ネットワークエミュレータを用いた iPhone と Android の通信実験環境構築

2011SE307 横川 史明
指導教員 後藤 邦夫

1 はじめに

近年スマートフォンの普及率は増大している．世界のスマートフォン市場では Android が多くの割合を占めており，日本では iPhone の方が多くの割合を占める．アプリケーションの開発を支援するために，Apple からは iOS シミュレータ [1]，Google からは Android エミュレータ [2] が提供されているが，これらは起動できる環境や数に制限があり，大規模なネットワークの利用を想定したアプリケーションの開発をするには，個人の力だけでは難しい問題となっている．

そこで本研究ではオープンソースのネットワークエミュレータである Common Open Research Emulator(CORE)[3] に Android エミュレータと iOS シミュレータを組み込むことで，通信実験が可能な環境を構築し，ネットワークアプリケーション開発環境を安価で容易に構築することを目的としている．

2 概要

本節では実験の構成要素について説明する．

本研究は Linux 搭載 PC と OSX 搭載の PC の計 2 台を使用して実験する．Linux の搭載された PC 上には Android エミュレータとネットワークエミュレータ CORE を起動し，OSX の搭載された PC 上には iOS シミュレータを起動する．CORE によってエミュレートされた仮想ネットワークへアクセスしようとする場合，Android エミュレータは同じ PC からの接続となり，iOS シミュレータは別の PC からの接続となる．各要素をつなぎあわせた状態は図 1 のように示す．

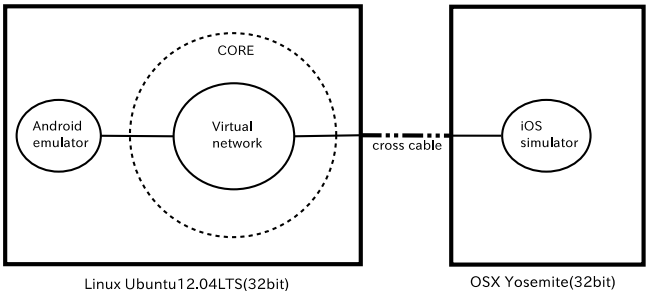

図 1 研究に使用する要素の構成図

2.1 Android エミュレータ

Android エミュレータは QEMU のゲスト OS として起動し，Android エミュレータを起動するとき，コマンドによって QEMU のオプションを変更することで，任意のネットワークインターフェースと接続ができる．本研究ではこの機能を用いて，作成した仮想ネットワークインターフェースと接続することで，同じ PC 上で起動している CORE と通信する．

2.2 iOS シミュレータ

iOS シミュレータは OSX 上でしか動作しないため，Linux で動作する CORE を利用するためには，クロスケーブル等を用いて PC 同士が通信できる環境を構築する必要がある．iOS シミュレータをネットワーク経由で CORE に接続することで，仮想ネットワークに組み込む．

2.3 CORE

CORE には仮想ネットワーク上にある通信路と他のネットワークインターフェースを接続できる RJ45 というノードがある．本研究では Android エミュレータと iOS シミュレータが接続しているネットワークインターフェースを仮想ネットワークの通信路に接続し，CORE を介したネットワークを構築する．また CORE にはノード間の通信路に対してパケットの遅延や損失を設定することが可能であり，ネットワークアプリケーションによる通信実験において，動作確認などを検証する．

3 ネットワーキング

本節では CORE に接続するためのネットワーキングについて説明する．CORE で作成した仮想ネットワークからは，ホスト上で見えているインターフェースとしか接続ができない．その解決策として，仮想のインターフェースを作成し，ネットワークをつなぐ仲介役として利用する．

3.1 iOS シミュレータからの接続

iOS シミュレータは OSX 用アプリケーションであるため，本研究において Linux 上で動作している CORE との通信には，それぞれ PC 同士のネットワークを構成する必要がある．

iOS シミュレータは NAT によって自動でホストが接続しているネットワークを利用しており，iOS シミュレータではネットワーク設定を変更することはできない．そこで，クロスケーブルによる接続でホスト同士のネットワークを構成し，ホスト OS のネットワーク設定を仮想ネットワークに属すようにアドレスを設定することでネットワークを構築する．

3.2 Android エミュレータからの接続

Android エミュレータは iOS シミュレータと同じ NAT によって接続されるが，QEMU で動作しているため，起動時にオプションによる接続先の変更ができる．また，起動する image ファイルを書き換えることでアドレスの変更が可能である．

