ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И ПРИКЛАДНАЯ МАТЕМАТИКА

УДК 681.317.75:519.2

В.П. Корячко, Д.А. Перепелкин, М.А. Иванчикова

АЛГОРИТМ АДАПТИВНОЙ МАРШРУТИЗАЦИИ В КОРПОРАТИВНЫХ СЕТЯХ НЕСКОЛЬКИХ ПРОВАЙДЕРОВ СВЯЗИ

Предложен алгоритм адаптивной маршрутизации, повышающий эффективность функционирования корпоративных сетей нескольких провайдеров связи.

***Ключевые слова**: адаптивная маршрутизация, алгоритмы маршрутизации, корпоративные сети, провайдеры связи.*

Введение. Цель работы – разработка нового эффективного алгоритма поиска оптимальных маршрутов в корпоративных сетях нескольких провайдеров связи, повышающего эффективность их функционирования.

В настоящее время большой класс распределенных и телекоммуникационных сетевых структур оптимально строить на основе базовых (опорных) сетей. На сегодняшний день такие задачи актуальны при формировании региональной телекоммуникационной инфраструктуры с учетом особенностей, действующих на территории региона операторов связи. Это дает возможность сократить общие затраты на создание высокоскоростных каналов связи, построить устойчивую сетевую структуру, к которой легко добавлять новые сегменты с учетом специфики предприятия. В связи с этим основное внимание уделяется эффективности применяемых в корпоративных сетях процессов маршрутизации.

Применение новых перспективных подходов для решения задачи маршрутизации позволяет повысить эффективность функционирования корпоративных сетей нескольких провайдеров связи за счет уменьшения затрат на их обслуживание и проектирование.

Теоретическая часть. Для повышения качества функционирования корпоративных сетей нескольких провайдеров связи наиболее важной задачей является выбор эффективного алгоритма маршрутизации. Для обеспечения этого условия необходимо сформировать и построить оптимальные маршруты передачи данных с учетом различных свойств той или иной корпоративной сети.

В общем случае корпоративная сеть состоит из множества базовых (коммуникационных) узлов, к которым подключаются пользователи сети, соединенных скоростными каналами связи. Стоимость, или цена, маршрута складывается из стоимости используемых каналов связи и базовых узлов.

Задача формирования и построения оптимальных маршрутов передачи данных часто формулируется как задача поиска кратчайших путей на графе. При этом вершинами графа являются базовые узлы, а ребрами – каналы связи, соединяющие эти базовые узлы. В настоящее время известно большое число алгоритмов поиска оптимальных маршрутов.

Для решения задачи поиска оптимальных маршрутов в корпоративных сетях широко используется алгоритм Дейкстры. Данный алгоритм применяется для построения таблиц маршрутизации в протоколе OSPF (Open Shortest Path First). Характеристики и параметры качества обслуживания протокола OSPF подробно рассматриваются в работах [1, 2 и 3].

В работах [1-9] предложены более эффективные алгоритмы построения оптимальных маршрутов передачи данных в динамических корпоративных сетях. Однако в этих работах рассматривается возможность построения эффективных схем маршрутизации при наличии одного оператора связи в корпоративной сети.

В реальных случаях задача поиска опти-

мальных маршрутов в корпоративной сети осложняется наличием нескольких альтернативных вариантов реализации, что приводит к необходимости стыковки каналов в узлах сети и возникновению зависимости стоимости узла от подключаемых к нему каналов связи. Примером может служить задача оптимального построения базовой сети региона при наличии нескольких операторов связи с различными зонами покрытия [10].

В общем случае для решения данной задачи применяется графовая модель корпоративной сети, в которой множество вершин графа соответствует множеству базовых узлов, а множество ребер соответствует возможным каналам связи между базовыми узлами. Каждое ребро, соответствующее каналу связи, также имеет свой вес. На практике весу ребра могут соответствовать стоимость аренды канала связи, затраты на оплату единицы трафика, передаваемого по каналу связи, соответствующему данному ребру, либо более сложная функция, учитывающая большее число параметров корпоративной сети. При нескольких провайдерах связи между отдельными базовыми узлами возможно наличие нескольких каналов связи, что соответствует нескольким ребрам, связывающим соответствующие вершины графа.

Трудоемкость построения таблиц маршрутизации с использованием классического алгоритма Дейкстры составляет величину $O(N^2)$, где N – число маршрутизаторов корпоративной сети. В общем случае данный алгоритм основан на присвоении вершинам временных пометок, определяющих верхнюю границу длины пути от начальной вершины до всех остальных, которые уменьшаются с помощью некоторой итерационной процедуры; после выполнения каждой итерации одна из пометок становится постоянной, что указывает на то, что данная пометка обозначает точную длину пути от начальной вершины до вершины с этой пометкой. Алгоритм заканчивает свою работу, когда просмотрены все вершины или когда найден путь от начальной вершины до искомой.

Трудоемкость построения таблиц маршрутизации корпоративных сетей нескольких провайдеров связи с применением алгоритма Дейкстры составляет величину порядка $O(N^3)$ за счет добавления цикла по всем вершинам для определения наименьшего канала связи между любой парой узлов сети.

