Twitter как транспорт в информационных системах

Намиот Д.Е.

***Аннотация*—В статье рассматриваются вопросы использования механизмов Twitter для создания информационных систем. В свете перехода мобильных абонентов от SMS к независимым приложениям для обмена сообщениями, как мы можем использовать Twitter для построения информационных систем? Описывается модель, которая позволяет использовать Twitter вместо традиционных в мобильных сервисах систем на базе SMS.**

***Ключевые слова*—Twitter, SMS, клиент-сервер, бот.**

I. Введение

Традиционно, в мобильных сервисах часто использовались (используются и по сей день) информационные системы на базе SMS. Это приложения, которые позволяют абоненту послать оформленное некоторым специальным образом сообщение на заданный сервисный номер. Результатом отправки такого сообщение может быть оформление платежа, резервирование товаров, запрос к некоторой информационной системе, результаты которого будут высланы ответным сообщением и т.д. Сервисный номер может быть, естественно, произвольным. Сам формат сообщения может меняться от сервиса к сервису. Но общая идеология (модель) построения остается одинаковой. С точки зрения стандартизации, этот подход достаточно отработан. Для приема и отправки SMS используется стандартный протокол SMPP [1] или какие-либо надстройки над ним. По аналогичной схеме работают сервисные системы на базе MMS. Приложение (сервис) читает полученное сообщение – это обычный текст, дополнительные атрибуты которого позволяют узнать об отправителе, времени, языке и т.д. Текст может быть разобран и обработан в соответствии с реализуемым алгоритмом. Алгоритм разбора [2], как правило, не представляет практической сложности, поскольку выразительные средства такого рода “языка запроса” ограничены разумно-допустимой сложностью набора текста на мобильном устройстве. Атрибуты сообщения используются для формирования отклика. Вместе с тем, необходимо отметить, что такого рода сервис по определению является операторским. Он привязан к некоторому номеру. Это может быть специальный сервисный номер (чаще всего - так называемый короткий номер) или обычный телефонный номер. Технически, в качестве устройства приема/отправки SMS может выступать GSM модем [3]. В простейшем случае в такой роли может выступать мобильный телефон, программный доступ к которому может осуществляться с помощью AT-команд [4]. Но в любом случае, присутствие телефонного номера (и, следовательно, телекоммуникационного оператора необходимо).

С другой стороны, все анализы современных тенденций развития телекоммуникационной отрасли показывают постоянную миграцию пользователей (абонентов) в сторону альтернативных программ обмена сообщениями [5].

Соответственно, в данной статье рассматривается модель, которая позволяет строить информационные системы на базе Twitter. Именно Twitter (Twitter API) играет здесь роль транспортного уровня.

II. Сервер сообщений на базе Twitter

Пользователи Twitter могут обмениваться сообщениями двумя возможными способами. Они могут упомянуть в своем сообщении другого пользователя (так называемые mentions), а также отправить другому пользователю сообщение (direct messages).

Идея сервера сообщений на базе Twitter состоит в создании приложения, которое привязано к некоторому аккаунту (регистрационному имени или учетной записи) в Twitter и отслеживает все сообщения (статусы), обращающиеся к данному аккаунту.

Например, для сервиса T411 for Twitter [6] в качестве такого сервисного аккаунта выбрано имя *@t411*. Это означает, что пользователи Twitter могут послать сервисный запрос, просто упомянув в своем статусе *@t411* или послав сообщение пользователю *@t411*. Например, типичный сервисный запрос выглядит таким образом:

@t411 t GOOG

Это статус (сообщение), которое пользователь Twitter может опубликовать в своей ленте сообщений для запроса котировок акций Google (устройство собственно сервиса будет описано ниже).

Для случая сообщений (direct messages) все выглядит аналогично. Основное видимое отличие состоит в том, что прямые сообщения, естественно, не видны в публичной ленте (timeline). Иными словами, другие

Статья получена 25 декабря 2013.

Д.Е. Намиот – старший научный сотрудник факультета ВМК МГУ им. М.В. Ломоносова (e-mail: dnamiot@ gmail.com).

пользователи Twitter не будут видеть ни самого запроса, ни отклика от сервиса.

Таким образом, регистрационная запись пользователя Twiitter выступает здесь в качестве сервисного номера SMS. А вместо отправки SMS в некотором формализованном формате пользователь использует статус в Twitter и/или сообщение в Twitter. Сервис сообщений делает далее то же самое, что и информационные сервисы на базе SMS. А именно – разбирает текст сообщения и выполняет предписанные действия. Сообщение также имеет отправителя. В данном случае – это регистрационная запись (аккаунт) от имени которого запрос был опубликован (от имени которого было отправлено сообщение). Этот адрес может быть использован для формирования отклика. Иными словами, отклик будет опубликован в публичной ленте сервисного аккаунта. Например, если пользователь *@abava* опубликовал в своей ленте следующий статус:

@t411 t GOOG

то в ответ в ленте пользователя *@t411* будет опубликовано:

@abava GOOG: 1100.62 http://ichart.finance.yahoo.com/t?s=GOOG

(в данном случае это текущая котировка и ссылка на график изменения цены).

