УДК 004.75 + 004.032.6

И. В. Масалов [1], И. Г. Таранцев [1, 2]

[1] Новосибирский государственный университет
ул. Пирогова, 2, Новосибирск, 630090, Россия
E-mail: masalov@sl.iae.nsk.su

[2] Институт автоматики и электрометрии СО РАН
пр. Акад. Лаврентьева, 1, Новосибирск, 630090, Россия
E-mail: egor@sl.iae.nsk.su

РАСПРЕДЕЛЕННАЯ СИСТЕМА АВТОМАТИЗАЦИИ УПРАВЛЕНИЯ ПОТОКАМИ МУЛЬТИМЕДИА ДАННЫХ

Данная работа содержит описание прототипа распределенной системы автоматизации управления потоками мультимедиа данных. Рассматриваемый прототип позволяет производить управление несколькими потоками с одного рабочего места оператора. Кроме того, возможно управление одним потоком несколькими операторами. Прототип реализован на базе персонального компьютера и построен в виде системного сервиса и двух загружаемых модулей.

Ключевые слова: распределенная система, автоматизация, мультимедиа, WCF (Windows Communication Foundation).

Введение

В настоящее время широкое распространение получают системы автоматизации управления потоками мультимедиа данных. При этом для систем автоматизации телевещания ключевым фактором является повышение надежности системы в целом. Следует отметить, что использование распределенной системы позволяет добиться значительного увеличения надежности и удобства управления: появляется возможность дублирования серверов, занимающихся воспроизведением; возможность управления несколькими серверами воспроизведения с одного рабочего места оператора и возможность управления одним сервером воспроизведения несколькими операторами. Целью представленной работы являлось построение прототипа распределенной системы автоматизации управления потоками мультимедиа данных.

Из всего многообразия существующих систем автоматизации телевещания можно выделить три основные группы. Первая группа – системы, основанные на управлении через порт RS-232/RS-422. Их основными недостатками являются сложности в администрировании и совмещении устройств от различных производителей. Примером подобных систем являются AutoPlay [1] компании JCSI и система VSN [2] компании «Just Edit». Вторая группа – системы типа «канал в коробке». Как понятно из названия, подобные системы представляют собой законченное решение, включающее все необходимые компоненты (место оператора и воспроизводящий сервер) и не позволяющее произведение интегрирования с внешними системами управления. Примером может служить система Xstation [3] компании «Miranda». Последняя группа включает в себя системы, основанные на управлении через MOS-протокол [4]. Основное ограничение систем, использующих MOS-протокол, заключается в том, что одному воспроизводящему серверу ставится в соответствие один объект, занимающийся воспроизведением. Это приводит к возникновению принципиальных трудностей при необходимости воспроизведения нескольких мультимедиа объектов одним сервером (например, видео

[1] Описание программно-аппаратной системы автоматизации вещания AutoPlay компании «JC System Integration», 2007 (http://www.jcsi.ru/tv/products/jcsi/autoplay/autoplay.htm).

[2] Описание программной системы автоматизации вещания VSN (Video Stream Networks) компании «JustEdit, S.L.», 2007 (http://www.vsn-tv.com/en-up/welcome.htm).

[3] Описание сервера вещания Xstation компании «Miranda Technologies», 2007 (http://www.miranda.com/product.php?l=1&i=353).

[4] Описание протокола MOS (Media Object Server Communications Protocol), MOS Group, 2007 (http://www.mosprotocol.com).

с набором титров и / или логотипов). Примером систем автоматизации телевещания данного типа является комплекс систем автоматизации вещания компании «SGT», отвечающих за управление контентом [5], процессом вещания [6] и планированием вещания [7][7]. Однако в стандарте MOS-протокола реализовано несколько важных идей построения распределенных систем, поэтому разработка системы базировалась на MOS-протоколе.

Толчком к построению распределенной системы послужила потребность дальнейшего развития системы автоматизации телевещания ФорвардТ [1], разрабатываемой в лаборатории программных систем машинной графики ИАиЭ СО РАН. Система ФорвардТ (рис. 1) состоит из двух частей: клиентского приложения и набора плееров. В настоящее время используется конфигурация, при которой клиентское приложение и набор плееров располагаются на одном персональном компьютере. Тот факт, что взаимодействие между плеером и клиентом основано на использовании технологии DCOM, позволяет разнести компоненты системы на различные персональные компьютеры, однако при этом все равно возникает ряд ограничений, накладываемых ОС Windows и препятствующих нормальной работе системы.