Разработка новых, более эффективных алгоритмов адаптивной маршрутизации позволяет повысить качество функционирования корпоративных сетей нескольких провайдеров связи за счет уменьшения затрат на их обслуживание и проектирование.

Разработка алгоритма. Для повышения качества функционирования корпоративных сетей нескольких провайдеров предлагается эффективный алгоритм адаптивной маршрутизации, который при одинаковой трудоемкости построения оптимальных маршрутов передачи данных позволяет снизить стоимость их обслуживания и проектирования.

Корпоративную сеть нескольких провайдеров можно представить в виде мультиграфа. Мультиграф – это математическая модель, представляющая собой граф, в котором существует пара вершин, которая соединена более чем одним ребром (каналом связи). При этом ребра называются кратными.

Рассмотрим мультиграф корпоративной сети $G = (V, E, W)$, где V – множество вершин мультиграфа (базовых узлов или маршрутизаторов), E – множество ребер (каналов связи между базовыми узлами), причем $E = \bigcup E_i$ ($i = 1...k$, где k – количество провайдеров связи корпоративной сети, E_i – множество ребер i-го провайдера связи), W – множество весов ребер (стоимость каналов связи между базовыми узлами), $W = \bigcup W_i$, ($i = 1...k$, W_i – множество весов ребер i-го провайдера связи).

Каждая вершина графа v_i также имеет свой вес $s(v_i)$. Вес вершины может определяться, например, числом пользователей, которые подключаются к узлу, которому соответствует данная вершина. Однако в случае подключения к узлу каналов связи нескольких провайдеров он выполняет функции по коммутации этих каналов, что, как правило, требует дополнительных затрат. Поэтому вес вершины должен учитывать этот фактор, что в нашем случае будет выглядеть следующим образом:

$$s(v_i) = \sum_{i=1}^{k} s_i(v_j),$$

где $s_i(v_j)$ – вес вершины j, связанный с обслуживанием каналов провайдера связи с номером i.

На рисунке 1 представлена корпоративная сеть нескольких провайдеров связи в виде мультиграфа.

В данном мультиграфе корпоративной сети нескольких провайдеров связи веса вершин по каждому провайдеру связи определены следующим образом:

$S^1 = \{30, 50, 30, 10, 40, 60, 20, 40\}$;

$S^2 = \{20, 10, 40, 50, 40, 90, 60, 20\}$.

При использовании классического алгоритма Дейкстры для сетей нескольких провайдеров

связи на мультиграфе G учитываются веса вершин и ребра минимального веса. Результат решения задачи поиска оптимальных маршрутов мультиграфа G до всех вершин множества $V_{\hat{b}} = V \setminus \{v_b\}$ из начальной вершины v_b, т.е. дерево кратчайших путей с корнем в вершине v_b, построено жирными линиями на рисунке 2.

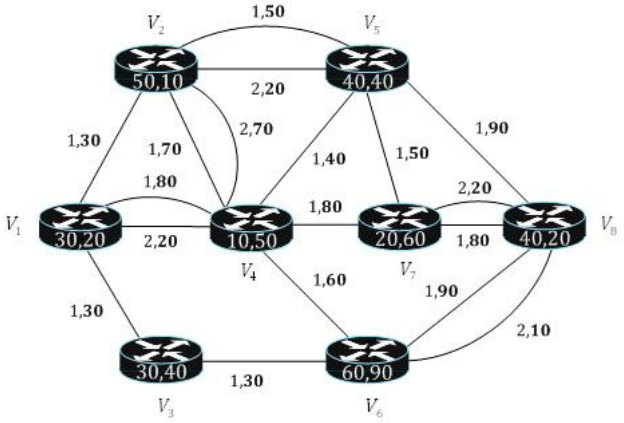

Рисунок 1 – Мультиграф G корпоративной сети нескольких провайдеров связи

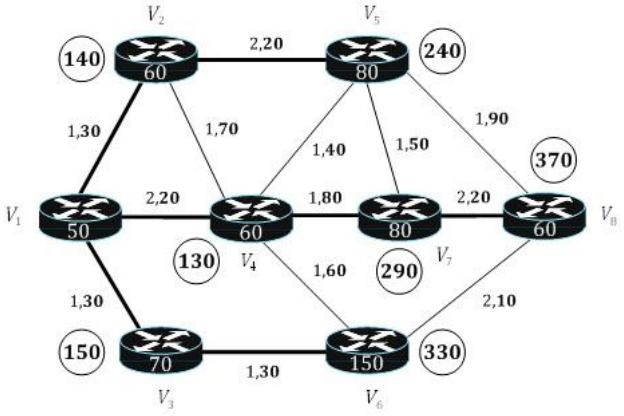

Рисунок 2 – Результат работы алгоритма Дейкстры для мультиграфа G корпоративной сети

