ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И ПРИКЛАДНАЯ МАТЕМАТИКА

УДК 004.724

В.П. Корячко, О.В. Лукьянов, А.П. Шибанов

НАХОЖДЕНИЕ СКРЫТОГО ПАРАЛЛЕЛИЗМА ПРОТОКОЛОВ ДЛЯ УЛУЧШЕНИЯ ХАРАКТЕРИСТИК СЕТИ ПЕРЕДАЧИ ДАННЫХ ПОЛИГОННОГО ИЗМЕРИТЕЛЬНОГО КОМПЛЕКСА

Рассматриваются вопросы улучшения показателей качества полигонной сети при передаче трафика от нескольких измерительных систем. Предлагается метод улучшения характеристик протоколов передачи информации путем выявления в них скрытого параллелизма и моделирования с использованием GERT-сетей (Graphical Evaluation and Review Technique) и системы массового обслуживания (СМО) M/M/1 – M/G/1.

Ключевые слова: *измерительный комплекс, GERT-сети, система массового обслуживания, синхронизация протокола, плотность распределения вероятностей.*

Введение. Рассматривается система сбора, предварительной обработки и передачи информации полигонного измерительного комплекса (ПИК). Система является двухфазной. В первой фазе выполняется сбор информации в специализированном процессоре (СП) от нескольких источников слежения за объектами испытаний, причем каждый источник передает в систему пуассоновский поток заявок. Эти потоки агрегируются, после чего выполняются операции предварительной обработки информации, связанные с отбраковкой недостоверных значений, прореживанием избыточной информации, ее сжатием, шифрованием и т. п. Интервал времени, отводимый на предварительную обработку, характеризуется экспоненциальным распределением. После этого СП передает информацию в канал, связанный двумя маршрутизаторами [1, 2, 3]. Канал работает под управлением протокола передачи пакетов, работа которого отражается полумарковской моделью, в которой время выполнения отдельных операций протокола характеризуется независимыми случайными величинами с известным законом распределения. Сетевая операционная система выполняет либо заданную последовательность операций, либо после выполнения очередной операции производится вероятностный выбор следующей операции. Таким образом, в модели допускаются ветвления и циклы. В современных процессорах и микроконтроллерах имеются специальные регистры, в которых предварительно может быть собрана статистическая информация как о длительностях выполнения операций, так и вероятностях ветвлений. Современные сетевые протоколы имеют достаточно сложную структуру, при этом в них часто реализуются альтернативные параллельные простые пути от входа к выходу алгоритма (или программы) с взаимными переходами между ними [4, 5]. То есть в полумарковской модели имеется несколько простых путей от источника s к стоку t, которые могут заметно отличаться друг от друга случайным временем прохождения пакетов. Если это так, то распределение времени передачи пакета между маршрутизаторами в общем случае будет многомодовым. Это является причиной увеличения очередей на входе канала и даже возможной потери пакетов при заполнении входных буферов. Следствием этого является и появление «джиттера» – колебаний времени передачи пакетов относительно среднего значения. При передаче синхронной информации (чувствительной к задержкам и их колебаниям) на величину «джиттера» накладываются достаточно жесткие ограничения. В противном случае появляются серьезные искажения при передаче изображений и речи, неприемлемые задержки при передаче команд управления на объекты и т.д.

Цель работы. Необходимо в достаточно сложном вероятностном графе протокола: 1) найти функции распределения всех простых $s-t$-путей; 2) вычислить вероятности их прохождения; 3) найти те операции в простых $s-t$-путях, которые вносят наибольший вклад в суммарные колебания времени передачи; 4) найти способы их уменьшения до необходимых величин. Назовем эти действия синхронизацией протокола передачи пакетов.

Улучшение характеристик протокола возможно, если время прохождения хотя бы одного параллельного простого $s-t$-пути характеризуется: 1) одномодовым распределением с сильно затянутым "хвостом"; 2) многомодовым распределением вероятностей.

Причинами появления затянутых "хвостов" являются повторно выполняемые операции, например пересылки искаженных пакетов. Чем меньше вероятность таких событий, тем более симметричной становится плотность распределения. Уменьшения длины "хвостов" распределений можно добиться: 1) уменьшением вероятности переходов на возвратные дуги, 2) уменьшением времени прохождения этих дуг путем повышения качества работы соответствующих компонент системы.

Появление нескольких мод объясняется скрытым или явным параллелизмом, присущим протоколам и алгоритмам передачи информации. Для каждого из параллельных простых $s-t$-путей можно выделить интервалы времени некоторой длины, в которые попадает большинство случайных исходов эксперимента. Если вышеуказанные интервалы не пересекаются друг с другом, то плотность распределения будет иметь число мод, равное числу интервалов, в которые попадает большинство случайных исходов. Таким образом, необходимо изменить параметры протокола для получения одномодового распределения времени передачи пакета с заданными характеристиками математического ожидания и среднеквадратического отклонения.

Постановка задачи. Режим работы системы стационарный. На вход СП подается n простейших потоков заявок (пакетов) с интенсивностями $\lambda_1, \lambda_2, \ldots, \lambda_n$; среднее время предварительной обработки заявки – $1/\mu$.

Необходимо найти резервы времени каждого параллельного простого $s-t$-пути (положительные или отрицательные) по отношению к некоторому выбранному (опорному) $s-t$-пути, а также предельно допустимые значения вариации интервала времени между соседними пакетами в агрегированном $s-t$-пути. В общем случае можно считать, что опорный $s-t$-путь реально не существует, а есть только его идеализированный образец.

