかつノイズの影響を受けやすい無線通信を利用する機会が多いため、その通信性能を向上させることが求められている。そこで、スマートフォンの中でも最もシェア率が高くオープンソースソフトウェアを用いている Android 端末を対象にして、同一アクセスポイントにつながる全ての端末がお互いの通信状況を知らせ合うことにより、連携した通信制御を行うことを目的としたミドルウェアの開発が行われている。

本研究は、通信性能を低下させる要因の一つとして各端末の通信時の往復遅延時間 (RTT) に焦点を当て、それがどのように通信性能に影響するかを明らかにした上で、このミドルウェアの改良を目指した検討を行う。

2 Android

Android 携帯は、Google 社を中心に開発されている携帯端末向けの OS であり、主にスマートフォンやタブレット型 PC で使用されている。Android は、Linux をベースとし、オープンソースで提供されているためキャリア間の制約がなく、様々なデバイスに自由に应用することができるという点で大変注目されている。そのような背景から、スマートフォン OS 中でのシェア率も年々上がってきており、2012 年第 3 四半期では全世界で 72.4% ものシェアを占めている [1]。

以上の理由から、本研究で取り扱うスマートフォン OS として Android を選択した。

3 既存研究

3.1 通信制御ミドルウェア

本研究で改良を加える通信制御システム [2] の概要を説明する。このシステムでは、Android 端末が広帯域有線ネットワーク接続されたクラウドサーバと通信する場合を想定し、輻輳が懸念されるアクセスポイント Android 端末間の無線帯域を共有している他端末の通信状況を考慮した制御を目指している。そこで、同一アクセスポイントを共有する無線 LAN 空間内において、互いの端末の通信状況、すなわち輻輳ウィンドウを通知し合うことで、全体のトラフィックを予測し、周囲の他端末の通信状況に応じて、輻輳制御アルゴリズムを適切に調整する。それにより、単独に通信するよりも精密な帯域見積りが実現可能となる。

3.2 カーネルモニタ

前項で述べたシステムのベースとして用いられているツールを紹介する。カーネルモニタは、通信時におけるカーネル内部のパラメータ値の変化を記録できる、オリジナルシステムツールである。カーネル内部の TCP ソースにモニタ関数を挿入し再コンパイルすることで、輻輳ウィンドウ値やソケットバッファのキュー長、各種エラーイベントの発生タイミングなどの TCP パラメータを見ることが出来る。

このツールを Android に組み込み、解析を行う。

4 端末数と通信性能に関する基礎実験

4.1 実験概要

図 1 に示すように、サーバ機と Android 端末の間に実環境を模擬するための人工遅延装置 dummynet を挟み、端末を 1~10 台通信させた時の各往復遅延時間におけるスループットとエラーイベントを取得し、さらに同時に 1 台の端末に対して ping コマンドを用いて実際にかかる往復遅延時間の時間変化を調べた。

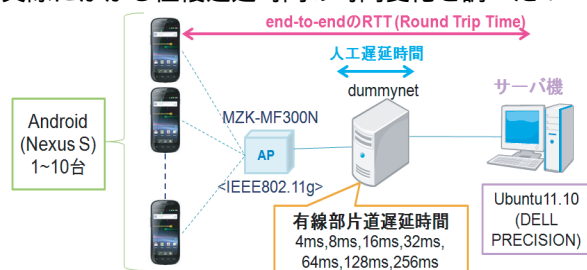


図 1: 実験トポロジ

4.2 実験環境

本実験で使用した実験環境を表 1 に示す。また、無線通信方式は IEEE802.11g で行った。

表 1: 実験環境

Android	Model number	Nexus S
	Firmware version	2.3.4
	Baseband version	I9023XXKD1
	Kernel version	2.6.35.7-hiromi0824
	Build number	GRJ22
server	OS	Ubuntu 11.10 (64bit) / Linux 3.0.1
	CPU	Intel(R) Core 2Quad CPU Q8400
	Main Memory	7.8GiB

4.3 実験結果と考察

図 2 に通信台数とスループット (平均と合計) の関係を示す。図 2 より、通信端末数を増やすと、合計通信速度が大幅に低下することが分かる。さらに、低下する台数も人工遅延時間によって違うことがわかった。

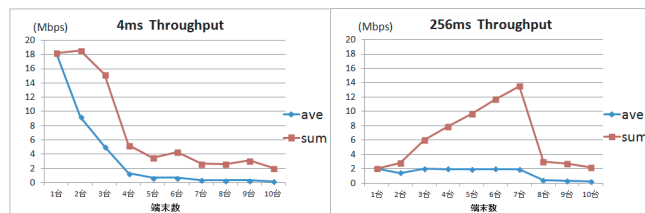


図 2: 通信台数の変化によるスループットの平均値、合計値

この原因を調べるために図 3 の輻輳ウィンドウ値と ping により測定した end-to-end の往復遅延時間を見ても、輻輳ウィンドウは所々落ちているものの比較的安定して高い値を保持しており、性能劣化に大きな影響を及ぼしているとは考えにくい。それに対して、往復遅延時間の遷移を見てみると人工遅延装置で設定した有線部の遅延時間の値よりもはるかに大きな遅延が観察された。

このことから本実験の考察として、往復遅延時間の大幅な増加が通信速度の低下につながるのではないかと予想できる。そこで、カーネルモニタで取得するパラメータに往復遅延時間とその最小値を追加し、常時観察できるように改変した。