Иными словами, ответ отправителю будет доставлен в форме упоминания его аккаунта (mentions) в публичном статусе.

Для сообщений это будет выглядеть аналогично, только сами статусы будут не видны остальным пользователям Twitter.

Приложение реализовано в виде Java сервлета, работающего в среде Google App Engine [7]. Среда поддерживает сервис планировщика (cron process). Это позволяет вызывать сервлет периодически (например, раз в секунду). Сервлет обращается к Twitter API, читает новые упоминания (сообщения) для регистрационной записи в Twitter, обрабатывает их и отправляет ответы.

Технически, это устроено несколько сложнее, поскольку необходимо соблюдать ограничения облачной среды на время обработки запроса. Поэтому приложение реализует набор очередей. Из одной из очередей считываются запросы, в другую помещаются готовые отклики, а процессы чтения и обработки выполняются в разных циклах для фиксированного количества элементов, чтобы обеспечить предсказуемое время выполнения операций.

В такой форме обращение к серверу сообщений может быть легко встроено в сторонние сервисы. Посылка запроса – это отправка (публикация) статуса в Twitter. При необходимости, можно также программно читать и отклики системы. Это публичные статусы (сообщения), упоминающие регистрационную запись отправителя, автором которых выступает сервисный аккаунт в Twitter. Например, в указанном выше примере можно программно читать упоминания *@abava*, автором которых выступает *@t411*.

Это краткое описание сервиса T411. Название его было выбрано исходя из известного телефонного справочного сервиса в США (411 service [8])

III. Информационные сервисы

Помимо технической организации транспортного уровня, сервер сообщений T411 предлагает также и модель организации информационных сервисов. Здесь мы следуем модели, которая многократно применялась (применяется) в SMS сервисах. Служебный SMS, отправляемый на сервисный номер обычно выглядит как некоторый цифровой код и, возможно, какой-то набор дополнительных параметров [9].

Сообразно этому, T411 предполагает, что сервисные твиты имеют стандартную форму:

Код_запроса дополнительные_параметры

Код запроса есть некоторая уникальная алфавитно-цифровая последовательность. При создании нового сервиса, пользователь в первую очередь выбирает и регистрирует именно этот код. Можно сказать, что это некоторый аналог домена. В запросе, после кода через один или несколько пробелов может быть представлен произвольный текст. Это – дополнительные параметры. Их интерпретация зависит от конкретного сервиса.

В приведенном выше примере *t* – это сервисный код, *GOOG* – дополнительные параметры.

Сервисный код, таким образом, привязан к конкретному пользователю системы (владельцу сервиса). После регистрации сервисного кода, владелец сервиса должен задать возможную реакцию. Здесь возможны две альтернативы.

Во-первых, можно просто задать некоторый текст. Естественно, он может включать и ссылки (URL) и не может превосходить 140 знаков (размер твита). Владелец сервиса может менять этот текст по своему усмотрению в произвольное время.

Другая возможность состоит в создании динамических сервисов. К сервисному коду можно привязать некоторый URL (CGI скрипт). Этот скрипт может располагаться на произвольном узле сети Интернет. Например, на веб-сайте разработчика. Таким образом, сервер сообщений не имеет доступа к его содержанию. При получении твита (сообщения) с соответствующим сервисным кодом, сервер выполняет HTTP GET запрос к указанному скрипту. Результат этого запроса (текст) и будет являться откликом.

При описании собственного CGI-скрипта можно указать на необходимость передачи ему информации об оригинальном твите (запросе). Это достигается путем задания параметров GET-запроса. Например, следующий URL:

http://your_host/script?t=text&u=from

уведомляет систему, что необходимо обратиться с скрипту по адресу *http://your_host/script*. При этом будут переданы два параметра: *u* – в качестве значения будет подставлен исходный текст твита и *t*, в качестве значения которого будет использовано имя отправителя. Например, для указанного выше модельного сервиса, который вычисляет котировки акций, был указан следующий скрипт:

http://linkstore.ru/t411/stock.jsp?t =text

а сам скрипт выглядит следующим образом (использовались компоненты из Coldtags suite [10]):

```
<%@ page contentType="text/plain; charset=utf-8" %>
<%@ taglib uri="taglib27.tld" prefix="get" %>

<%
String t = request.getParameter("t");
if (t==null)
{ out.println("unknown");
return; }

// the pattern is: t <space> stock_symbol

int i = t.indexOf(" ");

if (i<=0)
{ out.println(t+"?? could not get ticket");
return; }

t = t.substring(i+1).trim();
%>

<get:Quote symbol="<%=t.toUpperCase()%>" id="A" />

<%=A.get(0)+": "+A.get(1)+" "+A.get(9)%>
```

Это позволяет создавать динамические сервисы. При этом сам код сервисов остается полностью под управлением их создателей (разработчиков) и недоступен серверу сообщений. Естественно, разработчики могут полностью менять код по своему усмотрению в любое время. Владельцу сервиса также доступна информация по статистике использования его сервиса (сервисов).