Для параметров опорного $s-t$-пути должно задаваться $t_{o\text{ср}} \geq t_{i\text{ср}}$ (при положительном значении резерва), $\sigma_o \leq \sigma_i$, где $t_{o\text{ср}}$ и $t_{i\text{ср}}$ – средние задержки передачи, а σ_o и σ_i – среднеквадратические отклонения времени передачи опорного и i-х простых $s-t$-путей соответственно.

Для синхронизированного протокола i должны выполняться условия:

– время Z – передачи пакетов "из конца – в конец" $s-t$-пути $Z = T + t_з$, где T – существующие задержки в $s-t$-пути, $t_з$ – остающийся запас по задержке передачи;

– $T = t_{\text{ср}} + |l\sigma| < Z$, где l определяет величину интервала, в который случайное время передачи пакета попадает с заданной вероятностью (например, в интервал "трех сигм").

Результатом работы алгоритма должно быть выполнение условия достаточно хорошего приближения значения σ – среднеквадратического отклонения времени передачи через агрегированный $s-t$-путь к значению σ_o опорного простого $s-t$-пути, т.е. $|\sigma - \sigma_0| \leq \varepsilon$, при этом величина ε задается изначально.

Применение GERT-сетей для синхронизации протокола. В GERT-сетях состояниям системы соответствуют узлы графа, а выполняемым в системе операциям – дуги графа. Случайные величины, приписанные дугам GERT-сети, должны обладать свойством аддитивности по дугам любого пути. Основными шагами при использовании GERT-сети являются [6]:

– представление системы в виде стохастической сети G = (**N**,**A**) с **N** GERT-узлами и **A** дугами;

– определение условной вероятности и производящей функции моментов каждой дуги;

– вычисление W– функции каждой дуги;

– нахождение эквивалентной W– функции GERT-сети $W_E(s)$, выражающее связь эквивалентных W– функций петель первого и r-х порядков. Согласно [6] петля первого порядка есть связная последовательность ориентированных дуг, каждый узел которых является общим ровно для двух дуг. Петля порядка r есть множество не связанных между собой петель первого порядка. Эквивалентная W– функция $W_E(s) = p_E M_E(s)$, где p_E – веро-

ятность выполнения стока, $M_E(s)$ – эквивалентная производящая функция моментов выходной величины GERT-сети;

– нахождение математического ожидания μ_{1E} и дисперсии $\sigma^2 = \mu_{2E} - (\mu_{1E})^2$ времени выполнения GERT-сети.

В работах [7-12] рассмотрен численный метод нахождения непрерывной плотности распределения вероятностей выходной величины GERT-сети с использованием эквивалентных упрощающих преобразований структуры GERT-сети. На каждом шаге преобразований исключается один узел GERT-сети, а преобразования продолжаются до получения единственной эквивалентной дуги. Ниже будет использована программная реализация этого метода.

Анализ процессов передачи пакетов на основе моделей GERT существенно облегчается, если GERT-сеть разлагается на сумму параллельных частичных графов [13, 14]. При этом эквивалентная *W*-функция GERT-сети с несколькими простыми *s*-*t*-путями равняется сумме эквивалентных *W*-функций частичных GERT-сетей, в каждой из которых имеется только один простой *s*-*t*-путь.

Определение графа $\widetilde{G}_i$**.** Пусть выполнены следующие условия:

1) P_i есть любая петля первого порядка, не включающая источник *s* и сток *t* и имеющая хотя бы один общий узел с *i*-м простым *s*-*t*-путем;

2) P_j есть любая петля первого порядка, не включающая исток *s* и сток *t* и имеющая хотя бы один общий узел с *j*-м простым *s*-*t*-путем;

3) любые петли P_i и P_j не имеют общих узлов.

Тогда совокупность дуг и узлов *i*-го простого *s*-*t*-пути и множества петель $P_i, i = \overline{1,m}$ есть частичный граф $\widetilde{G}_i, \overline{i = 1,k}$.

Модели такого рода могут быть использованы для анализа и улучшения характеристик типовых протоколов передачи информации.

В GERT-сети могут быть *s*-*t*-пути, содержащие петли 1-го порядка, не имеющие с данным простым *s*-*t*-путем общих узлов. Рассмотрим GERT-сеть на рисунке 1,а. Петля 1-го порядка (3, 3) не имеет общих узлов с простым *s*-*t*-путем (1,2,4), но входит, например, в *s*-*t*-путь (1, 2, 3, 3, 2, 4). В частности петля 1-го порядка (1, 2, 4, 1) вместе с петлей 1-го порядка (3, 3) составляет петлю 2-го порядка. Очевидно, GERT-сеть произвольной структуры не может быть представлена через разложение на графы $\widetilde{G}_i, \overline{i = 1,k}$. Однако можно показать, что GERT-сеть любой структуры может быть эквивалентно преобразована с разложением на графы $\widetilde{G}_i$ с использованием метода эквивалентных упрощающих преобразований [7-12].

Для сети, изображенной на рисунке 1,а топологическое уравнение Мейсона [6] выглядит следующим образом:

$$W_1 W_5 W_A + W_2 W_3 + W_4 - W_1 W_5 W_A W_4 = 1 .$$

Так как эквивалентная *W*-функция GERT-сети $W_E(s) = 1/W_A(s)$, то, разрешая топологическое уравнение относительно $W_E(s)$, получаем

$$W_E = \frac{W_1 W_5 - W_1 W_5 W_4}{1 - W_4 - W_2 W_3} . \qquad (1)$$

Сеть на рисунке 1,а последовательно преобразовывается следующим образом (рисунки 1,б, 1,в, 1,г).

