УДК 519.876.7

Е.А. КОПЫТОВА

АНАЛИЗ ТЕХНОЛОГИЙ ПОСТРОЕНИЯ РАСПРЕДЕЛЕННЫХ УПРАВЛЯЮЩИХ СИСТЕМ

Введение

В настоящее время существует необходимость использовать распределенные управляющие системы (РУС) для эффективного управления территориально распределенной структурой. Создание такой системы позволяет организовывать сбор данных о работе сети, вести единый учет данных, использовать эту систему для оперативного принятия управляющих решений при условии большой территориальной распределенности. На сегодняшний день разработаны и внедрены ряд идей по построению РУС.

Основная часть

Существующие варианты технологий построения распределенных систем можно условно разделить на три группы:

- иерархическая;
- платформенная (COM, DCOM, CORBA, RMI, SOA и ДР.);
- агентная (МАС).

Рассмотрим эти технологии детальнее.

Одна из общеизвестных технологий построения распределенных систем – иерархическая, в которой подсистемы расположены на уровнях с различными степенями иерархичности. Подсистема на каком-либо уровне управляет или координирует подсистемы, расположенные на уровне ниже нее, и, в свою очередь, управляется или координируется подсистемой расположенной уровнем выше. Иерархическое управление в системах большой размерности имеет свои недостатки. Многоуровневость и связанная с этим многократная передача информации с уровня на уровень вызывают задержки, снижающие оперативность оценки обстановки и реализации управленческих решений, приводят к искажениям в процессе передачи информации и при ее обработке на промежуточных уровнях. Также к недостаткам иерархических систем стоит отнести жесткую структуру, которая приводит к сложности масштабирования.

Выделяются несколько различных платформенных технологий построения РУС. К ним относятся технологии RMI, CORBA и DCOM и др.

COM (англ. Component Object Model – Объектная Модель Компонентов) – это технологический стандарт от компании Microsoft, предназначенный для создания программного обеспечения на основе взаимодействующих распределенных компонентов, каждый из которых может использоваться во многих программах одновременно [1]. На основе COM была создана технология DCOM. Технология DCOM (англ. Distributed COM – распределенная COM) основана на технологии DCE/RPC (разновидности RPC). DCOM позволяет COM-компонентам взаимодействовать друг с другом по сети.

На рис. 1 представлена архитектура DCOM. Управление в сетях, основанных на технологии DCOM, производится посредством COM run-time, который предлагает клиентам и компонентам объектно-ориентированные сервисы и использует RPC и провайдер безопасности для генерации стандартных сетевых пакетов, соответствующих стандарту протокола DCOM. RPC (Remote Procedure Call) – стандарт сетевого программирования, определяющий мощную технологию для того, чтобы создавать распределенные программы клиент/сервер. RPC заглушки (stub) и библиотеки времени выполнения (run-time library) управляют большинством процессов, касающихся сетевых протоколов и связи. DCE RPC – бинарный протокол на базе различных транспортных протоколов, в том числе TCP/IP, который определяет стандарт для преобразования структур данных и параметров в сетевые пакеты.