После проведения анализа недостатков системы ФорвадТ было принято решение вынести работу по управлению плеерами и хранению расписания в отдельный модуль [2].

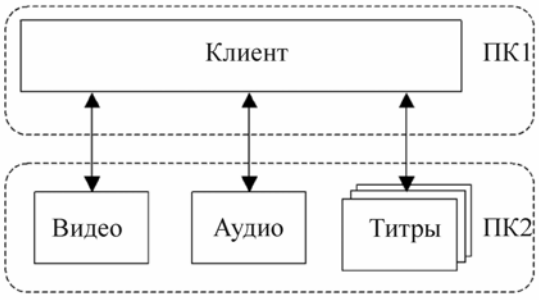

Рис. 1. Архитектура системы ФорвардТ

[5] Система управления контентом компании «SGT», 2007 (http://www.sgt.fr/solutions_media.php).

[6] Система управления процессом вещания компании «SGT», 2007 (http://www.sgt.fr/solutions_transmission.php).

[7] Система управления планированием вещания компании «SGT» (2007, http://www.sgt.fr/solutions_traffic.php).

При этом необходимо было реализовать поддержку взаимодействия локальных и удаленных клиентских приложений с модулем, управляющим плеерами. В процессе выполнения работы была разработана и детализирована архитектура прототипа (рис. 2).

При этом выделяется три основных компонента прототипа: сервер взаимодействия, клиентское приложение и ресурс. Сервер взаимодействия ответственен за передачу сообщений между компонентами прототипа, находящимися как на различных персональных компьютерах, так и на одном. Кроме того, сервер взаимодействия отвечает за авторизацию клиента. К каждому серверу взаимодействия может быть подключено произвольное количество локальных клиентских приложений. В случае, если сервер взаимодействия сконфигурирован на работу с ресурсами, возможно подключение к нему произвольного количества ресурсов.

При выборе способа взаимодействия между компонентами системы был рассмотрен ряд технологий взаимодействия клиент – сервер: DCOM [3], Raw TCP/IP, CORBA [4], RMI (Java) [5], .NET Web Service [6], XML/RPC (SOAP) [7], WCF [8]. Из приведенного списка была выбрана технология WCF, так как в ней уже решены такие важные вопросы, как удаленная авторизация и встроенные средства интеграции с существующими DCOM-компонентами.

Компоненты распределенной системы

Сервер взаимодействия можно разбить на четыре блока:

1. блок взаимодействия сервера с клиентским приложением (сервер);
2. блок взаимодействия сервера с ресурсами системы (сервер);
3. блок приема запросов (сервер);
4. блок передачи запросов (клиент).

Для обеспечения возможности обращения клиентского приложения к серверу взаимодействия в адресное пространство клиентского приложения загружается специальный интерфейсный модуль. В адресное пространство ресурса также загружается соответствующий интерфейсный модуль. Из всех ресурсов выделяется два основных типа: «база данных заданий» и «плеер заданий». Ресурс «база данных заданий» предназначен для хранения заданий и их параметров. Ресурс «плеер» позволяет организовать воспроизведение заданий по расписанию. Для данной цели он

обладает набором плееров, позволяющих воспроизводить определенные задания, и расписанием для каждого из них. Расписание представляет собой набор блоков заданий. Внутри блока задания воспроизводятся последовательно одно за другим. Каждый блок может запускаться одним из трех способов: по команде оператора; по времени; после указанного блока.

Из всех режимов работы прототипа можно выделить следующие основные режимы:

1) инициализация сервера взаимодействия;

2) соединение сервера с ✉ блоками приема запросов удаленных серверов взаимодействия;

3) инициализация ◒ интерфейсного модуля клиентского приложения;

4) авторизация клиента;

5) отправка запроса от клиентского приложения к ресурсу;

6) инициализация ◒ интерфейсного модуля ресурса;

7) обработка запроса ресурсом;

8) отправка запроса от ресурса к ресурсу.

Инициализация сервера взаимодействия

При инициализации сервера взаимодействия используется два конфигурационных файла. Первый файл включает список сетевых адресов всех серверов взаимодействия, предоставляющих доступ к ресурсам. Этот файл должен дублироваться на все персональные компьютеры, так или иначе пользующиеся услугами распределенной системы. Второй файл может отсутствовать. Он включает список локальных ресурсов, запускаемых при старте сервиса. Сервис везде присутствует в системе в единственном экземпляре, поэтому очень удобно запускать ресурсы именно при старте сервиса.