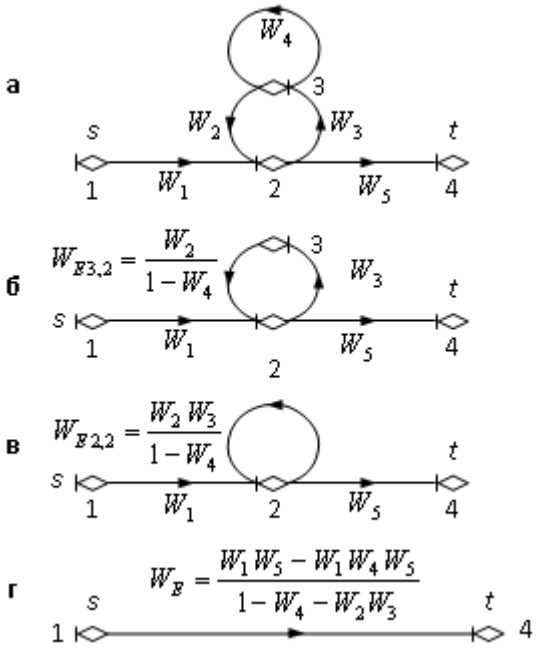


Рисунок 1 – Упрощающие преобразования сети

1. Исключается петля (3,3) с эквивалентным пересчетом *W*-функции дуги (3, 2): $W_{E2} = W_2/(1 - W_4)$.

2. Исключается узел 3 с заменой последовательно соединенных дуг (2, 3) и (3, 2) на эквивалентную: $W_{E\,2,3,2} = W_2 W_3/(1 - W_4)$.

3. Исключается петля в узле 2: $W_{E\,2,4} = W_5/\{1 - [W_2 W_3/(1 - W_4)]\}$.

4. Исключается узел 2 с получением единственной эквивалентной дуги GERT-сети: $W_E = W_1 W_5/\{1 - [W_2 W_3/(1 - W_4)]\}$. Имеем выражение, совпадающее с формулой (1).

Представленные выше преобразования

можно выполнять либо до получения структуры графа $\widetilde{G}_i$ (частный случай приведен на рисунке 1,в), либо до эквивалентного графа данного $s-t$ -пути.

Эквивалентная *W*-функция параллельно соединенных частичных графов $\widetilde{G}_1,\ldots,\widetilde{G}_k$

$$W_E(s)=W_{\widetilde{G}_1}(s)+\ldots+W_{\widetilde{G}_k}(s)=$$
$$=\frac{\prod W_{(s-t)_1}(s)}{1-\sum\prod W_{(\overline{s-t})_1}(s)}+\ldots+\frac{\prod W_{(s-t)_k}(s)}{1-\sum\prod W_{(\overline{s-t})_k}(s)}.$$

Числитель каждого слагаемого равен произведению *W*-функций дуг $W_{(s-t)_i}(s)$, составляющих *i*-й простой *s*-*t*-путь. Знаменатель каждого слагаемого равен разности между единицей и суммой произведений *W*-функций петель всевозможных порядков $W_{(\overline{s-t})_i}(s)$, не включающих в себя как источник, так и сток GERT-сети. В знаменателе петли нечетных порядков имеют знак “минус”, а четных порядков – знак “плюс” [6].

Алгоритм решения задачи. Алгоритм состоит из следующих шагов.

1. Задаем среднее время прохождения “опорного” простого $s-t$ -пути $t_{o\text{ср}}$ и его среднеквадратическое отклонение σ_o.

2. Выбираем достаточно малое значение $|\sigma_i-\sigma_o|\le\delta, i=\overline{1,q}$, где q – число простых $s-t$ -путей, имеющих резервы времени.

3. Производим расчет плотности распределения вероятностей времени прохождения исходной GERT-сети от источника к стоку. Если выполняется соотношение $\varepsilon\le|\sigma-\sigma_0|$, то – переход на п. 11.

4. Выполняем разложение исходной GERT-сети на параллельно соединенные частичные графы $\widetilde{G}_i, \overline{i=1,k}$.

5. Производим расчет плотностей распределения вероятностей времен прохождения частичных графов $\widetilde{G}_i, \overline{i=1,k}$.

6. Изменяем значения резервов $t_{зi}$, так, чтобы $t_{i\text{ср}}=t_{o\text{ср}}$, $i=\overline{1,q},\ i\ne o$.

7. Находим наибольшее значение из среднеквадратических отклонений $\sigma^*=\max\{\sigma_i\}$ всех *q* простых $s-t$ - путей.

8. Последовательно уменьшаем значение σ_i того пути, для которого найдено $\sigma^*=\sigma_i$. Значение σ_i изменяется за счет уменьшения вероятности и времени выполнения возвратных дуг $\widetilde{G}_i^*$. Рассчитываем уточненную плотность распределения вероятностей времени прохождения частичного графа $\widetilde{G}_i^*$. Итерации продолжаются до тех пор, пока не будет выполняться $|\sigma_i-\sigma_o|\le\delta$.

9. Если $q=1$, то переход на п. 10, иначе уменьшаем значение *q* на единицу и переходим на п. 7.

10. Производим расчет плотности распределения вероятностей времени прохождения измененной GERT-сети от источника к стоку. Если выполняется соотношение $\varepsilon\le|\widetilde{\sigma}-\sigma_0|$ (где $\widetilde{\sigma}$ – среднеквадратическое отклонение времени прохождения измененной GERT-сети), то – переход на п. 11, иначе уменьшаем значение σ_o и переходим на п. 7.

