УДК 004.738

МЕЖСЕРВЕРНАЯ МАРШРУТИЗАЦИЯ HTTP/SOAP-ВЗАИМОДЕЙСТВИЙ В РАСПРЕДЕЛЕННЫХ СИСТЕМАХ

Р.Э. Асратян

Рассмотрен подход к повышению эффективности HTTP/SOAP-взаимодействий в глобальных сетях путем организации тоннеля между HTTP-клиентами и HTTP-серверами, обеспечивающего возможность межсерверного взаимодействия и межсерверной маршрутизации данных в сети, защиту от несанкционированного доступа и концептуальную устойчивость к сетевым сбоям. Описаны принципы организации такого тоннеля, включая принципы функционирования сетевых шлюзов, необходимых для его построения.

Ключевые слова: распределенные системы, Интернет-технологии, сетевые протоколы, Web-сервисы.

ВВЕДЕНИЕ

В последние годы возрастает интерес разработчиков распределенных систем к технологии Web-сервисов, основанных на сетевом протоколе SOAP (вариация протокола HTTP) [1]. Этот интерес объясняется рядом признанных достоинств этой технологии:

- языковой и платформенной независимостью;
- эффективной поддержкой взаимодействий в двухзвенной (клиент—сервер) и трехзвенной (клиент—сервер приложений—сервер баз данных) архитектурах в режиме on-line;
- ориентацией на машинную обработку результатов информационных запросов благодаря строгой формализации структуры электронных документов в сети (XML-формат);
- обеспечением строго формализованных спецификаций сервисов (WSDL) и удаленного доступа к этим спецификациям в процессе разработки приложений;
- поддержкой технологии в ряде развитых интегрированных сред разработки (MS Visual Studio, Borland C++ Builder), обеспечивающих удобный программный интерфейс (API), основанный на модели вызовов методов удаленных объектов, а также информационную связь с сервисами в период разработки и отладки приложений.

Хотя последнее из перечисленных достоинств является по сути конъюнктурным, оно играет едва ли не решающую роль в успехе Web-сервисов на «рынке» Интернет-технологий.

Тем не менее, технология Web-сервисов не свободна от ряда недостатков, которые особенно ощутимо проявляются в разработках больших распределенных систем, (т. е. систем, включающих в себя десятки и сотни взаимодействующих компонентов, удаленных друг от друга на сотни и тысячи километров).

- Как и большинство Интернет-технологий, работающих в режиме on-line, Web-сервисы крайне неустойчивы к сетевым сбоям. Если в процессе обработки клиентского запроса (возможно, весьма длительной) возникает разрыв сетевого соединения с сервером, то результаты обработки, как правило, оказываются потерянными. (Именно этим объясняется тот факт, что разработчики больших систем до сих пор часто строят удаленные взаимодействия на базе технологической электронной почты, т. е. путем отказа от режима on-line.)
- Web-сервисы не содержат встроенных средств защиты от несанкционированного доступа.
- Web-сервисы не поддерживают собственных средств межсерверного взаимодействия и межсерверной маршрутизации данных в сети. Это означает, что клиент может требовать обслуживания только у того сервера, с которым он непосредственно соединился, а два клиента могут взаимодействовать только при условии обслуживания общим сервером. Это ограничение может оказаться чрезмерно жестким для распределенных систем, предназначенных для функционирования в неоднородных сетях (т. е. в сетях, не обеспечивающих возможность прямого сетевого соединения между любыми двумя узлами). Все более часто встреча-