То есть оптимальный маршрут до вершины V_2 составит $\pi_2 = \{e_{1,2}\}$, а его оценка $d_2 = 140$. Оптимальный маршрут до вершины V_3 составит $\pi_3 = \{e_{1,3}\}$, а его оценка $d_3 = 150$. Оптимальный маршрут до вершины V_4 составит $\pi_4 = \{e_{1,4}\}$, а его оценка $d_4 = 130$. Оптимальный маршрут до вершины V_5 составит $\pi_5 = \{e_{1,2}, e_{2,5}\}$, а его оценка $d_5 = 240$. Оптимальный маршрут до вершины V_6 составит $\pi_6 = \{e_{1,3}, {}_{3,6}\}$, а его оценка $d_6 = 330$. Оптимальный маршрут до вершины V_7 составит $\pi_7 = \{e_{1,4}, e_{4,7}\}$, а его оценка $d_7 = 290$. Оптимальный маршрут до вершины V_8 составит $\pi_8 = \{e_{1,4}, e_{4,7}, e_{7,8}\}$, а его оценка $d_8 = 370$.

Для разработки алгоритма адаптивной маршрутизации в корпоративных сетях нескольких провайдеров связи с учетом приведенных ранее положений необходимо ввести временные метки по каждому провайдеру и уменьшать их в соответствии с классическим алгоритмом Дейкстры, отличие будет заключаться в выборе постоянной пометки, равной

$$\min_{i=1..K} \left\{ d_j^i \right\},$$

где d – метка вершины с номером j по провайдеру связи с номером i ($i = 1...k$, $j = 1...N$, где k – число провайдеров связи, N – число вершин).

Рассмотрим работу алгоритма адаптивной маршрутизации в корпоративных сетях нескольких провайдеров связи. Укрупненно алгоритм имеет следующий вид.

Пусть d_j^i – пометка вершины v_j по провайдеру i, v_b – начальная вершина, v_p – текущая вершина.

Шаг 1. Присвоение начальных значений. Положить $d^i*(v_b) = s^i$ и считать эту пометку постоянной. Принять $d^i(v_j) = \infty$ (бесконечность) для всех $v_j \neq v_b$ и считать эти пометки временными. Присвоить $p = b$.

Шаг 2. Обновление пометок. Для всех вершин $v_j \in G(v_p)$, инцидентных вершине v_p и имеющих временные пометки, изменить пометки в соответствии с выражением:

$$d_j^i = \min_{i=1..K} (d_j^i, \min_{i=1..K} \left\{ d_p^i \right\} + w_{pj}^i + s_p^i + s_j^i), \qquad (1)$$

если d_p^i не включает стоимость s_p^i i – го провайдера связи или по выражению:

$$d_j^i = \min_{i=1..K} (d_j^i, \min_{i=1..K} \left\{ d_p^i \right\} + w_{pj}^i + s_j^i), \qquad (2)$$

если d_p^i уже включает стоимость s_p^i i-го провайдера связи, где d_p^i – метка текущей вершины, d_j^i – метка вершины, смежной с текущей вершиной, w_{pj}^i – вес ребра, соединяющего вершину v_p с вершиной v_j.

Шаг 3. Превращение пометки в постоянную. Среди всех вершин с временными пометками найти такую, для которой

$$d_j^* = \min_{i=1..K} \left\{ d_j^i \right\}. \qquad (3)$$

Шаг 4. Считать пометку вершины v_j* постоянной и положить $v_p = v_j^*$.

Шаг 5. Если все вершины имеют постоянные пометки, то конец алгоритма. Иначе перейти к шагу 2.

Сам маршрут можно найти, применяя рекурсивно процедуру, в которой реализовано выражение:

$$d_j^* = \min_{i=1..K} \left\{ d_j^{i*} \right\} + w_{jk}^i + s_p^i + s_j^i.$$

Оно справедливо для последующей и предыдущей вершин, принадлежащих одному пути. Кроме того, если узел связи имеет каналы нескольких провайдеров связи, то необходимо изменить длину пути по следующему выражению:

$$d_j^{**} = d_j^{*} + s_j^{i}.$$

На рисунке 3 жирными линиями показано дерево кратчайших путей после применения предложенного алгоритма адаптивной маршрутизации нескольких провайдеров связи.