11. Конец алгоритма.

Пример синхронизации протокола передачи пакетов. GERT-сеть протокола передачи представлена на рисунке 2,а.

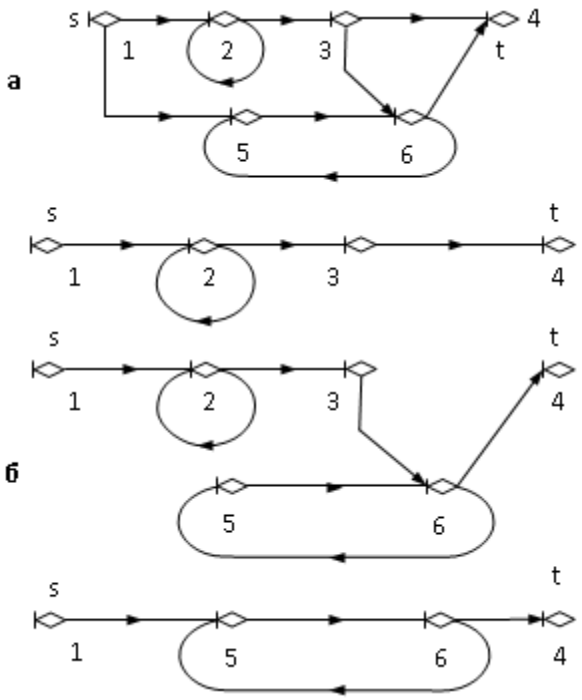


Рисунок 2 – Исходная GERT-сеть (а) и ее разложение на частичные графы (б)

В ней можно выделить 3 параллельно соединенных частичных графа $\widetilde{G}_1$, $\widetilde{G}_2$, $\widetilde{G}_3$, изображенных на рисунке 2,б**.** Характеристики выполняемых операций приведены в таблице 1.

В сети имеется три простых *s*-*t*-пути: (1, 2, 3, 4); (1, 2, 3, 6, 4); (1, 5, 6, 4).

Выполняя разложение GERT-сети на частичные графы $\widetilde{G}_i$, получаем

$$W_E = W_{\widetilde{G}_1} + W_{\widetilde{G}_2} + W_{\widetilde{G}_4} = \frac{W_{1,2}W_{2,3}W_{3,4}}{1-W_{2,2}} +$$

$$+\frac{W_{1,2}W_{2,3}W_{3,6}W_{6,4}}{1-W_{2,2}-W_{5,6}W_{6,5}+W_{2,2}W_{5,6}W_{6,5}} + \frac{W_{1,5}W_{5,6}W_{6,4}}{1-W_{5,6}W_{6,5}}.$$

Каждое слагаемое в этом выражении определяет плотность распределения вероятностей времени прохождения одного из частичных графов $\widetilde{G}_i$.

Таблица 1

Опер.	Вероятн.	Распред.	Параметры
(1,2)	0,5	Норм.	$m = 1$; $\sigma = 1$
(2,2)	0,01	Эксп.	$\lambda = 1$
(2,3)	0,99	Бином.	$p = 0{,}5$; $n = 4$
(3,4)	0,5	Норм.	$m = 25$; $\sigma = 1$
(1,5)	0,5	Норм.	$m = 1$; $\sigma = 1$
(5,6)	1	Норм.	$m = 5$; $\sigma = 1$
(6,5)	0,2	Равном.	$a = 2$; $b = 4$
(3,6)	0,5	Норм.	$m = 15$; $\sigma = 1$
(6,4)	0,8	Пуасс.	$\lambda = 1$

Рисунок 3 – Плотности распределения времени прохождения: а-частичных графов, б-GERT-сети

Плотность распределения времени прохождения пакетов от входа к выходу модели исходного протокола приведена на рисунке 3,б.

Данное распределение имеет три моды и большую вариацию выходной величины, значение 3σ = 27,54 мс (таблица 2). На рисунке 3,а показаны плотности распределения времени прохождения графов $\widetilde{G}_1$, $\widetilde{G}_2$, $\widetilde{G}_3$. Графы $\widetilde{G}_2$ и $\widetilde{G}_3$ (рисунок 3,а) имеют двухмодовые условные распределения выходной величины.

Изменим характеристики протокола путем изменения отдельных параметров его модели. Причиной появления второй моды у графов $\widetilde{G}_2$ и $\widetilde{G}_3$ является неприемлемо большое значение вероятности перехода на дугу (6, 5), равное 0,2. Изменяем ее на значение 0,01, что достигается качественным совершенствованием протокола передачи. В результате этого вторая мода распределения уменьшается до незначительной величины. Заметим, что в результате корректировки вероятности выбора дуги (6, 5) подобным образом изменится и вид кривой распределения графа $\widetilde{G}_3$. После этого модифицированный за счет изменения вероятности выбора дуги (6, 5) граф $\widetilde{G}_2$ принимаем за опорный, относительно которого будем смещать частичные графы $\widetilde{G}_1$ и $\widetilde{G}_3$. Смещение частичного графа $\widetilde{G}_1$ влево обеспечиваем изменением параметра математического ожидания нормального распределения дуги (3, 4) со значения 25 на значение 15 за счет дополнительных материальных затрат. Частичный граф (5, 6) имеет запасы по величине задержки, и поэтому в протокол вводится задержка на 10 единиц путем изменения параметра математического ожидания нормального распределения дуги (5, 6) со значения 5 на значение 15. В результате проведенных изменений (синхронизации протокола передачи информации) мы получаем распределение времени передачи пакета, показанное на рисунке 3,б. Существенно уменьшился интервал «трех сигм» до значения 7,551 (см. таблицу 2 и рисунок 3,б).

