6U-01

Swiftに対応したNTMobileフレームワークの設計

鈴木 聡人 [†1]　山田 貴之 [†2]　鈴木 秀和 [†2]　内藤 克浩 [†3]　渡邊 晃 [†2]
[†1] 名城大学理工学部　[†2] 名城大学大学院理工学研究科 [†3] 愛知工業大学情報科学部

1　はじめに

筆者らはIPv4/IPv6混在環境において通信接続性の確保と移動透過性を同時に実現するNTMobile（Network Traversal with Mobility）を提案してきた[1]．これまでにNTMobileの機能をフレームワークとして整備し，モバイルアプリケーションに組み込むことでユーザ空間だけでNTMobileを実現している[2]．しかし，現状のフレームワークはJavaとObjective-Cには対応しているものの，今後普及が予想されるSwiftには対応していない．

本稿では，NTMobileフレームワークを新たにSwiftに対応させるための設計方式について述べる．

2　NTMobileフレームワーク

NTMobileフレームワーク（以後NTMフレームワーク）は，図1に示すようにC言語で開発されたNTMobileライブラリと，NTMobileライブラリを呼び出すNTMobileソケットAPI（以後NTMソケットAPI）から構成されている．アプリケーション開発者はNTMソケットAPIを利用することにより，開発アプリケーションがNTMobileの枠組みで動作することができる．

従来のNTMフレームワークにはJavaおよびObjective-CのNTMソケットAPIが実装されている．このため，Android OSおよびiOSの各アプリケーションの開発をサポートしている．しかし，近年のiOS向けアプリケーション開発で多用されつつあるSwiftには対応していない．

3　NTMフレームワークのSwift対応

3.1　概要

本稿では，従来のJavaおよびObjective-CのNTMソケットAPIと同様の仕組みにより，新たにSwift対応のNTMソケットAPIをNTMフレームワークに定義する．これにより，Swiftで開発した通信アプリケーションにNTMobileを適用する．

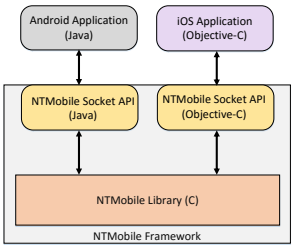

図1　従来のNTMobileフレームワークの構成

3.2　設計方針

アプリケーション開発者がNTMフレームワークを意識することなく利用するには，NTMソケットAPIを簡単に利用できる必要がある．従来のNTMソケットAPIは標準ソケットAPIのC関数やJavaメソッドの名前に接頭辞を付けたり，同名で設計していた．しかし，Swiftには標準ソケットAPIが存在しないため，Swiftに特化した設計を行う必要がある．

Swiftにおけるソケット通信の実現手法としては，WebSocketを利用する方法とObjective-CまたはC言語の標準ソケットを利用する方法がある．WebSocketは接続先サーバのURLを指定して接続するクライアント/サーバ型通信を前提としている．これに対し，NTMobileではクライアント間を直接接続するエンドツーエンド通信を想定しているため，WebSocketの利用は不向きである．そのため，Objective-CまたはC言語の標準ソケットAPIに類似した仕様でSwift用NTMソケットAPIを設計する必要があるが，NTMobileライブラリがC言語で実装されていることを考慮すると，SwiftからObjective-Cの関数を介した設計より，Swiftから直接C言語の関数を呼び出す設計の方がオーバヘッドが少ないと考えられる．

以上により，C言語の標準ソケット関数と同じ使い方ができるようにSwift対応のNTMソケットAPIを設計する．

3.3　Swift対応のNTMソケットAPI

図2にSwift対応のNTMフレームワークの内部構成を示す．C言語の標準ソケットAPIの関数名に準拠したSwift関数を定義し，関数内でNTMobileライブラリのC関数を呼び出す．これにより，Swift関数を呼び出すことで端末間の通信開始時のシグナリング処理や，データ通信の暗号化処理およびカプセル化処理な

Design of NTMobile Framework for Supporting Swift
Soto Suzuki[†1], Takayuki Yamada[†2], Hidekazu Suzuki[†2], Katsuhiro Naito[†3] and Akira Watanabe[†2]
[†1] Faculty of Science and Technology, Meijo University
[†2] Graduate School of Science and Technology, Meijo University
[†3] Faculty of Information Science, Aichi Institute of Technology