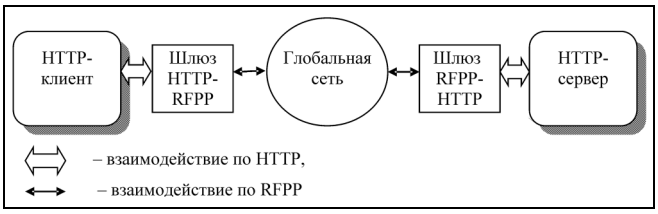

Рис. 1. RFPS-тоннель

ется ситуация, в которой сетевая среда распределенной системы представляет собой множество удаленных друг от друга частных локальных сетей, объединенных глобальной сетью (именно эту ситуацию мы и будем иметь в виду в данной статье). Желание «спрятать» сетевые узлы (и клиентские, и серверные) в частных сетях может быть продиктовано соображениями безопасности и (или) нехваткой уникальных IP-адресов. В этом случае возможность взаимодействия по схеме «клиент—сервер—сервер—...—сервер», т. е. взаимодействия через «посредников», оказывается чрезвычайно важной. (Отметим, что применение в этом случае защищенных VPN-тоннелей [2] для организации межсетевого взаимодействия является весьма жестким решением и может оказаться совершенно неэффективным, если число частных сетей больше двух.)

Разумеется, в каждой конкретной разработке эти недостатки могут быть хотя бы отчасти «компенсированы» на прикладном уровне, не это всегда хуже, чем доступность универсальных и готовых к использованию решений. Одно из таких решений предлагается в данной статье.

В работах [3, 4] описываются новая сетевая служба RFPS (Remote File Packets Service) и, соответственно, сетевой протокол RFPP, предназначенные для организации сетевых взаимодействий в распределенных системах. В разработке этой службы основной акцент сделан именно на средствах межсерверного взаимодействия, межсерверной маршрутизации данных и защиты от несанкционированного доступа, а также на концептуальной устойчивости к сетевым сбоям. В данной работе описывается подход к объединению преимуществ обеих сетевых технологий путем организации RFPS-тоннеля между HTTP-клиентом и HTTP-сервером (рис. 1). Идея подхода основана на использовании HTTP-взаимодействий в пределах частных локальных сетей и переходе на RFPP-взаимодействия для обменов данными в глобальной сети. Разумеется, главное требование к тоннелю заключается в его абсолютной «прозрачности» и для HTTP-клиентов, и для HTTP-серверов.

1. КРАТКИЕ СВЕДЕНИЯ О RFPS

Главная особенность RFPS состоит в том, что «контекст» сеанса связи ассоциируется не с сетевым соединением, а с новым, явно определяемым понятием — удаленным пакетом файлов (далее будем называть его просто «пакетом»), существующим независимо от наличия или отсутствия сетевого соединения. Клиент RFPS сам открывает новый пакет на RFPS-сервере при необходимости удаленного взаимодействия и закрывает его, когда он больше не нужен. В промежутке между этими событиями он может наполнить пакет данными, вызвать те или иные удаленные обработчики данных, получить результаты их работы, передать пакет другому клиенту или переслать его по сети на другой сервер, продолжить обработку на другом сервере, а потом получить его обратно. Причем, все это может быть сделано или за одно TCP-соединение, или за несколько. Сохраняя полученный от сервера уникальный идентификатор пакета, клиент всегда может восстановить контекст сеанса после случайного (или намеренного) разрыва соединения и продолжить работу.

В смысле управления пакетами файлов RFPS-сервер имеет дело с объектами пяти типов:

- *клиенты* — источники запросов на обслуживание (в том числе — на открытие и закрытие пакетов);
- *обработчики* — программы (исполняемые модули), запускаемые сервером для обработки наборов данных в пакетах;
- *агенты* — постоянно активные (или периодически запускаемые) программы, объединяющие в себе свойства клиентов и обработчиков; в отличие от обработчиков, агенты могут выполняться на отдельных машинах (т. е. не на тех, на которых работает сервер и (или) клиенты); пакеты передаются на обработку агентам через связанные с агентами очереди пакетов;
- *процедуры* — подпрограммы из динамических библиотек, подключаемых к серверу для расширения его функций;
- *серверы* — другие RFPS-серверы, доступные для взаимодействия.

Объекты всех типов должны быть предварительно зарегистрированы на сервере RFPS. Другими словами, сервер отвергает запросы, исходящие от незарегистрированных клиентов или агентов, или же запросы на запуск незарегистрированных обработчиков. При регистрации объекту присваивается уникальное имя; кроме того, с ним связывается определенная управляющая информация (пароль, права доступа, спецификация исполняемого модуля обработчика и т. п.). Отметим, что обращение клиента к серверу сопровождается «двусторонней проверкой подлинности» (защита сервера от неавторизованного клиента и защита клиента от фальсифицированного сервера) и «скрытой» передачей пароля по сети на основе примерно той же идеи, на которой построен протокол MS-Chap v2 [2].