Инициализация сервиса начинается с разделения списка сетевых адресов на локальные и удаленные адреса, основываясь на IP-адресах локального персонального компьютера. После разделения списка сетевых адресов (при условии, что сервер взаимодействия предоставляет доступ к ресурсам) происходит инициализация ✉ блока взаимодействия с ресурсами и ✉ блока приема запросов. Инициализация ✉ блока взаимодействия с ресурсами состоит из запуска сервиса взаимодействия с ресурсами и активации указанных ресурсов.

В отличие от предыдущих двух блоков инициализация ✉ блока передачи запросов и ✉ блока взаимодействия с клиентами происходит всегда. При инициализации ✉ блока передачи производится инициализация объектов соединения для каждого из удаленных серверов взаимодействия и запуск потока восстановления соединений. При этом каждый из объектов соединения создает отдельный поток, осуществляющий отправку сообщений из очереди для соответствующего удаленного сервера взаимодействия и проверку соединения при пустой очереди. Последним этапом является инициализация ✉ блока взаимодействия с клиентами, состоящая в запуске сервиса взаимодействия с клиентскими приложениями.

Соединение с блоками приема запросов удаленных серверов взаимодействия

При соединении с удаленными серверами взаимодействия используется список удаленных сетевых адресов. При успешном соединении посылается запрос имени и идентификатора сервера взаимодействия. В том случае, если соединение с сервером взаимодействия, обладающим полученным идентификатором, не установлено, посылается запрос на регистрацию. При получении запроса данного типа удаленный сервер взаимодействия создает уникальный идентификатор соединения и возвращает его.

Если же соединение с удаленным сервером взаимодействия, обладающим полученным идентификатором, уже было установлено, но помечено как неработоспособное, то также посылается запрос на регистрацию. Однако в этом случае в запросе передается

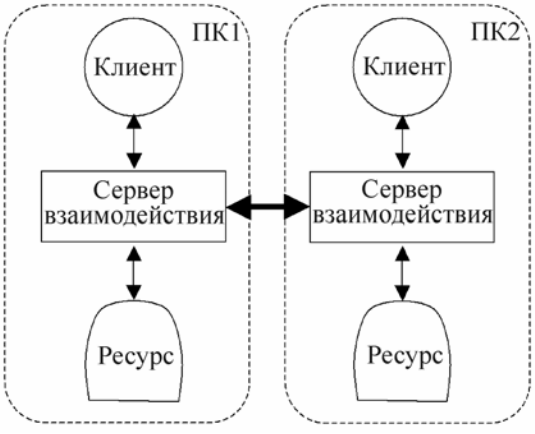

Рис. 2. Архитектура прототипа

идентификатор предыдущего соединения. При этом удаленный сервер взаимодействия изначально производит поиск объекта, описывающего соединение с полученным идентификатором. В случае успешного поиска производится обновление параметров соединения, в противном случае порядок действий совпадает с порядком действий при установке нового соединения.

При завершении работы сервера взаимодействия производится нотификация всех зарегистрированных удаленных серверов взаимодействия.

Инициализация интерфейсного модуля клиентского приложения

После запуска клиентского приложения происходит инициализация интерфейсного модуля, первым шагом которой является соединение с локальным сервером взаимодействия. При успешном соединении интерфейсный модуль посылает запрос на регистрацию. Получая запрос на регистрацию клиентского приложения, локальный сервер взаимодействия создает уникальный идентификатор клиента и на его основе объект, описывающий регистрируемое клиентское приложение. Идентификатор клиента используется для разделения запросов от различных клиентских приложений, а также для определения, какому клиентскому приложению следует передать результат запроса. По завершении регистрации интерфейсному модулю возвращается идентификатор клиента. После этого интерфейсный модуль посылает запрос на открытие сессии на каждом из доступных серверов взаимодействия.

Авторизация клиента

Для определения прав клиентского приложения на исполнение какой-либо операции с ресурсом необходимо реализовать механизм авторизации клиента у конкретного ресурса. Для этой цели вводится понятие сессии. Каждое клиентское приложение перед отправкой запросов ресурсу должно пройти операцию открытия сессии на сервере взаимодействия, на котором находится данный ресурс. Результатом данной операции является объект сессии, обладающий уникальным идентификатором сессии и списком идентификаторов безопасности [8]. В дальнейшем каждый запрос сопровождается идентификатором сессии. Идентификатор сессии является временным, и при повторном открытии создается новый идентификатор. Отличительной чертой отправляемых при исполнении данной операции запросов является использование возможности делегирования контекста пользователя. Данная возможность предоставляется технологией WCF и позволяет производить исполнение запроса в контексте пользователя, отправившего запрос.

