カーネルモニタを用いた Android 端末の無線 LAN 通信時の TCP 解析

理学専攻　情報科学コース 三木　香央理

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フォームとして Android[1] が注目されている．汎用の PC などとはアーキテクチャが異なる組込み機器において, カーネルの中の振舞を把握することができるカーネルモニタツールを開発し, 通信時の内部動作を解析することで, Android の通信を評価する．また，特定の環境下で効率良く各端末が平等に通信できるように TCP のソースコードに手を加えた独自の制御手法を提案する．

2　Android のアーキテクチャ

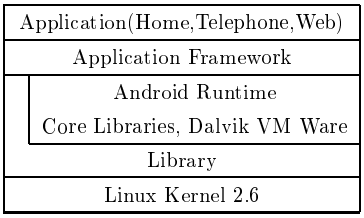

図 1: Android **のアーキテクチャ**

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．通信は Linux カーネルの中のプロトコルスタックを用いて行われているため, 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 Android における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．

3　カーネルモニタ

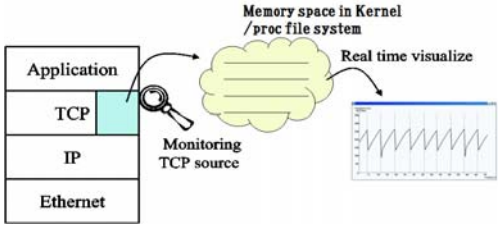

図 2: **カーネルモニタの概要**

カーネルモニタは通信時に, カーネルのどの部分がいつ実行されて, その結果カーネル内部のパラメータの値がどのように変化したかを記録することができるツールである．図 2 に示すように，カーネル内部の TCP ソースにモニタ関数を挿入しカーネルを再コンパイルすることで TCP パラメータをモニタ可能にしている．これによりモニタできるようになった値は，輻輳ウィンドウ, ソケットバッファのキュー長の他，各種エラーイベント (Local device congestion，重複 ACK，SACK 受信，タイムアウト検出) の発生タイミング等がある [2]．本研究ではこのカーネルモニタを組込み機器である Android に応用できることを確認し，解析を行った．

4　提案手法

Android 端末を用いた基礎実験において, 通信のタイミングや周囲の状況によって不必要に輻輳ウィンドウサイズを減少させ不公平な通信状況に陥ることが確認された．そのような状況の場合，本研究では独自の TCP に切り替えて通信性能の向上を図る．実際に輻輳が起きている訳でないにも関わらず, 不必要に輻輳ウィンドウを下げてしまうと，輻輳が起こる前の状態まで輻輳ウィンドウサイズを増加させるのに時間が掛かってしまう．そのため，輻輳ウィンドウサイズが不必要に下がらないように，輻輳ウィンドウサイズダウンのフェーズを省き, また輻輳ウィンドウサイズ閾値 (ssthresh) の値を変更し，安定して高い値の輻輳ウィンドウサイズを保てるようにした．

独自の TCP は輻輳ウィンドウサイズを任意の値まで一気に上昇させ，輻輳が起きた際は輻輳ウィンドウを少し減少させ, その後輻輳回避フェーズで線形に上昇させる．独自のアルゴリズムは RTT=32(ms) 以上の高遅延環境において性能が向上することを確認した．また，性能の良い端末が帯域を占有してしまう場合，輻輳ウィンドウサイズの上限値を変更することで，その端末の送り過ぎを防ぐ．図 3 に独自のアルゴリズムを示す．

図 3: **独自の** TCP

5　実験システムと基本性能測定

5.1　実験環境

図 4 に示すように Android 端末として，HTC 社製と Samsung 社製の携帯電話（スマートフォン）を用いた．IEEE802.11g 無線 LAN 機能を用いて AP を経由でソケット通信した場合の Android 端末の基本性能を測定した．

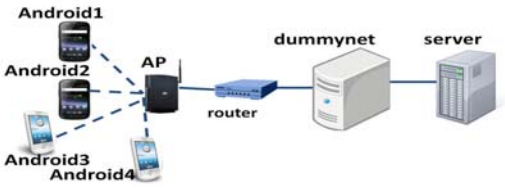

図 4: **ヘテロ端末環境**

5.2 基本性能

ここで HT-03A は初期に発売された Android 端末で Nexus S は比較的最近発売された Android 端末であり，ハードウェアの性能比として HT-03A:Nexus S は約 1:2 であることから，スループットもこの割合で帯域を割った値である 3.5(Mbps):7.5(Mbps) 程出るのが望ましい．しかし実験結果は図 5 に示すように全ての端末において通信性能が著しく低下することが確認された．