К основным формам обработки пакетов файлов относятся:

— создание и уничтожение пакета;

— заполнение пакета данными от клиента;

— применение к пакету одной или нескольких обрабатывающих программ (обработчиков), зарегистрированных на обрабатывающем сервере;

— постановка пакета в очередь на обработку к постоянно активному или периодически запускаемому агенту;

— перемещение пакета с одного сервера на другой;

— выборка клиентом результатов обработки.

Все перечисленные формы обработки можно комбинировать в разных сочетаниях. Создав удаленный пакет файлов на каком-либо сервере RFPS, клиент может последовательно или параллельно применить к нему несколько обработчиков и (или) последовательно поставить его в очередь на обработку к нескольким агентам и (или) переместить его на другой RFPS-сервер для продолжения обработки. Все эти действия инициируются одной и той же клиентской функцией Process, которая имеет два основных параметра: идентификатор пакета файлов и командную строку, задающую последовательность имен обработчиков и агентов, которые должны принять участие в его обработке. В смысле организации RFPS-тоннеля наиболее важно, что эти обработчики и агенты могут быть «размещены» не только в том RFPS-сервере, с которым соединился клиент, но и на других (удаленных) серверах.

Рассмотрим следующий пример. Предположим, что клиентская программа выполнила соединение с RFPS-сервером с именем alpha (рис. 2), открыла на сервере пакет файлов и «заполнила» его какими-то данными. Вслед за этим программа выполняет функцию Process, которая передает серверу alpha довольно длинную командную строку для обработки пакета (остальные параметры функции Process мы отбросим) и переходит в состояние ожидания результатов обработки. Обработка пакета на сервере alpha начнется с вызова обработчика, зарегистрированного под именем unzip (во время работы обработчика пакет файлов предоставляется ему в форме рабочего каталога с полным доступом ко всем, содержащимся в нем, файлам).

Следующий элемент командной строки требует организации межсерверного взаимодействия. Он содержит регистрационное имя удаленного сервера (beta), отделенное знаком $ от «вложенной» командной строки (wk,gamma$trans,kw), которая должна быть выполнена на сервере beta. Организация межсерверного взаимодействия включает в себя:

- поиск параметров доступа к серверу beta (IP-адрес или Интернет-имя, порт, пароль и т. д.) в регистрационных данных сервера alpha и установление сетевого соединения с ним;
- создание «пакета-клона» на сервере beta, являющегося копией обрабатываемого пакета;

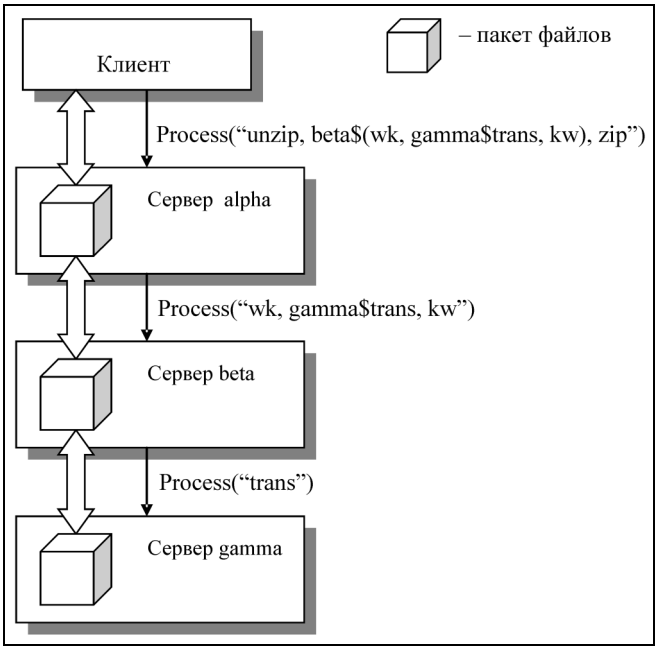

Рис. 2. Обработка пакета файлов на удаленных серверах

- передачу серверу beta вложенной командной строки для продолжения обработки.