При установке сессии на локальном сервере взаимодействия производится получение списка идентификаторов безопасности и сохраняется соответствие «идентификатор клиента – список идентификаторов безопасности».

Отправка запроса от клиентского приложения ресурсу

После инициализации интерфейсного модуля пользователь может отправить запрос выбранным ресурсам на выполнение одной из требуемых операций. Для этого клиентское приложение передает набор идентификаторов серверов взаимодействия и идентификаторов ресурсов с параметрами запроса интерфейсному модулю. При этом выделяется специальное значение идентификатора – «любой». В случае, если идентификатор ресурса принимает это значение, запрос посылается всем ресурсам указанного сервера взаимодействия. Если же идентификатор сервера взаимодействия и идентификатор ресурса принимают значение «любой», запрос посылается всем ресурсам всех серверов. Отправка запроса конкретному ресурсу для всех серверов невозможна. Кроме того, клиентское приложение при передаче запроса интерфейсному модулю указывает тип запроса: синхронный или асинхронный. При отправке синхронного запроса возврат из функции осуществляется только после получения интерфейсным модулем ответов от всех ресурсов, которым отправлялся запрос, или по истечении заданного интервала времени. При отправке асинхронного запроса возврат из функции осуществляется сразу после передачи запроса интерфейсным модулем

[8] Идентификатор безопасности (Security Identifier, SID) – уникальный идентификатор, используемый ОС Windows для авторизации пользователя или группы. Информация об идентификаторах безопасности общедоступна и обладание ей не предоставляет возможности выполнения действий с правами, предоставляемыми данным идентификатором.

✉ блоку взаимодействия с клиентами локального сервера взаимодействия. В данном случае клиентское приложение также передает ◒ интерфейсному модулю объект обратного вызова. При получении ответа от каждого из запрашиваемых ресурсов ◒ интерфейсный модуль вызывает полученный от клиентского приложения объект. Кроме того, данный объект используется для сигнализации получения ответов от всех ресурсов, а также для сигнализации истечения времени ожидания ответов.

Получая запрос от клиентского приложения, ◒ интерфейсный модуль создает новый объект, описывающий запрос, с уникальным идентификатором и добавляет его в список отправленных запросов. В качестве идентификатора запроса используется его последовательный номер. При асинхронном запросе данный идентификатор возвращается функцией отправки запроса, а также указывается при передаче ответа запроса ◒ интерфейсным модулем клиентскому приложению. Это позволяет клиентскому приложению произвести разделение полученных результатов на разные запросы. Следующим шагом является преобразование запроса в XML-формат и передача его локальному серверу взаимодействия. В случае, если запрос направлен локальному ресурсу, происходит добавление запроса в очередь для соответствующего ресурса. В противном случае запрос добавляется в очередь на отправку удаленному серверу взаимодействия. После того, как запрос передается удаленному серверу взаимодействия, происходит добавление данного запроса в очередь для соответствующего ресурса. В итоге запрос вместе с набором идентификаторов безопасности, определенных по идентификатору сессии, передается ресурсу. Ресурс распаковывает запрос из XML-формата и по полученному списку идентификаторов безопасности определяет возможность выполнения запроса. После выполнения запроса ресурс передает результат по цепочке ◒ интерфейсному модулю клиентского приложения. При получении ответа ◒ интерфейсный модуль производит поиск объекта запроса по идентификатору запроса. При этом производится проверка на то, что данный запрос был отправлен указанному ресурсу. Дальнейшая последовательность действий зависит от типа запроса. В случае, если объект запроса содержит флаг синхронности, у найденного объекта выставляется информация о полученном ответе. При получении ответов от всех ресурсов или по истечении времени ожидания объект запроса удаляется из списка отправленных запросов. После этого происходит возврат из функции запроса. Если же объект запроса содержит флаг асинхронности, то ресурс, от которого был получен ответ, удаляется из списка ресурсов у объекта запроса. После этого ответ передается клиентскому приложению. При удалении всех ресурсов из списка посылается нотификация клиентскому приложению. По истечении времени ожидания объект запроса удаляется из списка отправленных запросов и посылается нотификация клиенту с указанием ресурсов, ответ от которых не был получен. При получении ответа на запрос, который не содержится в списке (был удален или такого запроса отправлено не было), или при отсутствии ресурса, от которого был получен ответ, в списке объекта запроса ответ игнорируется.