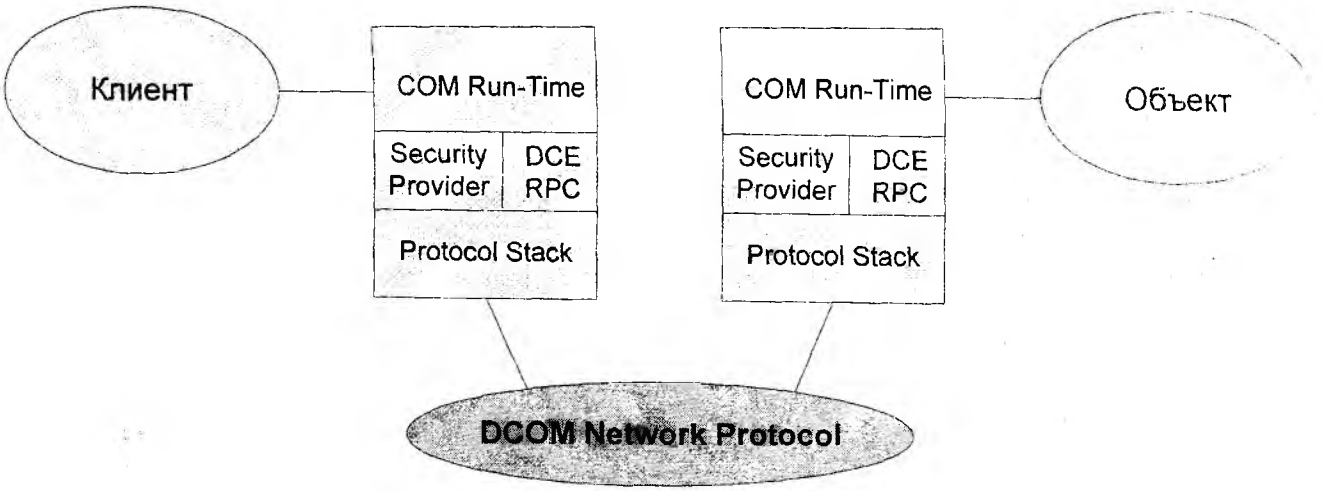

Рис. 1

Среди достоинств данной технологии стоит отметить языковую независимость, динамический/статический вызов, динамическое нахождение объектов, масштабируемость. Но при переходе от технологии COM к DCOM разработчики не устранили сложность реализации данной технологии и зависимость от платформы (каждый компонент распределенного DCOM-приложения должен обязательно выполняться под управлением операционной системы Windows, что не всегда удобно).

Главным конкурентом DCOM является другая известная распределенная технология CORBA.

CORBA (сокр. от англ. Common Object Request Broker Architecture – общая архитектура брокера объектных запросов) – это технологический стандарт написания распределенных приложений. Задача CORBA – осуществить интеграцию изолированных систем, дать возможность программам, написанным на разных языках, работающим на разных узлах сети, взаимодействовать друг с другом так же просто, как если бы они находились в адресном пространстве одного процесса.

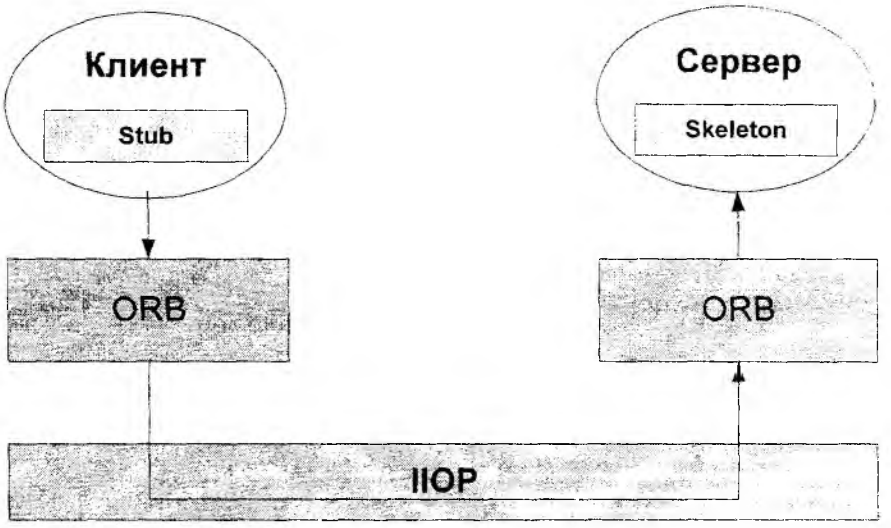

Рис. 2

Взаимодействие объектов в архитектуре CORBA представлено на рис. 2. Основу управления в CORBA составляет объектный брокер запросов (Object Request Broker). ORB управляет взаимодействием объектов в распределенной сетевой среде. IIOP (Internet Inter-ORB Protocol) – это протокол для организации взаимодействия между различными брокерами, опубликованный консорциумом OMG. IIOP используется GIOP в среде Интернет и обеспечивает отображение сообщений между GIOP и слоем TCP/IP. В адресном пространстве клиента функционирует специальный объект, называемый заглушкой (stub). Получив запрос от клиента, он упаковывает параметры запроса в специальный формат и передает его серверу, а точнее, скелету. Скелет (skeleton) – объект, работающий в адресном пространстве сервера. Получив запрос от клиента, он распаковывает его и передает серверу. Также скелет преобразует ответы сервера и передает их клиенту (заглушке).