Во все время обработки пакета на сервере beta сервер alpha будет находиться в состоянии ожидания. После завершения обработки вся информация из пакета-клона будет скопирована в исходный пакет, а сервер alpha продолжит обработку пакета вызовом обработчика с именем zip. (Существуют варианты взаимодействия без ожидания и без возврата результатов, но мы не будем останавливаться на них.)

В процессе обработки вложенной командной строки сервер beta, в свою очередь, обнаружит в ней имя удаленного сервера (gamma) и обратится к нему (по аналогичным правилам) для выполнения единственного обработчика trans.

Как видно из данного примера, RFPS совмещает в себе функции передачи и распределенной обработки данных, обеспечивая возможность аккумуляции результатов обработки одного и того же пакета файлов на разных серверах.

2. ПРИНЦИПЫ ОРГАНИЗАЦИИ RFPS-ТОННЕЛЯ

Рассмотрим множество частных локальных сетей (LAN), объединенных глобальной сетью, и предположим, что проектируемая распределенная система требует организации взаимодействия между узлами, размещенными не только в одной частной сети, но и в разных. Объединение частных сетей через глобальную сеть предполагает наличие в каждой из них, по крайней мере, одного «пограничного» узла, соединенного и с локальной, и с

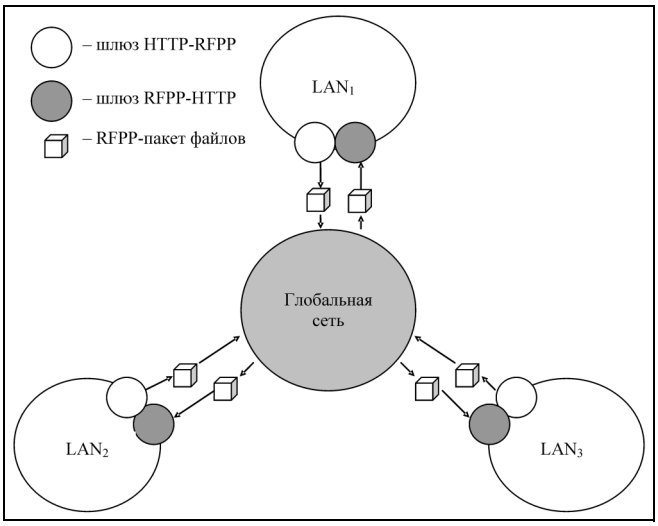

Рис. 3. Шлюзы между HTTP и RFPP

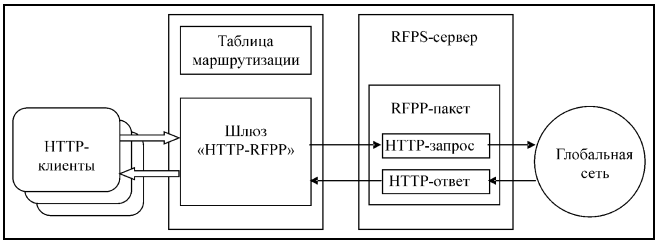

Рис. 4. Шлюз HTTP-RFPP

глобальной сетью и выполняющего роль IP-маршрутизатора [5]. Будем считать, что именно на таких узлах устанавливаются программные средства шлюзов HTTP-RFPP и RFPP-HTTP, обеспечивающих энкапсуляцию (и деэнкапсуляцию) потоков HTTP-запросов и HTTP-ответов в RFPP-пакеты файлов таким образом, что каждая пара {HTTP-запрос, HTTP-ответ} помещается в один пакет (рис. 3).

Шлюз HTTP-RFPP включает в себя две постоянно активные программы (службы): собственно шлюз и RFPS-сервер, обеспечивающий создание и обработку RFPP-пакетов (рис. 4). Каждой программе отводится выделенный сетевой порт. Шлюз приводится в действие путем глобальной настройки всех HTTP-клиентов в локальной сети на работу через «проки-сервер» с указанием IP-адреса и порта шлюза в качестве параметров настройки. В результате все исходящие HTTP-соединения от клиентов (независимо от URL адресуемого Интернет-ресурса) будут поступать непосредственно «на вход» шлюза.