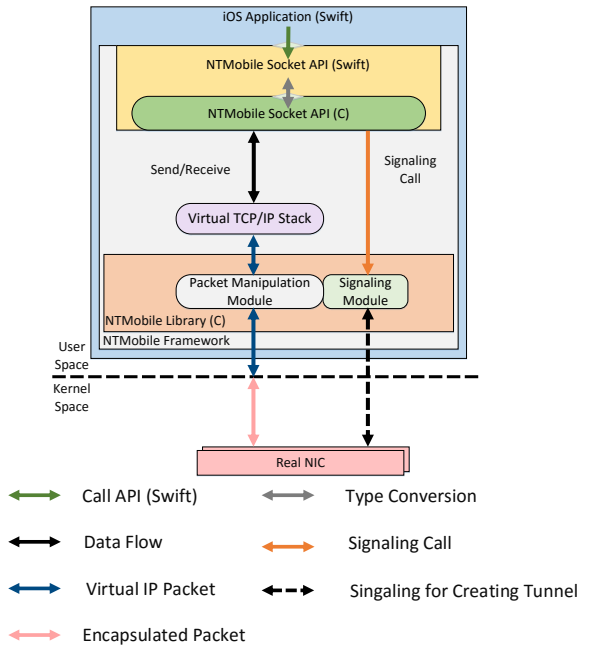

図 2　Swift 対応の NTMobile フレームワークの内部構成

ど，NTMobile 特有の処理を実行することができる．また，Swift と C 言語間で必要となる型およびポインタの変換に関する処理は Swift 関数内で行う．これにより，Swift アプリケーション開発者に C 言語のライブラリを利用する際に必要となる変換コードを意識させない設計ができる．

4　実装と評価

従来の Objective-C 対応の NTM ソケット API および今回検討した Swift 対応の NTM ソケット API を利用して開発した iOS 用通信アプリケーションにおいて通信性能上の違いが発生するか明らかにするために，本稿では基礎的な検証として C 言語で開発されている iperf3[3] の関数を Objective-C および Swift の関数から呼び出してスループットを測定する実験を行った．図 3 に測定環境を，表 1 に各機器の仕様を示す．クライアントには iPhone5 を，サーバには Ubuntu を使用し，クライアントからサーバへ 30 秒間の TCP 通信および UDP 通信を 30 回行い，スループットの平均値を求めた．

図 4 に測定結果を示す．Objective-C アプリケーションおよび Swift アプリケーションの TCP 通信のスループットはそれぞれ 80.47[Mbps]，81.97[Mbps]，また UDP 通信のスループットはそれぞれ 84.55[Mbps]，84.58[Mbps] となり，結果として Swift アプリケーションと Objective-C アプリケーションの通信性能に大きな違いは生じなかった．この理由として各アプリケーションでは，Objective-C および Swift 特有の処理は UI および C 言語の標準ソケット API を呼び出す処理だけであり，主となる通信処理は両方とも C 言語で行われて

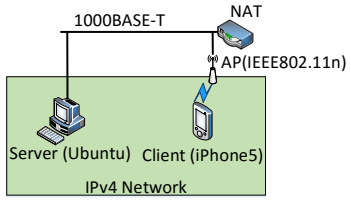

図 3　測定環境

表 1　各機器の仕様

	Client	Server
Hardware	iPhone5	Intel NUC DN2820FYK
OS	iOS8.4.1	Ubuntu14.04
Kernel	arm-apple-darwin14.0.0	Linux 3.13
CPU	Apple A6 1.3GHz	Intel Celeron CPU N2830 2.16GHz
Memory	1GB	4GB

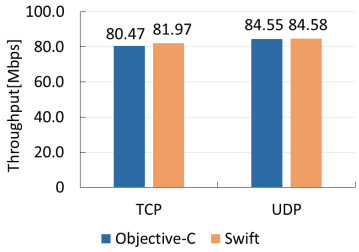

図 4　スループット測定結果

いるためだと考えられる．NTM フレームワークを利用した場合，端末間の通信開始時のシグナリング処理やデータ通信の暗号化処理およびカプセル化処理は全て C 言語で実装された NTMobile ライブラリ側で行われる．従って，本稿で設計した Swift 対応の NTM ソケット API を利用して開発したアプリケーションも，従来の Objective-C 対応の NTM ソケット API を利用して開発したアプリケーションと同等のスループットが見込める．

5　まとめ

本稿では，Swift に対応した NTMobile フレームワークの設計について述べた．iperf3 を利用した基礎的検証実験を実施した．結果，従来の NTM フレームワークと同等のスループットが見込める Swift 対応アプリケーションが設計できることを確認した．

参考文献

[1] 上酔尾, 他：情処学論, Vol. 54, No. 10, pp.2288-2299 (2013).
[2] Naito, K., et al.: Proc. of IEEE CCNC 2014, p.276-281 (2014).
[3] iperf3：https://github.com/esnet/iperf