Android センタープッシュサーバの実装

ソリューション本部　社会システム部

中尾　浩一

1. はじめに

高機能で、アプリケーションの開発が比較的容易なスマートフォンは、今やビジネスの世界では、単なる通話機能の使用だけに留まらず、様々な分野で業務端末としての活用が進んでいる。

このような業務システムにおいては、遠隔にあるスマートフォンとの情報交換や機器の制御、スマートフォン・アプリケーションとの連携など、通信ネットワークを前提とした機能向上のニーズが増えつつある。

携帯電話とメッセージ通信を行う方法としては、SMS センタープッシュサービスの利用が考えられるが、スマートフォンの普及が急激に進んでいることもあり、通信キャリアによって対応がまちまちだったり、利用が特定の機種に限定されていたり、また、プリインストールアプリでしかメッセージを受信できないなど、現状は十分な活用が難しい状況である。

このような背景を踏まえ、今回、スマートフォンとメッセージ通信を行う方法として、「NAT 越え」に関する技術である STUN（Simple Traversal of UDP through NATs）を利用した Android センタープッシュサーバの構築を行った。

本稿では、その概要を紹介する。

2. SMS センタープッシュサービス

2. 1 概要

SMS センタープッシュサービスは、携帯電話の音声網サービスの一つである SMS（Short Messaging Service）を活用することで、携帯電話アプリケーションに向けてメッセージの送信を行うサービスである。

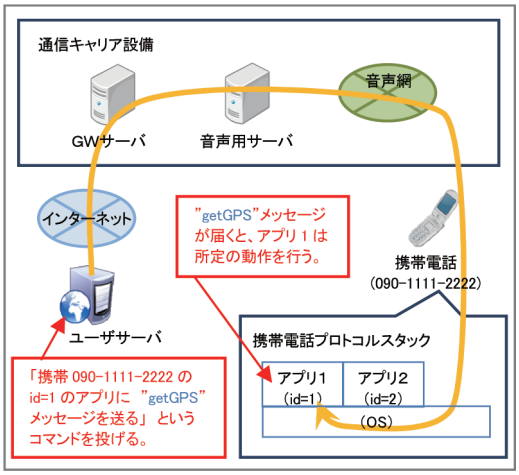

図 1　SMS センタープッシュサービス

例えばドコモでは、2011 年 4 月より、スマートフォン・携帯電話・通信モジュール内蔵の車両・建機・パソコンなどの端末に、制御文やテキストメッセージの SMS を送信できるサービスを開始しており、メールアドレスを管理せずとも、電話番号を指定するだけで、端末を持つ社員宛にメッセージを送信したり、遠隔からスマートフォンのセキュリティロックをかけるなどの制御ができるサービスを提供している。

図 1 に SMS センタープッシュサービスの仕組みを示す。

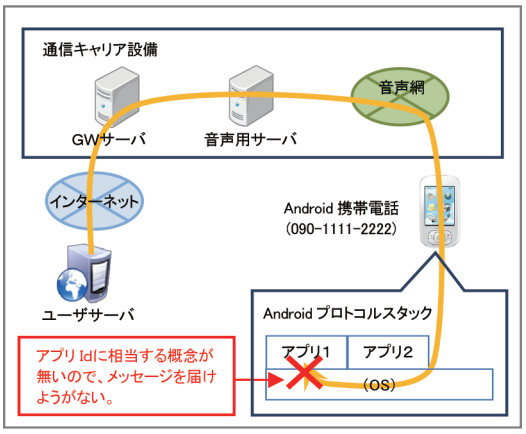

図 2　SMS センタープッシュサービスが NG の例

2. 2　SMS センタープッシュサービスの問題点

はじめに述べた通り、SMS センタープッシュサービスは、現状では、スマートフォン上で稼働するユーザアプリケーションからは利用しづらい状況にある。

図 2 は、Android 携帯電話に SMS を送信したものの、送信先としてユーザが開発した個別アプリケーションを指定できないために、メッセージを届けることができない状況を示している。

3.　STUN の概要

既存の SMS センタープッシュサービスが不十分な提供状況にあるため、個々のスマートフォンを電話番号で識別することは不可能である。そこで、電話番号以外で、端末を識別する手段としては IP アドレスを指定する方法が考えられるが、IP アドレスにはアドレス枯渇の問題があり、実際、ほとんどのスマートフォンでは、IP アドレスは動的に割り当てられ、固定化されていない。つまり、スマートフォンは通信キャリアのルータ(NAT)の内側(ローカルネットワーク)に存在しており、ルータの外側からこれらのスマートフォンにメッセージを送信するには、必然的に「NAT 越え」を