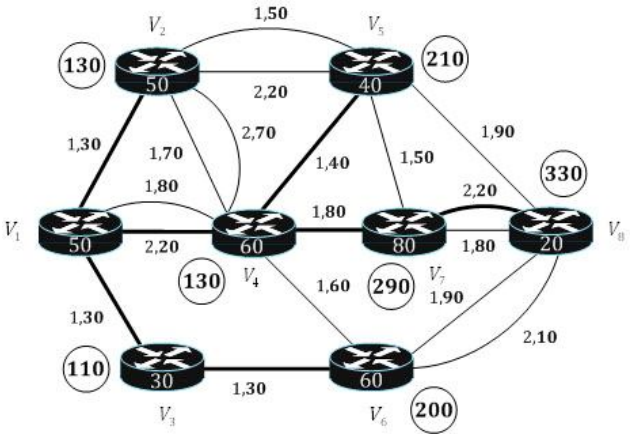

Рисунок 3 – Результат работы предложенного алгоритма адаптивной маршрутизации

Из рисунка 3 видно, что оценки длин оптимальных маршрутов до вершин V_2, V_3, V_5, V_6 и V_8 уменьшились. Таким образом, оптимальный маршрут до вершины V_2 составит $\pi_2 = \{e_{1,2}\}$, а его оценка $d_2 = 130$. Оптимальный маршрут до вершины V_3 составит $\pi_3 = \{e_{1,3}\}$, а его оценка $d_3 = 110$. Оптимальный маршрут до вершины V_4 составит $\pi_4 = \{e_{1,4}\}$, а его оценка $d_4 = 130$. Оптимальный маршрут до вершины V_5 составит $\pi_5 = \{e_{1,4}, e_{4,5}\}$, а его оценка $d_5 = 210$. Оптимальный маршрут до вершины V_6 составит $\pi_6 = \{e_{1,3}, e_{3,6}\}$, а его оценка $d_6 = 200$. Оптимальный маршрут до вершины V_7 составит $\pi_7 = \{e_{1,4}, e_{4,7}\}$, а его оценка $d_7 = 290$. Оптимальный маршрут до вершины V_8 составит $\pi_8 = \{e_{1,4}, e_{4,7}, e_{7,8}\}$, а его оценка $d_8 = 330$.

Результаты моделирования. Для подтверждения правильности предложенного алгоритма адаптивной маршрутизации в корпоративных сетях нескольких провайдеров связи разработано программное обеспечение моделирования процессов маршрутизации.

При разработке основное внимание уделялось корректности предлагаемого алгоритма и размерности решаемой задачи.

Были проведены исследования графовых моделей корпоративных сетей, состоящих из 10, 50 и 100 вершин с учетом двух провайдеров связи. Исследование разработанного алгоритма адаптивной маршрутизации в корпоративных сетях нескольких провайдеров связи показало, что трудоемкость построения оптимальных маршрутов передачи данных сопоставима с алгоритмом Дейкстры и составляет величину порядка $O(N^3)$. Однако предложенный алгоритм адаптивной маршрутизации позволяет уменьшить стоимость аренды каналов связи и затраты на оплату единицы трафика по каждому из маршрутов за счет исключения из стоимости маршрутов – стоимости неиспользуемых каналов и узлов связи.

На рисунке 4 представлены результаты моделирования алгоритма адаптивной ускоренной маршрутизации в корпоративных сетях нескольких провайдеров связи для приведенного ранее мультиграфа G.

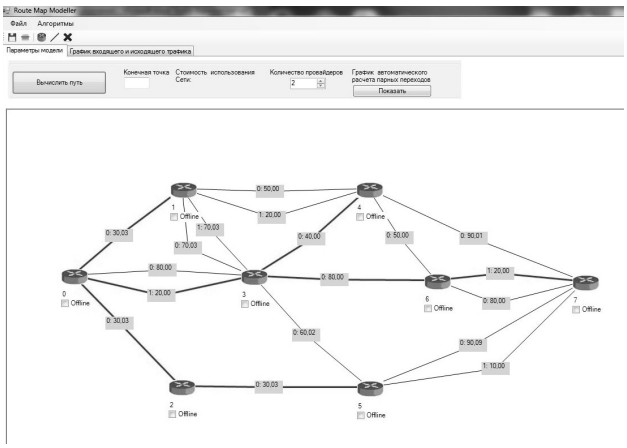

Рисунок 4 – Результат программной работы алгоритма адаптивной маршрутизации

На основе этого можно сделать вывод, что предложенный алгоритм адаптивной маршрутизации является эффективным при поиске оптимальных маршрутов в корпоративных сетях нескольких провайдеров связи.

Заключение. Разработанный алгоритм адаптивной маршрутизации позволяет повысить эффективность функционирования корпоративных сетей нескольких провайдеров связи за счет оптимального построения маршрутов передачи данных и исключения из их стоимости – стоимости неиспользуемых каналов и узлов связи.

Библиографический список

1. *Корячко В.П., Перепелкин Д.А.* Корпоративные сети: технологии, протоколы, алгоритмы. – М.: Горячая линия – Телеком, 2011. 219 с.

2. *Перепелкин Д.А.* Алгоритм адаптивной ускоренной маршрутизации на базе протокола OSPF при динамическом добавлении элементов корпоративной сети // Вестник Рязанского государственного радиотехнического университета. № 4 (выпуск 34), 2010. С. 65-71.

3. *Корячко В.П., Перепелкин Д.А.* Анализ и проектирование маршрутов передачи данных в кор-

поративных сетях. – М.: Горячая линия – Телеком, 2012. 235 с.

4. *Перепелкин Д.А., Перепелкин А.И.* Разработка алгоритмов адаптивной маршрутизации в корпоративных вычислительных сетях // Вестник Рязанской государственной радиотехнической академии. № 19, 2006. С. 114-116.

5. *Перепелкин Д.А., Перепелкин А.И.* Разработка алгоритма динамической маршрутизации на базе протокола OSPF в корпоративных вычислительных сетях // Вестник Рязанского государственного радиотехнического университета. № 2 (выпуск 28), 2009. С. 68-72.