Таблица 2

GERT-сеть	$\overline{S}$, мс	3σ, мс	σ, мс	C^2
Исходная GERT-сеть	16,736	27,54	9,18	0,3008
Итоговая GERT-сеть	17,8876	7,551	2,52	0,0198

Наиболее важным для дальнейшего анализа является квадратический коэффициент вариации $C^2 = \left(\sigma/\overline{S}\right)^2$. Известно, что значения наиболее важных характеристик СМО *M*/*G*/1 в стационарном режиме, таких как среднее число заявок в системе, среднее время нахождения заявки в системе, среднее число заявок в очереди, сред-

нее время нахождения заявки в очереди линейно зависят от величины $1+C^2$. Значение квадратического коэффициента вариации времени передачи заявки изменилось с 0,3008 до 0,0198. Ниже рассматривается влияние изменения параметров модели на характеристики системы массового обслуживания *M/M/1 – M/G/1*.

Моделирование характеристик сети передачи данных ПИК. Обобщенная структура сети передачи данных ПИК представлена на рисунке 4.

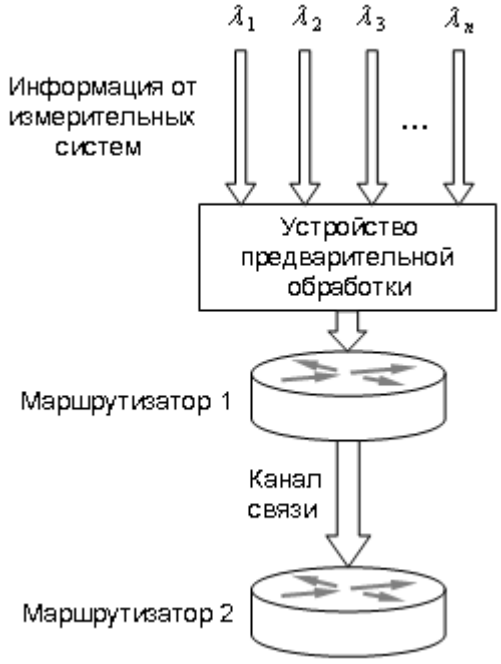


Рисунок 4 – Схема передачи информации ПИК

На первой фазе процесса передачи пакетов выполняется сбор информации от измерительных систем и ее предварительная обработка на специализированном процессоре СП. Она моделируется СМО *M*/*M*/1. Время передачи пакета между маршрутизаторами зависит от многих факторов, определяющих случайные длительности операций протокола передачи и, в общем случае, от вероятностного выбора следующей операции. Распределение времени передачи пакетов по каналу связи непрерывное и произвольное.

Заданы интенсивности пуассоновских (первичных) потоков заявок λ_i, $i=\overline{1,n}$. При слиянии первичных потоков λ_i образуется общий поток с интенсивностью $\Lambda=\sum_{i=1}^{n}\lambda_i$. Тогда фаза первичной обработки информации будет моделироваться СМО *M*/*M*/1 с пуассоновским входным потоком интенсивностью Λ и интенсивностью обслуживания μ. После этого пакеты поступают на вторую фазу – для передачи по каналу связи между маршрутизаторами. В соответствии с *теоремой Берке* задача вычисления распределения промежутков времени между последовательными требованиями, поступающими во вторую фазу, эквивалентна задаче нахождения распределения промежутков времени между последовательными требованиями, уходящими с фазы первичной обработки. Следовательно, на вход второй фазы системы также подается пуассоновский поток с интенсивностью Λ.

Функционирование протокола передачи пакетов между маршрутизаторами отражается некоторой GERT-сетью. Одним из важнейших выходных параметров GERT-сети является математическое ожидание времени ее прохождения от входа к выходу. В терминологии теории массового обслуживания – это среднее время прохождения заявки через обслуживающий прибор (напомним, что рассматривается стационарный режим работы системы). Первый момент выходной случайной величины GERT-сети относительно начала координат, определяемый дифференцированием эквивалентной *W*-функции GERT-сети W_E по формуле $\mu_1=\left[\partial W_E(s)/\partial s\right]\big|_{s=0}=\overline{S}$, и задает интенсивность обслуживания во второй фазе, равной $1/\overline{S}$. Вторая фаза системы может быть представлена СМО *M*/*G*/1 с входным потоком интенсивностью Λ и произвольным законом распределения времени работы обслуживающего прибора со средним значением $\overline{S}$. Итак, рассматриваемая сеть передачи данных ПИК отражается сетью массового обслуживания *M*/*M*/1-*M*/*G*/1.

Наиболее важные *характеристики* СП определяются моделью *M*/*M*/1 при оговоренных условиях относительно входного потока:

– среднее число заявок в СП

$\overline{N}=\rho/(1-\rho)$, где $\rho=\Lambda/\mu$, $\rho<1$;

– среднее время прохождения пакета через СП

$\overline{T}=\rho/\left[\Lambda(1-\rho)\right]$;

– среднее число заявок в очереди СП

$\overline{Q}=\rho^2/(1-\rho)$;

– среднее время нахождений в очереди СП

$\overline{W}=\rho/(\mu-\Lambda)$.