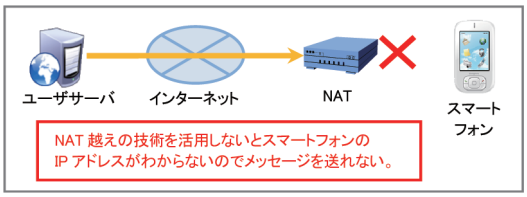

図 3　NAT 越え

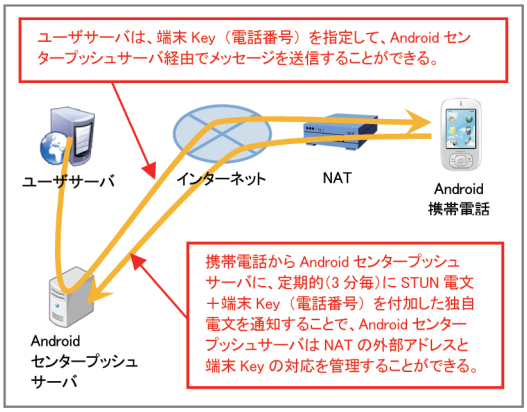

図 4　Android センタープッシュサーバの概要

実現しなければならない(図 3)。

この「NAT 越え」の技術の一つが STUN である。STUN の基本的な仕組みは UDP Hole Punching の原理に基づいている。考え方としては、内部(スマートフォン)から外向き(STUN サーバ)に通信を行うことでルータのマッピングテーブルにエントリが記録されるので、そのエントリを利用して外部から内部への通信を行うというものである。代表的な例としては、Skype は STUN 技術に基づいていると言われている。STUN プロトコルは、RFC 3489 に定められており、このプロトコルにより、リモートホスト(スマートフォン)向けの UDP(User Datagram Protocol)通信用に NAT が割り当てたグローバル IP アドレスとポート番号を知ることができる。

この技術を利用して、Android 携帯電話にメッセージを送信する機能を提供するのが Android センター

プッシュサーバである。図4に Android センタープッシュサーバの概要を示す。NAT の外にいるユーザサーバ(業務システム)は、Android センタープッシュサーバを経由することで、Android 携帯電話へメッセージを送信することができるようになる。

4. Android センタープッシュサーバの実装

4. 1 概要

今回は、実験として、ユーザサーバからは電話番号を指定することで Android 携帯電話にメッセージを送信し、Android 携帯電話からは自身の GPS 座標を返信させる仕組みを実装した。図 5 に Android センタープッシュサーバの構成を示す。

センタープッシュサーバは、Android 携帯電話の端末 Key(電話番号)と NAT 外部アドレスの対応を管理するためのデータベースを保持しており、内部的には次の 3 つのサーバから構成されている。

(1) STUN サーバ

UDP プロトコルにより、Android 端末(以下、端末と記述)と電文のやり取りを行う。次の 2 つの機能を有する。

① 端末と STUN 電文をやり取りする機能

② ユーザサーバからのメッセージをメッセージ転送サーバ経由で受け取り、端末に送信する機能

(2) 対端末 UDP データ受信サーバ

端末からの NAT アドレス通知電文、および、ユーザメッセージ応答(端末がメッセージを受信したという応答)を受信する。

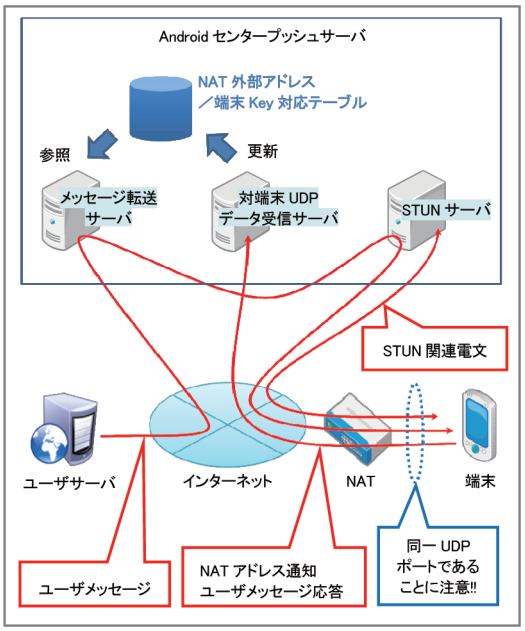

図 5 Android センタープッシュサーバの構成

NAT アドレス通知電文受信時は、端末 Key/NAT 外部アドレス対応管理テーブルを更新し、ユーザメッセージ応答受信時には、メッセージ送信状態管理テーブルを更新する。