Значительным преимуществом CORBA относительно технологии COM и DCOM является платформенная независимость и широкий круг производителей продуктов, поддерживающих данную технологию, в сравнении с аналогичным кругом для DCOM. Архитектура CORBA разработана для поддержки интеграции самых разнообразных объектных систем.

Технология RMI (Remote Method Invocation, т.е. вызов удаленного метода), реализует распределенную модель вычислений (рис. 3).

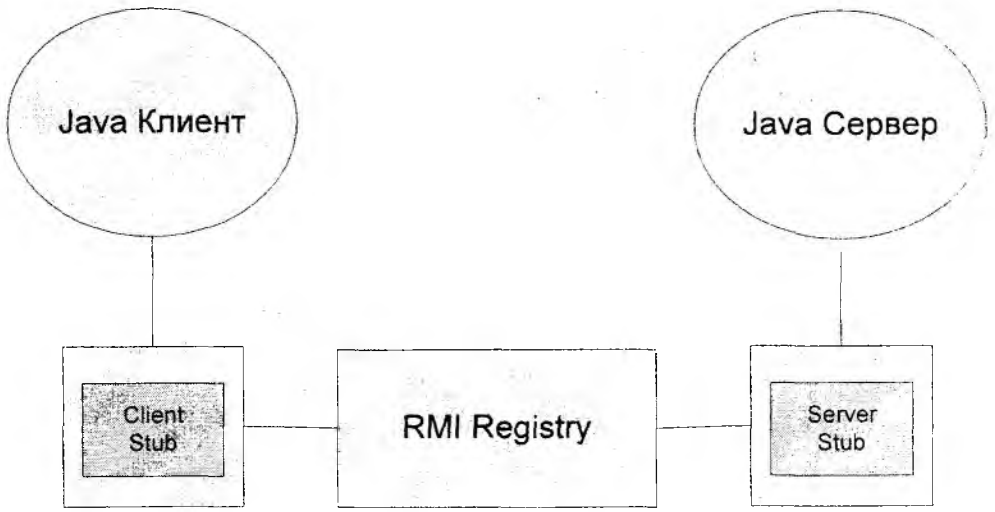

Рис. 3

Управление в RMI осуществляется посредством Client Stub (переходника для клиента) и Server Stub (переходника для сервера), которые порождены от общего интерфейса, но различие между ними в том, что client stub служит просто для подсоединения к RMI Registry, а server stub используется для связи непосредственно с функциями сервера.

Хотя RMI считается легковесной и менее мощной, чем CORBA и DCOM, тем не менее, она обладает рядом уникальных свойств, таких как распределенное, автоматическое управление объектами и возможность пересылать сами объекты от машине к машине. Существенными недостатками данной технологии является поддержка только одного языка – Java, сложность интеграции с существующими приложениями и плохая масштабируемость.

Однако технологии DCOM не хватало универсальности; даже применение CORBA зависело от реализации в продуктах разных поставщиков, появлялись новые объектные модели, не поддерживающие CORBA, интеграция по-прежнему реализовывалась на достаточно низком уровне, практически исключая возможность динамичного изменения связей между приложениями в ходе выполнения.

Все предлагаемые средства интеграции фокусировались на технологических особенностях реализации приложений и не позволяли учитывать специфику бизнес-процессов, в которых эти приложения использовались. Эти интеграционные проблемы и привели к появле-

нию идеи сервис-ориентированной архитектуры, основанной на уже существующей технологии Web-служб и протоколе SOAP.

Сервис-ориентиированная архитектура (англ. SOA, service-oriented architecture) – новый подход к созданию распределенных инфраструктур, в которых программные ресурсы рассматриваются как сервисы, предоставляемые по сети. В основе SOA лежат принципы многократного использования функциональных элементов, сервисов, которые имеют единый интерфейс и используют единый набор правил для определения того, как вызывать сервисы и как они будут взаимодействовать друг с другом. Использование стандартных интерфейсов позволяет прозрачно работать на основе разнообразных платформ и на их границах.[2]