Рассмотрим характеристики *канала связи между маршрутизаторами* и возможности их улучшения за счет изменения параметров операций протокола. Сразу отметим, что условием стационарного режима является неравенство $\Lambda<\mu_1$, поэтому величина нагрузки $\rho_2=\Lambda/\mu_1$

для второй фазы не совпадает со значением ρ для первой фазы.

Нормированная дисперсия времени передачи пакетов (среднеквадратический коэффициент вариации) через канал $C^2 = \sigma^2 / \overline{S}^2$.

Тогда на основании теоремы Литтла и формул Полячека-Хинчина для СМО *M*/*G*/1 [15] получаем следующие результаты:

среднее число пакетов в канале

$$\overline{N_к} = \rho_2 + \frac{\rho_2^2\left(1+\sigma^2/\overline{S}^2\right)}{2\left(1-\rho_2\right)},$$

среднее число пакетов, находящихся в очереди входного буферам канала

$$\overline{Q}_к = \frac{\rho_2^2\left(1+\sigma^2/\overline{S}^2\right)}{2\left(1-\rho_2\right)},$$

среднее время передачи пакета через канал

$$\overline{T}_к = \overline{S} + \frac{\Lambda\left(1+\sigma^2/\overline{S}^2\right)}{2\left(1-\rho_2\right)}. \qquad (2)$$

Общее среднее время нахождения пакета в системе состоит, очевидно, из среднего времени, проведенного на обслуживании, и среднего времени пребывания пакета в очереди входного буфера канала. Так как первое слагаемое правой части равенства (2) есть, очевидно, среднее время обслуживания, то второе слагаемое должно представлять собой среднее время ожидания. Следовательно, *среднее время нахождения пакета в очереди*

$$\overline{W}_к = \frac{\Lambda\left(1+\sigma^2/\overline{S}^2\right)}{2\left(1-\rho_2\right)}. \qquad (3)$$

Сравним среднее время $\overline{T}_к$ пребывания пакета в канале с величиной $\overline{S}$ среднего времени обслуживания пакета. Отношение $\overline{T}_к / \overline{S}$ – отношение времени, проведенное в канале, к времени, приходящемуся в среднем на обслуживание, характеризует *коэффициент неудобства* системы, обусловленный наличием в ней других пакетов. Нормируя равенства (2) и (3), получаем формулы, в которых время выражено в единицах среднего времени обслуживания:

$$\frac{\overline{T}_к}{\overline{S}} = 1 + \frac{\Lambda\left(1+\sigma^2/\overline{S}^2\right)}{2\left(1-\rho_2\right)},$$

$$\frac{\overline{W}_к}{\overline{S}} = \frac{\lambda\left(1+\sigma^2/\overline{S}^2\right)}{2\overline{S}\left(1-\rho_2\right)}.$$

Общее среднее время передачи пакета через рассматриваемую систему

$$\overline{T}_\Sigma = \frac{1}{\mu} + \overline{T}_к = \frac{1}{\mu} + \overline{S} + \frac{\Lambda\left(1+\sigma^2/\overline{S}^2\right)}{2\left(1-\rho_2\right)}.$$

Пример улучшения характеристик канала. Улучшение характеристик протокола передачи ПИК на основе изменения в GERT-сети параметров случайных величин отдельных дуг, как это было описано выше и проиллюстрировано на рисунке 3, подтверждается данными, приведенными в таблице 3 (исходная система) и в таблице 4 (система с улучшенными параметрами).

Таблица 3

ρ_2	$\overline{N_к}$	$\overline{Q}_к$	$\overline{T}_к$	$\overline{W}_к$
0,5	0,825	0,325	16,775	0,039
0,6	1,185	0,585	16,794	0,058
0,7	1,762	1,062	16,826	0,09
0,8	2,881	2,081	16,891	0,155
0,9	6,168	5,268	17,087	0,351
0,95	12,69	11,740	17,484	0,748
0,99	64,736	63,746	20,908	4,172
0,999	650,099	649,1	61,844	45,108

Таблица 4

ρ_2	$\overline{N_к}$	$\overline{Q}_к$	$\overline{T}_к$	$\overline{W}_к$
0,5	0,755	0,255	7,618	0,067
0,6	1,059	0,459	7,652	0,101
0,7	1,533	0,833	7,709	0,158
0,8	2,432	1,632	7,821	0,27
0,9	5,03	4,13	8,159	0,608
0,95	10,154	9,204	8,832	1,281
0,99	50,965	49,975	14,193	6,642
0,999	509,88	508,881	74,829	67,278

При рекомендуемых уровнях загрузки $\rho_2 = 0{,}7 - 0{,}8$ среднее время, проводимое пакетами в системе и очереди, сократилось более чем вдвое. Затраты памяти во входном буфере маршрутизатора 1 уменьшились на 27,5 %.

Заключение. Предложен метод синхронизации протоколов передачи пакетов в компьютерной сети на основе использования моделей GERT. Он заключается в выявлении скрытого параллелизма путей передачи пакетов в протоколе, разложении GERT-сети протокола на частичные графы, сведении многомодового распределения к одномодовому распределению за счет изменения отдельных характеристик протокола передачи пакетов. Одним из наиболее важных применений метода является возможность уменьшения среднеквадратического отклонения времени передачи пакетов от его среднего значения. На практике во многих случаях удается уменьшить «джиттер» пакетов в 3 – 4 раза, что существенно повышает показатели качества сетевого канала при передаче синхронной информации. Несколько меньший в абсолютном выра-

жении, но достаточно важный результат достигается и за счет уменьшения среднего числа пакетов в канале ПИК, среднего времени передачи пакетов через канал, а также зависящих от них величин – среднего числа пакетов в буферной памяти и среднего времени ожидания обслуживания пакета каналом (ожидания начала передачи).