(3) メッセージ転送サーバ

ユーザサーバからのメッセージを http プロトコルで受け付け、STUN サーバに転送するとともに、メッセージ送信のタイムアウト管理を行う。また、ユーザサーバからのメッセージ送信状況確認の対応を行う。

表 1 に本システムの電文の一覧を示す。

なお、端末への送信処理を STUN サーバに集約させているのは、NAT が Symmetric の場合、NAT は送信元のポート番号まで記憶しており、異なるポートからの電文は通さないためである。

表 1　電文一覧

No.	電文名	M	U	S	T	説明
1	STUN Shared Secret Request			◀	●	＜未使用＞　RFC3489 で規定された NAT の外側のアドレスを知るためのコマンドを利用するための認証コマンド。詳細は RFC3489 を参照。
2	STUN Shared Secret Response			●	▶	＜未使用＞　上記のレスポンス
3	STUN Binding Request			◀	●	RFC3489 で規定された NAT の外側のアドレスを知るためのコマンド。 詳細は RFC3489 を参照。 NAT テーブルは一般的に 180 秒で消滅することが多いため、180 秒に一回は当コマンドを発行し、アドレスをチェックする。ただし、180 秒は「標準化された規定値」ではないため、外部から変更可能とする。
4	STUN Binding Response			●	▶	上記のレスポンス
5	NAT アドレス通知		◀	―	●	NAT 外部アドレスと、自端末の Key 情報を通知する。 センタープッシュサーバでは、この情報をテーブルに記録する。 当メッセージは、STUN Binding Response 受信後必ず通知する。
6	User Message	●	▶	●	▶	ユーザサーバからのプッシュメッセージ。 ユーザサーバは端末 Key と任意のメッセージを指定し、センタープッシュサーバに通知する。 センタープッシュサーバは、NAT 外部アドレス／端末 Key 対応テーブルより、NAT 外部アドレスを引き、そこに対し、メッセージを STUN サーバ経由で端末に送る。
7	User Message Response		◀	―	●	上記のレスポンス。 対端末 UDP データ受信サーバで、NAT 外部アドレス／端末 Key 対応テーブルを更新し、処理が完結する。

※凡例　　M：メッセージ転送サーバ　　U：対端末 UDP データ受信サーバ　　S ： STUN サーバ　　T ： 端末

表 2　電文 API の一覧

No.	API 名	説明
1	ユーザメッセージ送信　(sendMsg)	ユーザサーバからメッセージ転送サーバへのプッシュメッセージ。 ユーザサーバは端末 Key(電話番号)と任意のメッセージを指定して、メッセージ転送サーバに通知する。 メッセージ転送サーバは、受け取ったメッセージを STUN サーバに転送する。 なお、メッセージ転送サーバは、端末からの応答を待たずに、送信行為が行えたかどうかのレスポンスを返す。
2	送信メッセージ確認　(confirmMsg)	直近の sendMsg の実行状態を返す。 メッセージが端末に届いたかどうかの状況を確認できる。

表 3　ユーザメッセージ送信(sendMsg)の Request フォーマット

種別	Name	Value	意味・備考
ヘッダ	POST　HTTP/1.1		メソッド
	HOST		ホスト
	User-Agent	test	独自 User-Agent
	Content-Type	application/x-www-form-urlencoded	固定
	Content-Length	999	BODY 長
ボディ	Mdn	09011112222	ハイフォンなし電話番号 カンマ区切りで複数指定可
	Msg	xxx	任意の文字列

表 4　ユーザメッセージ送信(sendMsg)の Responseフォーマット

種別	Name	Value	意味・備考
ヘッダ	HTTP/1.1 [Status Code and Message]		OK:200 ・・・　※http 送信成功
	Date	xxx	処理日時
	Server	xxx	サーバ名
	Connection	close	固定
	Content-Type	text/xml	固定
	Content-Length	999	BODY 長
ボディ	<response> <result> [result code] </result> </response>		[result code] → コマンドの実行結果 ・OK :成功 ・NG :失敗