На рис. 5 представлена общая схема управления SOA. Поставщик сервиса регистрирует свои сервисы в реестре, а потребитель обращается к реестру с запросом. Web-сервисы, которые базируются на широко распространенных и открытых протоколах (HTTP, XML, UDDI, WSDL и SOAP), занимают центральное место в SOA. Именно эти стандарты реализуют основные требования SOA – во-первых, сервис должен поддаваться динамическому обнаружению и вызову (UDDI, WSDL и SOAP), во-вторых, должен использоваться интерфейс, не зависящий от платформы (XML). HTTP обеспечивает функциональную совместимость и устанавливает отдельную TCP-сессию на каждый запрос.[3]

Рис. 5

Попытки реализовать архитектуру управления приложениями, в которой распределенные прикладные модули представлены как объекты, взаимодействующие посредством четко описанных интерфейсов, предпринимаются давно и с определенным успехом (CORBA и DCOM). Внешне они действительно похожи на SOA, но более детальный анализ показывает, что в этих ранних подходах к сервисной ориентации программных систем не выполняются или выполняются с определенными ограничениями базовые принципы SOA.

В CORBA и DCOM взаимодействующие объекты являются сильно связанными, и внесение изменений в управление компонентами, предоставляющими сервис, должно быть согласовано между собой. В DCOM взаимодействие управляющих компонентов основано на закрытых интерфейсах Microsoft. Кроме того, и CORBA, и DCOM имеют существенные ограничения по поддержке действительно распределенных систем, их протоколы взаимодействия объектов слишком сложны для организации связи сервисов, реализованных на удаленных машинах.

Основное достоинство SOA состоит в том, что такая архитектура позволяет строить слабо связанные приложения, то есть такие, в которых одну часть можно менять, не затрагивая другую. При этом SOA основывается на хорошо прописанных стандартах и сервисах.

В области распределенного искусственного интеллекта основное внимание сосредоточено на многоагентных системах (МАС), которые строятся из множества взаимодействующих агентов, совместно решающих поставленную задачу в распределенной среде. Агент – это объект, обладающий функциями приема информации из внешнего мира, воздействия на внешний мир и принятия решения о роде воздействия. Агенты предназначены для автоматического сбора, фильтрации и организации информации в РУС.

На рис. 6 приведен пример архитектуры МАС. Предполагается, что агенты расположены на разных серверах системы с интегрированными базами данных, расположенных в удаленных географических местах и взаимодействуют друг с другом при выполнении запроса, поступившего от пользователя.

В качестве агентов в данном случае выступают независимые исполняемые модули приложения, написанные на одном из языков сценариев. В серверную часть МАС входит ControlAgent, а в клиентскую часть – InterfaceAgent (Web-интерфейс). [4]