Работа поддержана Российским Фондом фундаментальных исследований, грант № 14-07-00106-а.

Библиографический список

1. *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. – 2010. – № 33. – С. 49-55.

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

3. *Корячко В.П., Перепелкин Д.А., Иванчикова М.А.* Алгоритм адаптивной маршрутизации в корпоративных сетях нескольких провайдеров связи // Вестник Рязанского государственного радиотехнического университета. – 2013. – № 2(44). – С. 52-56.

4. *Рудаков В.Е., Скворцов С.В.* Построение базового множества независимых путей потокового графа для тестирования программных модулей // Системы управления и информационные технологии. – 2012. – Т. 50. – № 4. – С. 67-70.

5. *Першин А.С., Скворцов С.В.* Распределение регистровой памяти в системах параллельной обработки данных // Системы управления и информационные технологии. – 2007. – № 1 (27). – С. 65-70.

6. *Pritsker A. A. B.* GERT: Graphical evaluation and review technique. Memorandum RM-4973-NASA, April 1966. P. 1-36.

7. *Шибанов А.П.* Нахождение плотности распределения времени исполнения GERT-сети на основе эквивалентных упрощающих преобразований // Автоматика и телемеханика. – 2003. – № 2. – С. 117-126.

8. *Shibanov A.P.* Finding the distribution density of the time taken to fulfill the GERT network on the basis of equivalent simplifying transformation. Automation and Remote Control. Plenum Press New York, NY, USA. February 2003 . Volume 64. Issue 2. P. 279-287.

9. *Shibanov A.P.* A software implementation technique for simulation of Ethernet local area networks // Programming and Computer Software. 2002. Т. 28. № 6. С. 349-3.

10. *Шибанов А.П.* Разработка программного обеспечения для моделирования локальных сетей Ethernet // Программирование. – 2002. – № 6. – С. 62-71.

11. *Шибанов А.П.* Метод эквивалентных упрощающих преобразований GERT-сетей и его приложения // Вестник Рязанского государственного радиотехнического университета. – 2012. – № 39-2. – С. 76-83.

12. *Шибанов А.П.* Нахождение дискретного распределения выходной величины GERT-сети // Вестник Рязанского государственного радиотехнического университета. – 2002. – № 10. – С. 43-48.

13. *Корячко В.П., Скворцов С.В., Таганов А.И., Шибанов А.П.* Эволюция автоматизированного проектирования электронно-вычислительных средств // Радиотехника. – 2012. – № 3. – С. 97-102.

14. *Козлов М.А., Скворцов С.В.* Алгоритмы параллельной сортировки данных и их реализация на языке Clojure // Вестник Рязанского государственного радиотехнического университета. – 2013. – № 4-1 (46). – С. 92-96.

15. *Хинчин А.Я.* Математическая теория стационарной очереди // Математический сборник. М. 1932. Т. 39. С. 73-84.

УДК 681.3

В.А. Антипов, О.В. Антипов, А.Н. Пылькин

ИНТЕГРАЦИЯ РАСПРЕДЕЛЁННЫХ ПРОГРАММНЫХ ПРИЛОЖЕНИЙ НА ОСНОВЕ МАРШРУТИЗАЦИИ ПО СОДЕРЖИМОМУ СООБЩЕНИЙ

На основе формальной спецификации систем Публикация/Подписка предлагается подход к разработке маршрутизатора сообщений на основе содержимого сообщений.

Ключевые слова: *распределённые приложения, сообщения, маршрутизатор.*

Введение. Корпоративные приложения не могут существовать обособленно одно от другого. Большинство интеграционных решений объединяет разнородные приложения: унаследован-

ные, коммерческие и созданные на заказ. При этом интеграционное решение должно учитывать все различия, существующие между объединяемыми системами (язык, платформу, формат данных). Кроме того, решение должно иметь возможность адаптации к изменению объединяемых им приложений. Преобразования в одной системе влекут за собой непредсказуемые последствия для других систем. При интеграции приложений важно уменьшить их взаимозависимость за счет так называемого *слабого связывания*, при этом можно воспользоваться подходом, именуемым как *обмен сообщениями* [1 – 6].

Обмен сообщениями – это технология высокоскоростного асинхронного взаимодействия между программными приложениями с гарантией доставки информации. Приложения взаимодействуют между собой, обмениваясь пакетами данных, называемыми *сообщениями*, передаваемыми через *канал. Канал* или *очередь* – это логический маршрут, объединяющий программы и используемый для транспортировки сообщений. Канал напоминает массив сообщений, доступный для одновременного использования многими приложениями. Функциональная часть обмена сообщениями обеспечивается отдельной программной системой, называемой *системой обмена сообщениями* или *связующим ПО*, *ориентированным на обмен сообщениями* (*Message-oriented middleware – MOM*).

В работе рассматривается один из важнейших вопросов, связанных с разработкой МОМ, – маршрутизация сообщений. Одна из наиболее распространенных форм поддержки маршрутизации сообщений – *маршрутизатор на основе содержимого* (*Content-Based Router – CBR*), который перенаправляет сообщение в исходящий канал в зависимости от используемого критерия [1, 7].

Целью работы являются формальное рассмотрение механизмов маршрутизации на основе содержимого и разработка базовой модели системы, построенной на асинхронном принципе передачи сообщений. При этом требования к конфигурированию маршрутов должны соответствовать спецификации систем Публикация/Подписка, представленных в [7].