Типичное применение – это, например, модели и сервисы, связанные со Smart Cities [11].

IV. Дальнейшее развитие

Говоря о дальнейшем развитии подобного рода сервисов, можно отметить следующее.

Во-первых, статусы в Twitter могут содержать информацию о гео-позиционировании. В Twitter сообщение с гео-тегами есть своеобразная форма того, что называется check-in в других системах [12]. Соответственно этому, сервисный статус (твит с запросом) также может иметь гео-теги. Для случая, когда реакция (сервис) задается с помощью внешнего CGI-скрипта, эти метаданные вполне могут быть учтены при обработке запроса. Сервер сообщений будет выделять координаты из статуса, а затем передавать их на обработку внешнему скрипту (так, как сейчас это можно сделать с текстом твита и именем автора запроса). Это даст возможность строить гео-сервисы на базе сервера сообщений. Отметим, что в случае SMS организовать такого рода сервис без участия оператора было бы невозможно. У SMS сервера нет информации о местоположении автора SMS. Единственная возможность там – это вставлять информацию о местоположении непосредственно в текст сообщения. Например, так как это делают Geo Messages [13].

Другим очевидным улучшением может быть использование атрибута in_reply_to в статусах ответов, что позволит выстраивать цепочки сообщений.

Другая полезная функция, которую можно реализовать непосредственно в сервере сообщений – это ограничение пользователей, которые могут общаться с конкретным сервисом. Тогда владелец сервиса сможет организовывать что-то вроде сервисов подписок, самостоятельно определяя, кто может использовать его сервис. Это может быть полезно, например, для корпоративных приложений. В настоящее время такой ограниченный сервис можно организовать, проверяя пользователей в собственном CGI-скрипте.

И, конечно, остается большой простор в развитии средств и инструментов статистики для сервера сообщений.

Дальнейшее развитие возможно в рамках работ, выполняемых в лаборатории ОИТ ВМК МГУ [14].

V. Заключение

В работе рассматривается вопрос организации информационных сервисов на базе Twitter. В статье рассматривается модель сервиса, который использует Twitter API как транспортный уровень для приема (доставки) приложений и позволяет отказаться от использования SMS. Также рассмотрены детали реализации.

Библиография

[1] Brown, Jeff, Bill Shipman, and Ron Vetter. "SMS: The short message service." Computer 40.12 (2007): 106-110.

[2] Shieber, S. M. (1988, August). A uniform architecture for parsing and generation. In Proceedings of the 12th conference on Computational linguistics-Volume 2 (pp. 614-619). Association for Computational Linguistics.

[3] Al-Ali, A. R., M. A. Rousan, and M. Mohandes. "GSM-based wireless home appliances monitoring & control system." Information and Communication Technologies: From Theory to Applications, 2004. Proceedings. 2004 International Conference on. IEEE, 2004.

[4] Bayley, Gwain, et al. "Method of invoking and canceling voice or data service from a mobile unit." U.S. Patent No. 5,590,406. 31 Dec. 1996.

[5] Church, Karen, and Rodrigo de Oliveira. "What's up with whatsapp?: comparing mobile instant messaging behaviors with traditional SMS." Proceedings of the 15th international conference on Human-computer interaction with mobile devices and services. ACM, 2013.
[6] T411 for Twitter http://t411.linkstore.ru
[7] Sanderson, D. (2010). Programming Google app engine. O'Reilly.
[8] Bacchiani, Michiel, et al. "Deploying GOOG-411: Early lessons in data, measurement, and testing." Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on. IEEE, 2008.
[9] Wilde E., Vaha-Sipila A. Uri scheme for global system for mobile communications (gsm) short message service (sms) //IETF, Disponível online em http://www. ietf. org/rfc/rfc3966. txt. – 2010.
[10] Coldtags Suite http://servletsuite.com/jsp.htm
[11] А.А. Волков, Д.Е. Намиот, М.А. Шнепс-Шнеппе. О задачах создания эффективной инфраструктуры среды обитания //International Journal of Open Information Technologies. – 2013. – Т. 1. –No. 7. –С. 1-10.
[12] Namiot, Dmitry, and Manfred Sneps-Sneppe. "Customized check-in procedures." Smart Spaces and Next Generation Wired/Wireless Networking. Springer Berlin Heidelberg, 2011. 160-164.
[13] Namiot, D. (2010, October). Geo messages. In Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2010 International Congress on (pp. 14-19). IEEE.
[14] Намиот Д., Сухомлин В. О проектах лаборатории ОИТ //International Journal of Open Information Technologies. – 2013. – Т. 1. – №. 5. – С. 18-21.

Twitter as transport layer for information systems

Dmitry Namiot

***Abstract*— The article examines the use of Twitter mechanisms for design and deployment of information systems. In view of the transfer of mobile subscribers from SMS to independent messaging applications, how can we use Twitter to build information systems? The paper describes a model that allows developers to use Twitter instead of traditional mobile services based on SMS.**

***Keywords*—Twitter, SMS, client-server, bot.**