本研究では Android エミュレータとネットワークエミュレータ CORE を同じ OS 上で動作させる．通常のように GUI から起動してしまうと，Android エミュレータは自動でホスト OS が利用しているネットワークに接続してしまうため，任意の接続先に指定することができない．そのため，コマンドによってエミュレータを起動させ，接続先を指定する必要がある．CORE と接続するために，アドレスを仮想ネットワークに属すように変更した image ファイルを作成する．そして，tunctl を用いて仮想ネットワークインターフェイスである tap を作成する．接続先が仮想インターフェース tap の Android エミュレータを起動する．tap と仮想ネットワークを接続することでネットワークを構成する．

上記の手順により，CORE を介したネットワークを構成する．通信実験で使用するネットワークは図 2 で示す．

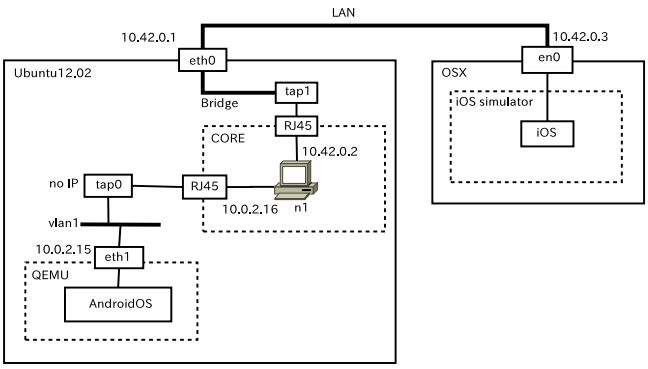

図 2 通信実験用のネットワーク図

4 実験

本節では CORE を介した iPhone と Android 間で通信が行えることをチャットアプリケーションを用いて確認し，仮想ネットワークの通信環境を評価する．

iOS と Android の開発言語は異なるため，クロスプラットフォームのアプリケーションの開発は困難である．しかし，ブラウザを利用した Web アプリケーションを使用することで，同じ開発言語のアプリケーションを各ブラウザから利用できる．

本研究では，サーバとクライアント間の通信を行うチャットアプリケーションを作成するために，Node.js を用いて，サーバ側となる JavaScript ファイルと，クライアント側となる html ファイルを作成した．

作成したチャットアプリケーションの処理は次のようになる．

1. クライアントは http サーバにアクセスする．
2. サーバはチャットを行うための html ファイルを返す．
3. クライアントはサーバにメッセージを送信する．
4. サーバはメッセージを受信すると，サーバに接続しているクライアント全てにメッセージを送信する．
5. クライアントはメッセージを受け取り，表示する

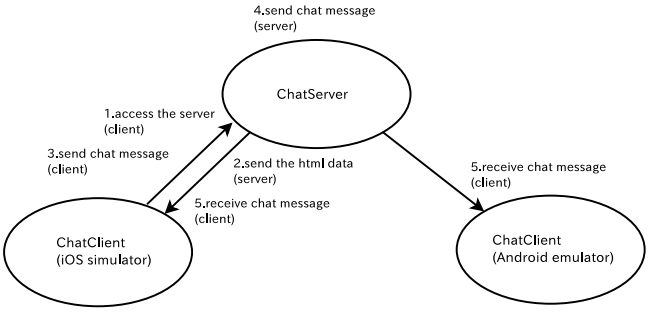

図 3 チャットアプリケーションの処理

チャットアプリケーションを用いてスマートフォン間でメッセージのやり取りができることを確認した．

結果，仮想ネットワークを介したソケット通信によるサーバとクライアント間のリアルタイムな双方向通信が可能であることが分かり，CORE を介して，Android エミュレータと iOS シミュレータが双方向通信できるネットワークを構成されたと言える．

また，CORE の機能を用いて，パケットの遅延や損失を通信路に設定し，それを ping を用いて，設定が反映されていることを確認した．これにより，ネットワークエミュレータを用いて構成したネットワークは，アプリケーションの通信実験環境として有用性があると言える．

5 おわりに

本研究によって，CORE を介した Android エミュレータと iOS シミュレータのネットワーク環境を安価で容易に構築する手法を提案した．本研究は 1 対 1 の小規模なネットワークによる実験を行ったが，個人でも大規模なネットワークを構築できる可能性を示した．今後は実際のネットワークをエミュレートして，ネットワークアプリケーションの通信実験を行う必要があると考える．

参考文献

[1] Apple: Apple Developer (accessed Jan. 2015). https://developer.apple.com/jp/.

[2] Google: Android Developers (accessed Jan. 2015). http://developer.android.com/index.html.

[3] U.S. Naval Research Laboratory Networks and Communication Systems Branch: Common Open Research Emulator (accessed Jan. 2015). http://www.nrl.navy.mil/itd/ncs/products/core.