表 5　送信メッセージ確認(confirmMsg)の Request フォーマット

種別	Name	Value	意味・備考
ヘッダ	POST　HTTP/1.1		メソッド
	HOST		ホスト
	User-Agent	test	独自 User-Agent
	Content-Type	application/x-www-form-urlencoded	固定
	Content-Length	999	BODY 長
ボディ	Mdn	09011112222	ハイフォンなし電話番号 カンマ区切りで複数指定可

表 6　送信メッセージ確認(confirmMsg)の Response フォーマット

種別	Name	Value	意味・備考
ヘッダ	HTTP/1.1 [Status Code and Message]		OK:200 ・・・　※http 送信成功
	Date	xxx	処理日時
	Server	xxx	サーバ名
	Connection	close	固定
	Content-Type	text/xml	固定
	Content-Length	999	BODY 長
ボディ	<response> <result> [status code] </result> <status mdn="09011112222" date="2012/05/18 12:12:12"> [status code] </status> </response>		[result code] → コマンドの実行結果 ・OK :成功 ・NG :失敗 [status attribute] ・mdn:携帯電話番号 ・date:ステータス更新時間 [status code] ・OK :成功 ・NG :失敗 ・SENDING:送信中 ・NONE:データなし

4．2　ユーザサーバからの電文 API

先述の通り、ユーザメッセージはメッセージ転送サーバが一旦受け付けた後、STUN サーバに転送され、端末に送信される。表 2 にユーザサーバからメッセージ転送サーバに送信される電文 API の一覧を示す（今回の実験用に仕様を定義した）。また、表 3〜4 にユーザメッセージ送信（sendMsg）の Request と Response のフォーマットを、表 5〜6 に送信メッセージ確認（confirmMsg）の Request と Response のフォーマットを示す。

sendMsg の引数には電話番号と任意の文字列（例えば、伝言文や GPS 位置要求等のコマンド）を与えることができる。また、confirmMsg により、送信したメッセージが端末に届いたかどうかの確認を行うことができる。

4．3　実験結果

実験では、ユーザサーバから送信した伝言文を Android 端末の画面に表示する機能と、Android 端末の位置問い合わせとして、GPS 測位による現在位置をユーザサーバに返信する機能を検証した。今回の実験のように、電文 API は、ユーザが自由に定義できるため、業務仕様に応じて、これを工夫・拡張することで、Android 端末との様々な連携機能を実現できることが確認できた。

5. おわりに

スマートフォンは、今後も急速なスピードで普及し、高性能化が進んでいくであろう。これに伴い、業務システムとスマートフォンの連携ニーズは益々高まってくるはずである。本稿で紹介した STUN 技術による NAT 越えは、原則、通信キャリアやスマートフォンの機種を問わない連携技術であり、その柔軟性は多様なシーンでビジネスをサポートすることが可能である。本稿がモバイル連携システムの構築の一助となれば幸いである。

＜謝辞＞

本システムの実装に際しては、株式会社 h2 ワークスの米澤真一氏に多大なるご支援を頂いた。ここに感謝の意を表します。

＜参考文献＞

1)　「法人ソリューション向け SMS センタープッシュサービス基盤の開発」（ドコモ R&D（研究開発）の広報誌 テクニカル・ジャーナル VOL19 NO.3、2011 年 10 月）
http://www.nttdocomo.co.jp/corporate/technology/rd/technical_journal/bn/vol19_3/012.html

2)　「次世代スマートフォン向けサービス特集／Android 向けアプリ共通基盤開発」（ドコモ R&D（研究開発）の広報誌 テクニカル・ジャーナル VOL20 NO.1、2012 年4月）
http://www.nttdocomo.co.jp/corporate/technology/rd/technical_journal/bn/vol20_1/012.html

3)　「RFC3489」
http://tools.ietf.org/html/rfc3489

4)　「PART2　Skype、その通信の仕組み」（IT media エンタープライズ、特集：Skype は企業 IP 電話を変えるか、2005 年5月 30 日）
http://www.itmedia.co.jp/enterprise/articles/0505/30/news070.html