6. *Корячко В.П., Перепелкин Д.А., Перепелкин А.И.* Алгоритм парных перестановок маршрутов в корпоративных сетях // Системы управления и информационные технологии. № 2 (выпуск 40), 2010. С. 51-56.

7. *Корячко В.П., Перепелкин Д.А., Перепелкин А.И.* Повышение эффективности функционирования корпоративных сетей при динамических изменениях в их структуре и нагрузках на линии связи // Вестник Рязанского государственного радиотехнического университета. № 3 (выпуск 33), 2010. С. 49-55.

8. *Перепелкин Д.А., Перепелкин А.И.* Алгоритм адаптивной ускоренной маршрутизации в условии динамически изменяющихся нагрузок на линиях связи корпоративной сети // Информационные технологии. № 3, 2011. С. 2-7.

9. *Перепелкин Д.А.* Алгоритм адаптивной ускоренной маршрутизации на базе протокола OSPF при динамическом отказе элементов корпоративной сети // Вестник Рязанского государственного радиотехнического университета. № 3 (выпуск 37), 2011. С. 53-58.

10. *Горшков С.Г., Никитин Е.В., Саксонов Е.А.* Задача формирования структуры базовой сети // Вестник ТОГУ, 2010. С. 59-66.

УДК 004.422.81

В.И. Побаруев

ОРГАНИЗАЦИЯ МЕЖПРОГРАММНОГО ВЗАИМОДЕЙСТВИЯ В СЛОЖНЫХ СИСТЕМАХ МЕТОДОМ «АССОЦИАТИВНОГО» ИНТЕРФЕЙСА

Рассматривается метод обеспечения взаимодействия между компонентами сложных программных систем, который основан на организации ассоциативных связей. Метод позволяет создавать наследуемые, быстродействующие и легко реализуемые средства межпрограммного взаимодействия. Метод прошел широкую апробацию благодаря его многолетнему использованию при реализации интерфейсов в крупных программных комплексах обработки данных дистанционного зондирования, разрабатываемых в НИИ «Фотон».

Ключевые слова*: интерфейс, модульность, компонент, межпрограммное взаимодействие, точка входа, бинарная совместимость.*

Введение. Нынешний мировой рынок продаж космических снимков измеряется миллиардами долларов в год и растет примерно на 20 % ежегодно. С ростом числа космических систем дистанционного зондирования Земли (ДЗЗ) происходит постоянное увеличение объемов поступающей на обработку информации. Причем увеличивается как количество снимков, так и их информационная емкость, что автоматически ведет к увеличению практических применений полученных данных.

В настоящее время успешно эксплуатируется значительное количество систем ДЗЗ, главным образом, природно-ресурсной, океанографической и метеорологической ориентации, во многом схожих по идеологии построения и функционирования [1-3]. Анализ существующих и перспективных систем ДЗЗ позволяет выявить определенные тенденции в их развитии.

Первая тенденция связана с неуклонным ростом пространственного разрешения сканирующих устройств, что естественным образом расширяет возможности дистанционного зондирования по оценке состояния все более мелких объектов на земной поверхности.

Вторая тенденция связана с получением гиперспектральных данных в видимом и инфракрасном (ИК) диапазонах спектра с высоким радиометрическим разрешением, на основе которых выявляются разного рода «тонкие» эффекты природных явлений.

Третья тенденция предполагает повышение оперативности съемки для более эффективного решения задач мониторинга окружающей среды

и ликвидации последствий чрезвычайных происшествий.

Общим для перечисленных тенденций является постоянный рост объемов, скоростей, видов поступающих видеоданных, а также областей их применения, что ставит задачи по повышению производительности и гибкости средств обработки. Разрабатываемые программные системы не могут представлять застывшее монолитное приложение [4]. Необходимо постоянно учитывать появление новых сканирующих устройств, изменения в операционных системах, аппаратуре и новые требования со стороны пользователей. В этих условиях спроектировать сложную систему можно только при грамотном решении вопросов, связанных с декомпозицией, то есть разложением единой системы на отдельные части или компоненты. В этой связи одним из наиболее важных аспектов проектирования сложных программных средств обработки является способ межпрограммного взаимодействия отдельных компонентов (интерфейс). Интерфейс межпрограммного взаимодействия определяет такие характеристики системы обработки, как:

– гибкость (адаптируемость) к различным типам задач и данных;

– производительность;

– возможность организации повторного и многократного использования компонентов;

– простота освоения, что обеспечивает возможность быстрой разработки приложений.

Цель работы – разработка базового метода межпрограммного взаимодействия, который позволяет строить на его основе крупные программные системы обработки данных ДЗЗ и учитывает специфику и особенности развития средств дистанционного зондирования. Предлагаемый в статье метод определяет двоичный (бинарный) интерфейс приложений, который является некритичным к способу его практической реализации, использованию языка программирования или операционной системы.

Анализ проблемной ситуации. В настоящее время выбор стандартов построения бинарного интерфейса приложений невелик. Одним из наиболее популярных инструментов, используемых для этих целей в среде Windows, является COM – модель компонентных объектов (Component Object Model) [5], предложенная фирмой Microsoft. COM и его дальнейшие развития (COM+, ActiveX) – это спецификация для создания бинарно-совместимых компонентов программного обеспечения.