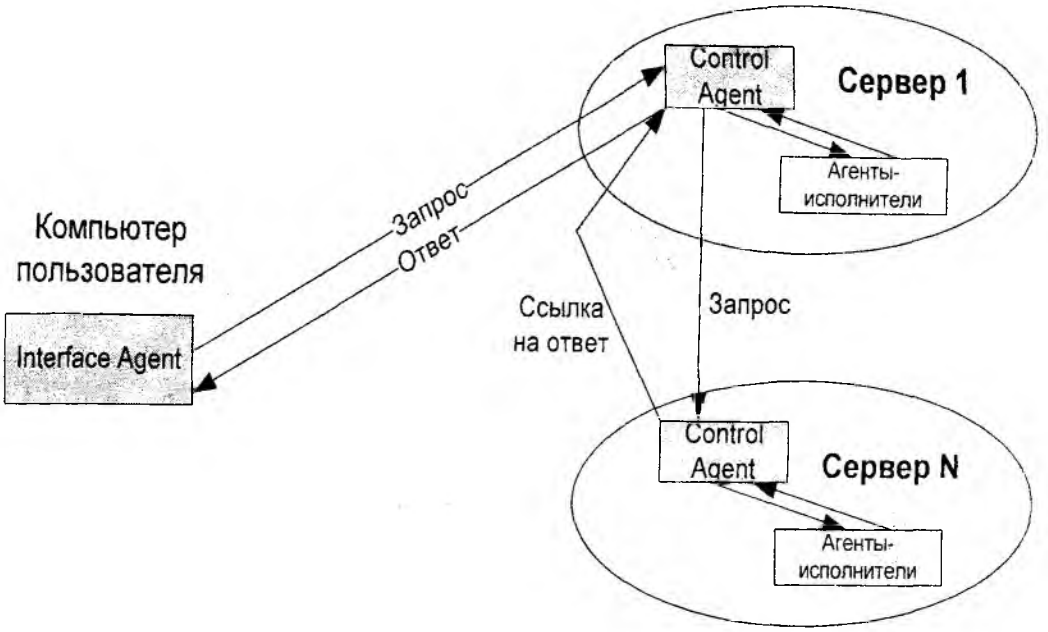

Рис. 6

Основными функциями InterfaceAgent являются: регистрация пользователя на сервере; прием от пользователя запроса на поиск данных во всех базах данных распределенной информационной системы и оказание ему помощи при формировании sql-запроса; установление связи с ControlAgent и передача ему запроса.

Основными функциями ControlAgent являются: прием от InterfaceAgent запросов на поиск данных в распределенной информационной системе; внесение запроса в очередь запросов, связь с другими серверами системы и передача им запросов.

Агент-исполнитель отбирает данные в базе данных сервера по запросу, полученному от ControlAgent; перекодирует информацию, получаемую из БД; записывает полученные данные в результирующий поток-файл; формирует общий ответ на запрос для отправки ответа пользователю.

В среде МАС общение между агентами осуществляется через Сокеты по стеку протоколов TCP/IP, а работа с базой данных осуществляется по протоколу ODBC.

Для анализа эффективности работы РУС на основе рассмотренных технологий актуальным является понятие «качество обслуживания» (Quality of Service, QoS), которое, трактуя более узко, включает только две самые важные характеристики сети – производительность и надежность.[5].

Производительность – это одно из основных свойств распределенных систем. Параметры производительности системы измеряются с помощью показателей двух типов – временных, оценивающих задержку, вносимую сетью при выполнении обмена данными, и показателей пропускной способности, отражающих количество информации, переданной сетью в единицу времени.

Важным свойством вычислительной сети также является надежность – способность правильно функционировать в течение продолжительного периода времени. Надежность формируется такими составляющими, как безотказность, долговечность, готовность, восстанавливаемость и сохраняемость.

Анализируя РУС, целесообразно выделить два уровня: передачи данных и прикладной. Уровень передачи данных рассматривает систему с точки зрения обмена данными между различными объектами.

К показателям качества уровня передачи данных относятся:

- пропускная способность;
- скорость передачи данных;
- вероятность безотказной работы;
- среднее время наработки на отказ.

В настоящее время уровень передачи данных достаточно изучен и существуют методы оценки показателей качества данного уровня.

Прикладной уровень рассматривает систему с точки зрения взаимодействия пользователя и сети, доступа приложений пользователя к сетевым службам, таким как обработчик запросов к базам данных, доступ к файлам и т.д. Также отвечает за передачу служебной информации, предоставляет приложениям информацию об ошибках и формирует запросы.

Для анализа эффективности РУС необходимо проводить оценку показателей качества обоих уровней. Результаты анализа рассмотренных выше технологий говорят о том, что основная работа с точки зрения взаимодействия сети и пользователя выполняется на прикладном уровне. Именно поэтому возникает необходимость в разработке методов оценки показателей качества данного уровня, таких как число обработанных запросов, время реакции, безопасность сети.

Список литературы: 1. *Роберт Дж. Оберг* Технология COM+. Основы и программирование Understanding and Programming COM+: A Practical Guide to Windows 2000 First Edition. М.: «Вильямс», 2000. 480с. 2. *Архитектура* SOA как она есть / Лори Маквитти. http://www.ccc.ru/magazine /depot/06_02/read.html. 3. *K. Channabasavaiah, K. Holley, E.M. Tuggle,* Migrating to a service-oriented architecture. IBM, December 2003 4. *Пономаренко Л.А., Филатов В.А., Е.Е. Цыбульник* Агентные технологии в задачах поиска информации и принятия решений // Управляющие системы и машины. 2003. №1. С. 36-41. 5. *Компьютерные сети. Принципы, технологии, протоколы* / Под ред. В.Г. Олифер., Н.А. Олифер. Санкт-Петербург: «Питер», 1999. 672 с.

Харьковский национальный университет радиоэлектроники

Поступила в редколлегию 03.10.2009