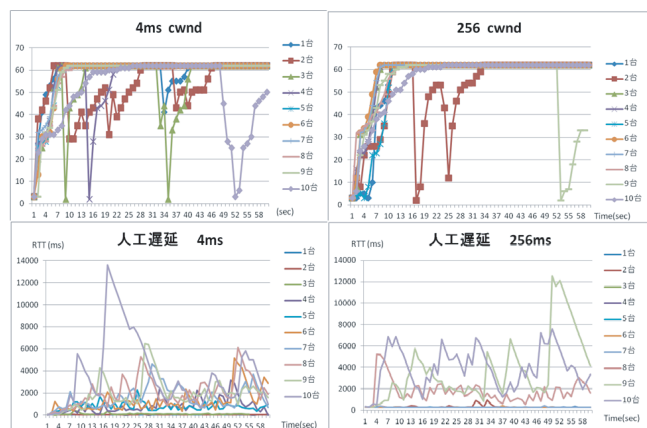


図 3: 輻輳ウィンドウと実際の遅延時間の遷移

5 ミドルウェア実装に向けた検証実験

5.1 実験概要

往復遅延時間が通信性能に与える影響を明らかにするために、前節と同じ実験環境において改変後のカーネルモニタを導入した Android 端末を用いた実験を行った。

5.2 実験結果と考察

有線部の人工遅延時間 16ms で Android 端末を 10 台で通信させた時のスループット、カーネルモニタで取得した往復遅延時間の遷移を図 4 に示す。グラフから、スループットが高いところでは遅延時間は小さく、逆にスループットが低いところでは遅延時間は大きいという対比が見られた。

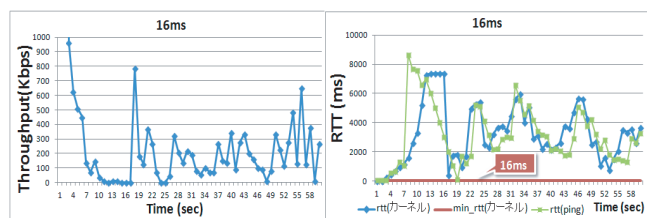


図 4: スループットと往復遅延時間の遷移

このことから、前節で予想した通り往復遅延時間の大幅な増加が通信性能劣化の大きな要因だと考えられる。よって、通信速度を向上させるためには、往復遅延時間の増減を考慮した制御が有効であると言える。

6 提案ミドルウェアの概要

ここで、本研究で提案する通信制御ミドルウェアの概要を説明する。

変更前のミドルウェアでは、発信部と受信部に分かれて、それぞれで制御を行っていたが、本研究では、実際に最適化チューニングを行う受信部においてもカーネルモニタの読み込みが必要となったため、この受信部と発信部を一括にまとめることで、導入や制御の簡単化を実現した。

このミドルウェアで実際に行われる制御としては、

まず通信中にカーネルモニタを常時監視し、RTT とその最小値を取得する。取得した値をもとに RTT/最小値で RTT の増減の比率を求める。また同時に、パケットを受信し、他端末の通信状況を把握してトラフィックを予測する。RTT の比率と通信台数の情報をもとに、外部プロセスから制御可能な proc インタフェースを用いて輻輳ウィンドウの上限値と下限値を設定し、最適化チューニングを行う。

7 提案ミドルウェアの評価実験

7.1 実験概要

4 節と同じ実験環境において、本研究で開発したミドルウェアを Android 端末 10 台に導入し、その評価実験を行った。

7.2 実験結果と考察

合計性能を表すトータルスループットの結果を図 5 に示す。青がミドルウェアなしの状態、赤が提案手法を用いた制御によるものである。グラフから人工遅延 64ms においては少数台では性能が向上したが通信端末数が多くなるとあまり成果が見られなかった。しかしながら人工遅延 256ms 高遅延環境においては多端末通信の時に大きく性能が向上した。よって、遅延時間が大きい場合においてこの提案手法は効果があると考えられる。

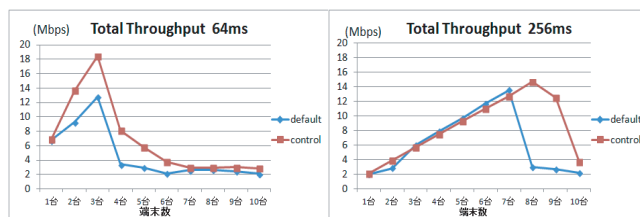


図 5: トータルスループット

8 まとめと今後の課題

本研究では、同時通信端末数が多い環境において合計通信速度が大幅に低下する問題に着目し、輻輳ウィンドウ値に加える制御パラメータとして往復遅延時間を用いる手法についての考察を行った。実験の結果から、輻輳ウィンドウ値以上に往復遅延時間の大幅な増加が通信速度の劣化につながるとわかった。そこで、その往復遅延時間の増加の比率を制御のパラメータとして取り入れ、より最適な帯域見積りを行う通信制御ミドルウェアを開発した。

今後の課題としては、さらなる高性能化を目指しより精密な最適化チューニングを行う。さらに、成果が見られなかった低遅延環境の多数台通信における性能向上に向けた他の制御手法を考案したい。

参考文献

- [1] <http://www.gartner.com/it/page.jsp?id=2237315>
- [2] 平井 弘実, 山口実靖, 小口正人: Android 端末を用いた周辺端末からの情報に基づく協調的制御手法の提案, DICOMO2012, 7H-2, 2012 年 7 月。
- [3] 早川愛, 平井 弘実, 山口実靖, 小口正人: Android 端末を用いた遅延時間を考慮したミドルウェアの高機能化, DEIM2013, 2013 年 3 月。