Однако при более глубоком изучении COM выяснилось, что данная модель построения многокомпонентного приложения применительно к разработке высокопроизводительных систем (особенно, при разработке систем обработки данных ДЗЗ) имеет существенные недостатки.

Во-первых, COM не поддерживает наследование. При добавлении новых функций в компонент необходимо создание нового уникального интерфейса [5] и поддержка старого, что не только замедляет и усложняет программирование (необходимо отслеживать версии и синхронизацию компонентов), но и приводит к непомерному «разбуханию» программного модуля из-за необходимости поддержки совместимости со старыми интерфейсами.

Во-вторых, приложение с COM-компонентами сложно отлаживать (в случае когда требуется отладка всего программного комплекса, включая компоненты).

В-третьих, в COM отсутствует непосредственная двусторонняя связь с конкретным экземпляром приложения (COM-сервера или COM-клиента), то есть нельзя просто получить двусторонний интерфейс для конкретного запущенного процесса (необходимо использовать механизм транспортировки – marshalling).

В-четвертых, COM не позволяет использовать в одной системе несколько версий компонентов с одинаковым интерфейсом (одинаковым GUID – глобально уникальным идентификатором), из-за чего нельзя размещать на одном компьютере две одинаковые программные системы разных версий (что часто требуется для отладки и выявления ошибок).

Предлагаемое решение. Принципиальные недостатки COM обусловили необходимость разработки другой спецификации межпрограммного взаимодействия, ориентированной на построение систем обработки данных ДЗЗ. Новый протокол межмодульного взаимодействия, назовем его «Ассоциативный интерфейс» (Associative Interface – AI), основан на получении доступа к интерфейсу (совокупности методов и данных для взаимодействия) посредством уникальных имен, в качестве которых могут выступать уникальные текстовые строки или GUID, используемый в COM. В отличие от COM в AI используются не совокупности методов (которые могут быть реализованы с помощью классов, например на языке C++), а отдельные элементы – указатели на функции и структуры данных. На основе этих отдельных компонентов формируются классы интерфейса применительно для конкретного языка программирования, то есть единый программный код, включаемый во все модули системы и обеспечивающий их взаимодействие (рисунок 1).

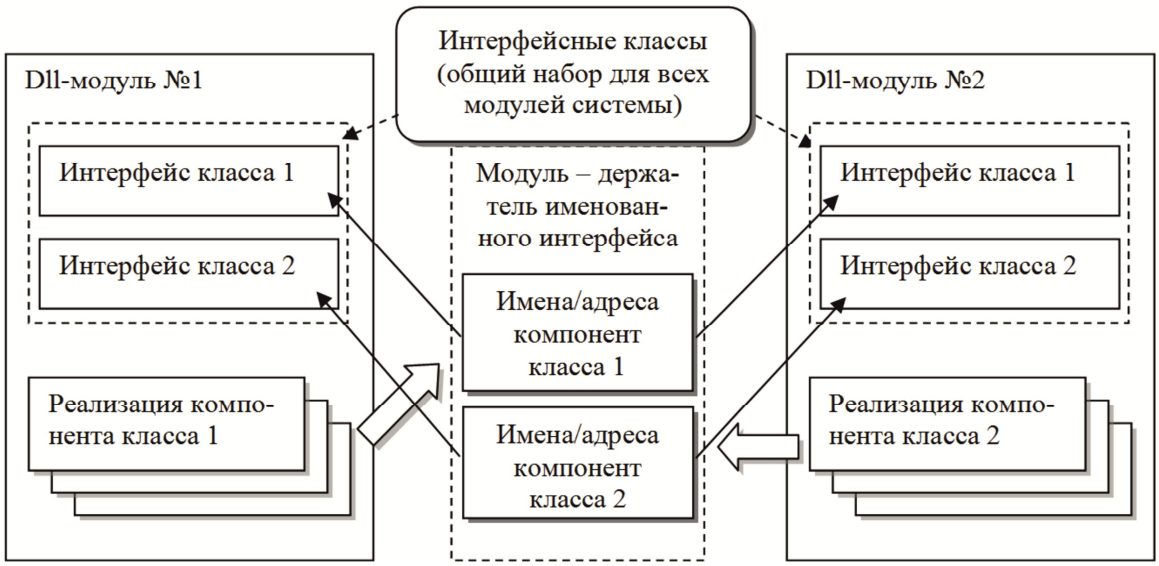

Рисунок 1 – Схема организации межпрограммного взаимодействия посредством интерфейсных классов

Благодаря этому при добавлении нового элемента (функции или структуры данных) в интерфейс он автоматически расширяется без дополнительных изменений. То есть в интерфейсный класс просто добавляется новый метод (в COM же в этом случае требуется создание нового интерфейса), посредством чего реализуется наследование. Разумеется, для использования нового метода в интерфейсе требуется перекомпиляция модуля, однако остальные модули со «старым» вариантом интерфейса остаются полностью совместимы с новым.