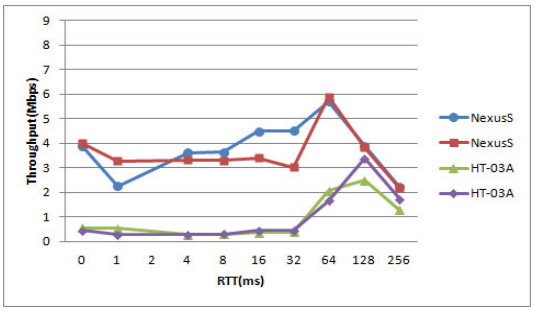

図 5: 4 台の Android 端末同時通信性能

6 提案手法を用いた制御方法

そこで提案手法である独自の TCP を用いて制御を行った．制御方法を表 1 に示す．

表 1: TCP 制御方法

RTT	Nexus S		HT03-A	
	TCP	上限	TCP	上限
0	default	20	original	70
1	default	20	original	70
2	default	20	original	70
4	default	20	original	70
8	default	20	original	70
16	default	20	original	100
32	default	30	original	100
64	default	55	original	100
128	default	55	original	100
256	default	100	original	100

ここで TCP の default は端末に default で実装されている TCP を用いる場合で，original は独自の TCP を用いる場合である．また上限は輻輳ウィンドウサイズの上限を遅延時間を考慮して変更していった時の値であり，ここで表 1 に示す値は本実験においての最適値である．Nexus S は帯域を占有しがちなので抑える方向に，HT-03A は弱気な TCP で帯域を確保することが難しいので，強気の TCP である独自の TCP に切り替える．また高遅延環境では輻輳ウィンドウを十分使いきれていないため，上限値を上げる制御を行った．

7 実験結果

7.1 独自の TCP による制御

図 6 に示すように独自の TCP を用いて制御を行った結果，全ての端末において通信性能が向上することが確認された．異種端末が混在したヘテロ環境で通信を行った場合，Nexus S は高性能な送出量を意図的に抑える方がむしろ自身の性能が向上することが確認された．しかし，HT-03A のスループットは向上したとはいえ低いままであり，帯域をなかなか取得することができていないと言える．

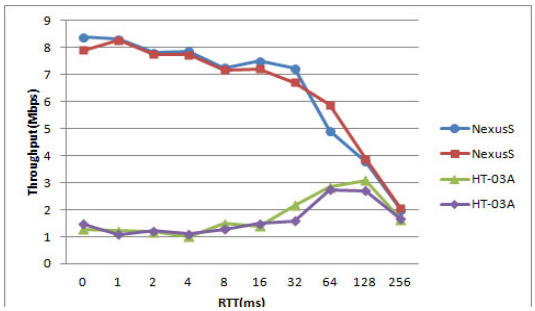

図 6: 4 台の Android 端末の制御時の同時通信性能

7.2 カーネル移殖と独自の TCP による制御

図 6 で HT-03A の通信性能が向上できないのはソースコードや通信解析を行った結果，両者の TCP で親和性がとれていないことが原因として挙げられた．親和性を保つために，Nexus S に HT-03A のカーネルを移植した上で独自の TCP を用いて輻輳ウィンドウサイズの上限値の制御を行った．結果を図 7 に示す．ここで輻輳ウィンドウサイズ上限値の制御方法は表 1 に示す値とほぼ同等であり，HT-03A に関しては default TCP を用いた．

図 7 から親和性が向上し，HT-03A の通信性能も向上することが確認された．また逆方向の移植，HT-03A に Nexus S のカーネルを移植した場合もこの結果とほぼ同等の性能が得られることを確認した．

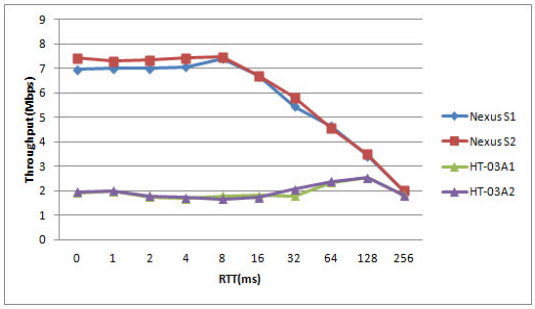

図 7: 4 台の Android 端末の制御時の同時通信性能

8 まとめと今後の課題

本論文では Android 実機にカーネルモニタツールを適用し，通信時の輻輳ウィンドウの値を解析して，TCP のソースコードの改良を行った．複数台の Android 端末を通信させた際，本手法とカーネル移植で各端末のスループットが向上することを確認した．今後は本研究内容を周囲の情報を把握した際に自動的に制御できるようなミドルウェア開発に応用させる．

参考文献

[1] Android:http://www.google.co.jp/mobile/android

[2] Reika Higa, Kosuke Matsubara, Takao Okamawari, Saneyasu Yamaguchi, and Masato Oguchi, "Analytical System Tools for iSCSI Remote Storage Access and Performance Improvement by Optimization with the Tools," In the 3rd IEEE International Symposium on Advanced Networks and Telecommunication Systems (ANTS2009), December 2009.