Обработка каждого входящего HTTP-соединения в шлюзе HTTP-RFPP опирается на таблицу маршрутизации шлюза, которая связывает доменные имена адресуемых Интернет-ресурсов с именами удаленных RFPS-серверов и IP-адресами узлов в удаленных локальных сетях. С помощью таблицы маршрутизации шлюза выполняется первый шаг маршрутизации — последующие шаги реализуются средствами RFPS.

Обработка входящего соединения состоит из следующих основных шагов.

- Прием HTTP-запроса от HTTP-клиента по сети.
- Извлечение Интернет-имени удаленного узла из URL адресуемого Интернет-ресурса (Web- страницы, Web-сервера и т. п.) и поиск записи, соответствующей этому имени в таблице маршрутизации шлюза. Если такая запись отсутствует, шлюз устанавливает соединение с удаленным узлом без «перехода» на протокол RFPP и выполняет обработку данного входящего соединения в режиме классического прокси-сервера (данный режим мы рассматривать не будем).
- Извлечение командной строки RFPS и IP-адреса адресуемого ресурса в удаленной локальной сети из найденной строки таблицы маршрутизации. Командная строка RFPS должна содержать полное имя шлюза RFPS-HTTP на удаленном сервере (т. е. имя RFPS-сервера и регистрационное имя шлюза в форме «имя_сервера$имя_шлюза»). Полученный из таблицы IP-адрес ресурса помещается в заголовок HTTP-запроса.
- Соединение с RFPS-сервером, открытие RFPP-пакета и размещение в нем HTTP-запроса в форме текстового файла.
- Инициация обработки пакета с помощью полученной командной строки RFPS. В результате пакет будет передан по глобальной сети в один из удаленных RFPS-серверов и обработан шлюзом RFPS-HTTP в другой локальной сети.
- Ожидание окончания обработки и прием обработанного пакета от удаленного RFPS-сервера по глобальной сети (обработанный пакет должен содержать не только файл HTTP-запроса, но и файл HTTP-ответа).
- Отправка HTTP-ответа HTTP-клиенту и закрытие RFPP-пакета.

Во все время обработки входящего соединения HTTP-клиент находится в состоянии ожидания.

Таблица маршрутизации шлюза представляет собой текстовый файл, каждая строка которого имеет следующий формат:

«Имя ресурса» «Командная строка RFPS-сервера» «IP-адрес ресурса»

Имя ресурса играет роль ключа при поиске строки, а остальные реквизиты представляют собой результат поиска. Рассмотрим следующий пример таблицы.

alpha Srv1$RfppHttp 192.168.0.10
beta Srv2$ RfppHttp 192.168.0.10
gamma Srv3$ RfppHttp 192.168.0.20

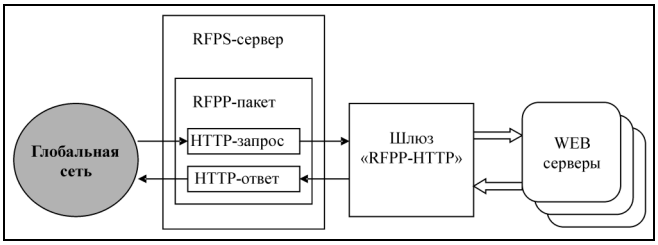

Рис. 5. Шлюз RFPP-HTTP

При обработке запроса к ресурсу http://beta/document1 будет найдена вторая строка таблицы. После создания RFPP-пакета он будет передан по глобальной сети в удаленный RFPS-сервер Srv2, где к нему будет применен обработчик с регистрационным именем RfppHttp — шлюз RFPP-HTTP. Этот шлюз должен обеспечить обращение к ресурсу http://192.169.0.10/document1 в обслуживаемой им локальной сети по протоколу HTTP.