Рассмотрим реализацию подобного интерфейса на практике с использованием языка программирования C++. Интерфейс представляет собой набор адресов функций и структур данных, каждому из которых присвоено уникальное имя. В программной системе должен присутствовать модуль, который обеспечивает управление ассоциированными адресами (содержит таблицу имен и обеспечивает их управление).

Собственно управление интерфейсом обеспечивается с помощью трех базовых функций:

– получения именованного адреса;

– добавления именованного адреса;

– удаления именованного адреса.

Приведем прототипы данных методов на языке C++.

*void *GetNamedAddress(char *AddressName); //Получение адреса*

*void AddNamedAddress(void *Address,char *AddressName); //Добавление адреса*

*void DelNamedAddress(char *AddressName); //Удаление адреса*

При данной реализации структура программной системы должна иметь следующий вид (рисунок 2).

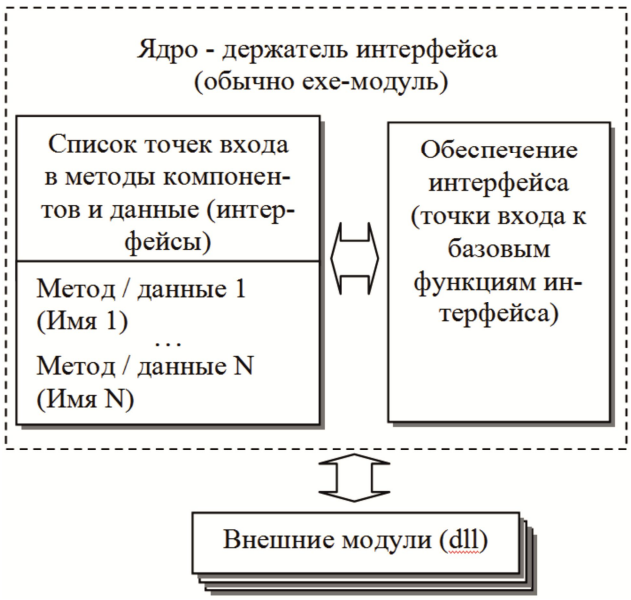

Рисунок 2 – Структура программной системы с именованным интерфейсом

Базовые функции реализуют «обеспечение интерфейса» и позволяют получить адрес процедуры или структуры данных (интерфейсного компонента), добавить интерфейсный компонент или удалить интерфейсный компонент с использованием его имени (уникального строкового идентификатора). Методы интерфейса (компоненты), к которым необходимо получить доступ, на языке C++ в общем виде описываются следующим образом:

```
TYPE Fun1(Arg1, Arg2, …, ArgN)
{
  static TYPE (*fp)(Arg1, Arg2, …,
ArgN)=NULL;
  if (!fp)  fp=( TYPE (*fp)(Arg1, Arg2, …,
ArgN)) GetNamedAddress(“Name”);
  return fp(Arg1, Arg2, …, ArgN);
}
```

где TYPE – тип возвращаемого значения функции, Arg1, Arg2, …, ArgN – список аргументов функции, «Name» – уникальный строчный идентификатор получаемого интерфейса.

Организованный таким образом компонент интерфейса программно доступен как обычная функция, что не требует никаких дополнительных инициализаций, обеспечивает простоту использования, высокую скорость выполнения (на уровне вызова локальной подпрограммы) и легкость отладки. Все компоненты программного комплекса (dll-модули) являются равноправными и могут добавлять и использовать интерфейсы (точки входа) в зависимости от поставленной задачи. Добавление и изменение интерфейсных компонентов в программной системе может производиться динамически в ходе выполнения (путем непосредственного вызова функции *«AddNamedAddress» / «GetNamedAddress»)*, что позволяет создавать интерфейсы с временем жизни выполнения процедуры обработки. Одна и та же интерфейсная функция может иметь разную реализацию (и меняться несколько раз в ходе выполнения программы) в зависимости от выполняемой задачи. Например, вывод отчета может производиться на экран, на принтер или в файл, иметь разную форму, а представляться одним прототипом, например *void Report(char *cReportText)*.

Объединение отдельных компонентов интерфейса в класс позволяет получить интерфейсный класс для взаимодействия с модулем – держателем компонента. При расширении функциональности компонента необходимо также расширять и интерфейсный класс. При этом сохранится работоспособность ранее созданных DLL-модулей с использованием предыдущей версии интерфейсного класса.

Техническая реализация «точек входа» к базовым функциям, например в операционной системе Windows, может быть выполнена посредством резервирования фиксированного адресного пространства для указателей перехода к базовым функциям именованного интерфейса. Такой подход обеспечивает доступ к интерфейсу еще до этапа вызова функции DllMain() в динамически подключаемой библиотеке, то есть интерфейсные функции могут использоваться для инициализации глобальных объектов dll-модуля (рисунок 3).