Инициализация интерфейсного модуля ресурса

Первым этапом инициализации ▢ интерфейсного модуля ресурса является считывание конфигурационного файла конкретного ресурса. Следующим этапом инициализации является соединение с локальным сервером взаимодействия. После успешного соединения производится регистрация ресурса. При этом ▢ интерфейсный модуль посылает запрос на регистрацию, указывая идентификатор ресурса. Сервер взаимодействия, получая запрос на регистрацию, проверяет уникальность идентификатора и добавляет ресурс в список зарегистрированных ресурсов. При завершении работы ресурса ▢ интерфейсный модуль сообщает об этом локальному серверу взаимодействия.

Отправка и обработка запросов ресурсом

▢ Интерфейсные модули для ресурса типа «база данных заданий» и ресурса типа «плеер» предоставляют несколько разную функциональность. В случае ресурса типа «база данных заданий» предоставляются только функции возврата результата запроса. В то же время для ресурса типа «плеер» предоставляется набор функций для отправки запроса в «базу данных заданий» на получение информации о заданиях. Посылка запроса от ресурса «плеер» к ресурсу «база

данных заданий» производится практически так же, как и посылка запроса клиентским приложением. Отличие состоит в том, что запрос передается со специальным идентификатором сессии. В этом случае ресурсу передается пустой список идентификаторов безопасности.

При получении запроса интерфейсный модуль первым шагом производит проверку прав на исполнение запрошенной операции. При этом операция разрешается в случае, если хотя бы один из полученных идентификаторов безопасности содержится в списке идентификаторов безопасности, полученных из конфигурационного файла, и набор прав для данного идентификатора позволяет выполнение запрошенной операции. В случае успешной проверки прав запрос передается ресурсу, который производит обработку запроса и выдает результат. Однако такой алгоритм не подходит для проверки прав при запросе от другого ресурса, так как обычно ресурс запускается системным пользователем и использование его идентификаторов безопасности лишено особого смысла. Поэтому при получении пустого списка идентификаторов безопасности интерфейсный модуль предоставляет права на получение конкретной информации о задании по указанному идентификатору. Использование в качестве идентификатора задания значения типа guid [9] позволяет предотвратить получение информации обо всех доступных заданиях «недобросовестным» клиентским приложением путем перебора идентификаторов заданий за разумное время. Под «недобросовестным» клиентским приложением в данном случае понимается клиентское приложение, передающее в запросе специальный идентификатор сессии.

Заключение

Прототип описываемой системы построен на базе технологии WCF в виде системного сервиса и двух загружаемых модулей. Для проверки работоспособности прототипа были разработаны тестовые сегменты клиентского приложения и два тестовых ресурса: один типа «плеер» и один типа «база данных». С помощью разработанных компонентов протестированы все основные сценарии взаимодействия компонент системы, а также произведено измерение среднего времени выполнения каждого из сценариев. Результаты тестирования подтвердили применимость описанного способа построения распределенной системы автоматизации управления потоками мультимедиа данных. В дальнейшем планируется провести комплексную оптимизацию системы с учетом особенностей технологии WCF.

Список литературы

1. *Арсенин И. М., Морозов Б. Б., Токарев А. С., Шадрин М. Ю.* «Форвард» – многофункциональный программно-аппаратный комплекс для видеопроизводства на базе персонального компьютера // Материалы 4-ой Всероссийской конференции РОАИ-4 98. Новосибирск, 1998.

2. *Масалов И. В.* Построение распределенной системы автоматизированного воспроизведения разнородных мультимедиа данных // Материалы XLV международной научной студенческой конференции «Студент и научно-технический прогресс», секция «Физика». Новосибирск, 2007.

3. *Модель* COM и применение ATL 3.0, Э. Трельсен, БХВ-Петербург, Санкт-Петербург, 2001.

4. *Advanced* CORBA Programming with C++, Michi Henning, Steve Vinoski, Addison Wesley, 1999.

5. *Java* RMI, William Grosso, O'Reilly, 2001.

6. *Programming* .NET Web Services, Alex Ferrara, Matthew MacDonald, O'Reilly, 2002.

7. *Programming* Web Services with XML-RPC, Simon St. Laurent, Edd Dumbill, Joe Johnston, O'Reilly, 2001.

8. *Programming* WCF Services, Juval Lowy, O'Reilly, 2007.

Материал поступил в редколлегию 05.06.2007

[9] Globally unique identifier – 128-битный идентификатор с гарантированной уникальностью, генерируемый стандартными средствами ОС Windows.