Шлюз RFPP-HTTP выполняет обратную задачу по отношению к шлюзу HTTP-RFPP. По своему «статусу» он представляет собой специализированный обработчик в RFPS-сервере (рис. 5). Это означает, что он получает управление в результате выполнения в RFPS-сервере командной строки, в которой обозначено его имя. Обработка RFPP-пакета в шлюзе состоит из следующих шагов.

- Просмотр заголовка HTTP-запроса и извлечение IP-адреса адресуемого ресурса в локальной сети.
- Установление соединения с Web-сервером на соответствующем узле по локальной сети.
- Передачу HTTP-запроса по установленному соединению.
- Ожидание окончания обработки запроса, прием HTTP-ответа и запись его в обрабатываемый RFPP-пакет в форме текстового файла.

Доставка пакета по глобальной сети в исходный RFPS-сервер осуществляется средствами RFPS.

ЗАКЛЮЧЕНИЕ

Описанные в данной работе принципы организации RFPS-тоннеля одинаково применимы для обслуживания взаимодействий двух видов:

- «тонких» HTTP-клиентов (Web-браузеров) с Web-сайтами,
- «толстых» HTTP-клиентов с Web-сервисами.

Безусловно, RFPS-тоннель вносит определенную задержку во взаимодействие HTTP-клиентов и HTTP-серверов. (Эта задержка особенно заметна в случае обращения «тонких» клиентов к «нагруженным» Web-страницам с большим числом картинок и внешних таблиц стилей или скриптов — ведь каждая такая «деталь оформления» влечет за собой отдельное HTTP-обращение и отдельную доставку в RFPP-пакете). Поэтому, область эффективного применения подхода включает в себя взаимодействия, связанные с достаточно длительной удаленной обработкой (информационные запросы к базам данных, формирование отчетов и т. п.) и с передачей в основном содержательных данных. Через Web-сервисы обычно реализуются взаимодействия именно такого типа.

Кроме основных преимуществ RFPS-тоннеля, перечисленных в начале статьи, упомянем еще одно: возможность организации дополнительной обработки передаваемых данных средствами RFPS. В тот период времени, в который HTTP-запрос и (или) HTTP-ответ содержатся в форме наборов данных в RFPP-пакете, они являются весьма удобным объектом для применения к ним самых различных форм обработки, включая, например, следующие:

- шифровка данных пред отправкой в глобальную сеть и дешифровка после приема из глобальной сети;
- отправка копий всех RFPP-пакетов специализированному удаленному агенту для централизованного мониторинга взаимодействий в распределенной системе;
- преобразования форматов данных (например, их XML в HTML) и т. п.

В настоящее время программные средства поддержки RFPS-тоннеля (шлюз HTTP-RFPP и шлюз RFPP-HTTP) реализованы в операционной среде Win32. Эксперименты подтвердили применимость и эффективность подхода для обслуживания разных форм «технологических» HTTP-взаимодействий: обращения к Web-сервисам из клиентских рабочих станций, вызов удаленных CGI-программ из Web-браузеров, обращения интегрированной среды разработки к удаленному Web-серверу за спецификациями Web-сервисов и т. п.

ЛИТЕРАТУРА

1. *Шапошников И.В.* Web-сервисы Microsoft.NET. — СПб: БХВ-Петербург, 2002. — 336 с.
2. *Widows* 2000: Server и Professional / Андреев А.Г. и др. — СПБ.: «БХВ-Санкт-Петербург», 2001. — 1055 с.
3. *Асратян Р.Э.* Интернет-служба для поддержки распределенных информационно-управляющих систем // Проблемы управления. — 2005. — № 6. — С. 73—77.
4. *Асратян Р.Э.* Интернет-служба для поддержки межсерверных взаимодействий в распределенных информационных системах // Там же. — 2006. — № 5. — С. 58—62.
5. *Джамса К., Коуп К.* Программирование для Интернет в среде Windows. — СПБ: Питер, 1996. — 659 с.

Статья представлена к публикации членом редколлегии В.Г. Лебедевым.

Асратян Рубен Эзрасович — канд. техн. наук, вед. науч. сотрудник, Институт проблем управления им. В.А.Трапезникова РАН, ☎ (495) 334-88-61, e-mail: rea@L9.ipu.rssi.ru