Рисунок 3 – Расположение точек входа базовых функций ассоциативного интерфейса в адресном пространстве на примере WIN32-приложения

Область адресов точек входа для базовых функций именованного интерфейса является фиксированной для конкретной реализации программной системы. Данная область может быть зарезервирована и защищена от модификации, например, функцией «VirtualAlloc» Windows API. Константный адрес области точек входа передается модулям вместе с исходным кодом интерфейсных классов. Благодаря этому не требуется инициализация базового интерфейса и обеспечивается его непосредственное использование на любом этапе выполнения программы (в том числе и до вызова функции DllMain в динамически подключаемой библиотеке).

Представленная схема взаимодействия является более гибкой, чем COM. В отличие от COM, она обеспечивает возможность расширения функций системы путем простого добавления компонентов (копирование DLL-модуля в определенный каталог), о которых ядро системы даже не «знает». Технически это реализуется посредством сканирования DLL-модулей в заранее заданном каталоге при запуске системы. При инициализации DLL компоненты сами регистрируют свои интерфейсные функции. Например, если требуется расширить функции системы для обработки данных от новой системы ДЗЗ, то необходимо в каталог модулей «импорта» скопировать DLL-модуль распаковки информации от нового датчика и запустить систему. То есть без изменения всего программного комплекса у него появляется возможность обработки данных нового типа.

Заключение. На основе «Ассоциативного интерфейса» реализованы десятки сложных программных систем обработки данных ДЗЗ, в том

числе и системы реального времени. Эти системы создавались по хоздоговорам с предприятиями Федерального космического агентства: Российским НИИ Космического приборостроения, ЦНИИМаш, Космическим научнопроизводственным центром им. М.В.Хруничева, Ракетно-космическим центром «ЦСКБ-Прогресс», Научным центром оперативного мониторинга Земли, НИИ точных приборов, а также по программам и грантам Минобрнауки России.

В рамках космических проектов «Ресурс-О1», «Океан-О», «Мир», «Метеор-3М», «Метеор-М», «Ресурс-ДК», «Монитор-Э», «Сич-1М», «Аркон», «Ресурс-ДК», «Канопус-В» и др. созданы уникальные системы межотраслевой обработки видеоинформации, принимаемой от природно-ресурсных, океанографических и метеорологических спутниковых систем (программные системы NormSat, NormScan, BankSat, TmVision, PlanetaMeteo, NormSatReg, GeoScan, OrthoPlan, NormSatB и др.) [6]. Данные программные продукты имеют единое ядро «ER-Set» [7, 8] и различаются лишь набором компонентов обработки и пользовательским интерфейсом, настроенным для реализации конкретных задач. Большинство компонентов используются в разных комплексах повторно. Успешная разработка целого семейства программных изделий, их внедрение, положительные отзывы заказчиков и длительная эксплуатация в Российских и зарубежных (EADS France, Армения) центрах обработки данных дистанционного зондирования подтверждает высокую степень применимости разработанного метода межпрограммного взаимодействия для построения крупных программных систем.

Библиографический список

1. *Хижниченко В.И.* Дистанционное зондирование Земли. Обзорная информация // Российское авиа ционно-космическое агентство / СПб.: Гидрометеоиздат, 2000. - 80 с.

2. *Гарбук С.В., Гершензон В.Е.* Космические системы дистанционного зондирования Земли. // М.: Издательство А и Б, 1997. - 296 с.

3. *Вебер Р.А., О'Коннелл К.М.* [Элестронный ресурс]: Американская коммерческая спутниковая съемка в 2020 г., варианты будущего. 2012. – URL: www.gisa.ru/file/file2201.doc.

4. Временные требования к использованию материалов дистанционного зондирования Земли при проведении мониторинга экзогенных геологических процессов в составе государственного мониторинга состояния недр / под ред. *Кочеткова М.В.* – М.: ЗАО «Геоинформмарк», 2000. - 52 с.

5. *Эш Рофэйл, Яссер Шохауд.* COM и COM+. Полное руководство: пер. с англ. – К.: ВЕК +, К.: НТИ, М.: Энтроп, 2000. – 560 с., ил.

6. *Гомозов О.А., Еремеев В.В., Кузнецов А.Е., Лось В.В., Пресняков О.А., Соловьева К.К.* Алгоритмы и технологии обработки информации от КА «Ресурс-ДК»: Современные проблемы дистанционного зондирования Земли из космоса: Физические основы, методы и технологии мониторинга окружающей среды, потенциально опасных явлений и объектов. Сборник научных статей. Выпуск 5. том I. – М.: ООО «Азбука-2000», 2008. – С. 69 – 76.

7. *Побаруев В.И., Пресняков О.А.* Унифицированная система для построения растровых ГИС: Тез. докл. 5-ой Международной научно-технической конференции «Космонавтика. Радиоэлектроника. Геоинформатика». РГРТА. Рязань, 2007. – С. 331 – 334.

8. Универсальная платформа быстрой разработки приложений для обработки данных дистанционного зондирования Земли «ER-Set»: свидетельство о государственной регистрации программы для ЭВМ № 2013610968, Рос. Федерация: *В.И. Побаруев*; заявитель и правообладатель ФГБОУ ВПО «Рязанский государственный радиотехнический университет»; заявл. 04.12.2012; опубл. 09.01.2013.