Формализация маршрутизации позволит получить полное представление о поведении распределённых систем, построенных на базе парадигмы Публикация/Подписка, и даст универсальный критерий оценки корректности конкретного экземпляра системы.

Модель системы Публикация/Подписка. Рассмотрим распределенную систему, состоящую из множества процессов, которые взаимодействуют друг с другом асинхронным способом передачи сообщений с использованием надежных каналов связи. Предполагается, что сообщения доставляются от отправителя в порядке FIFO и что они не дублируются, не потеряны, не испорчены и не ошибочно отправлены.

Модель распределенной системы на базе парадигмы Публикация/Подписка реализуется в виде набора взаимодействующих процессов, называемых *брокерами*, которые соединены между собой в виде ациклической топологии. Каждый брокер B выступает в качестве местной *точки доступа* к системе и управляет эксклюзивным набором локальных клиентов L_B, который является подмножеством множества всех клиентов C. Кроме того, B взаимодействует с соседними брокерами N_B, к которым он непосредственно подключён (рисунок 1). Таким образом, брокеры уведомлений и управляющих сообщений реализуют функциональность распределенного *сервиса уведомлений*. Стратегия передачи сообщений определяется маршрутизационным алгоритмом на основе содержимого, который определяет, как уведомления направляются через брокеров сети и как осуществляется фильтрация уведомлений при посредничестве брокеров.

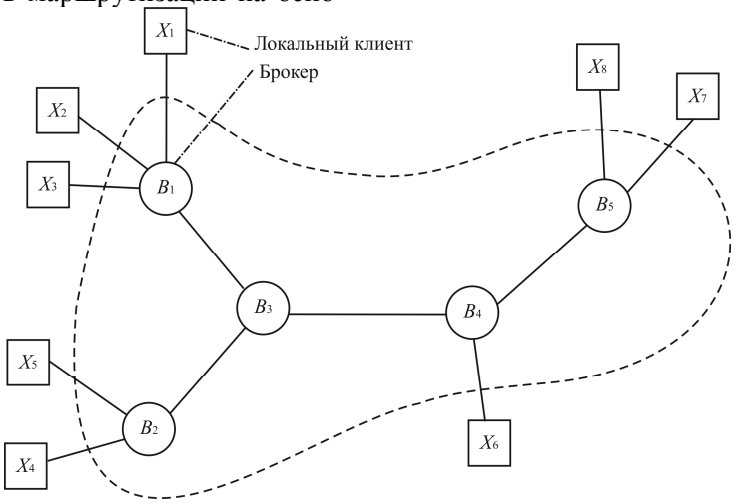


Рисунок 1 – Распределенная система Публикация/Подписка

Формально топология взаимосвязей брокеров представляется как связный неориентированный ациклический граф $G = (V, E)$ с множеством вершин $V = (B_1, ..., B_n)$, соответствующих брокерам, и множеством рёбер $E \subseteq \{(B_i, B_j), 1 \le i \le j \le n\}$, представляющих двунаправленные связи между ними. Определим некую функцию $e(B_i, B_j)$, возвращающую (B_i, B_j), если $i < j$ и (B_j, B_i) в противном случае.

Конфигурация маршрутизации. ***Переадресация уведомлений на основе таблиц маршрутизации.*** Каждый брокер B функционирует на основе своей таблицы маршрутизации T_B, которая содержит набор маршрутизационных записей. Каждая запись есть пара (F, U), состоящая из фильтра F и точки назначения $U \in N_B \cup L_B$. Граф G вместе со всеми таблицами маршрутизации называется *текущей конфигурацией маршрутизации* системы Публикация/Подписка. Конфигурация маршрутизации одного брокера B состоит из двух непересекающихся частей: *удаленной конфигурации маршрутизации*, которая содержит все записи маршрутизации, пунктом назначения которых является соседний брокер B, и *локальной конфигурации маршрутизации*, состоящей из всех записей маршрутизации, пунктом назначения которых является локальный клиент брокера B.

Текущая конфигурация маршрутизации инициирует *множество уведомлений* о том, что именно брокер направляет к месту назначения (соседним брокерам и локальным клиентам, подключённым к нему).

Формально набор уведомлений n брокера для текущей маршрутизации подмножества W из $N_B \cup L_B$ можно представить в виде

$$\nu_B(W) = \{n \mid \exists (F,U) \in T_B . n \in N(F) \wedge U \in W\}, \quad (1)$$

где $\nu_B(W)$ – есть множество уведомлений, которые брокер B направляет к любому члену W и которые принадлежат $N(F)$ – уведомлениям, соответствующим F. Согласно этому определению всегда можно выделить два подмножества α, β из $N_B \cup L_B$, для которых $\alpha \subseteq \beta \Rightarrow \nu_B(\alpha) \subseteq \nu_B(\beta)$ и $\nu_B(\alpha \cup \beta) = \nu_B(\alpha) \cup \nu_B(\beta)$. Точки назначения, локальные клиенты и соседние брокеры, которым брокер B перенаправляет уведомления n, задаются как $F_B(n), F_B^L(n)$ и $F_B^N(n)$:

$$F_B(n) = \{D \mid D \in N_B \cup L_B \wedge n \in \nu_B(\{D\})\}, \quad (2)$$

$$F_B^L(n) = F_B(n) \cap L_B, \quad (3)$$

$$F_B^N(n) = F_B(n) \cap N_B, \quad (4)$$

С учетом данных утверждений, легко определить алгоритм перенаправления уведомлений брокером, основанный на его таблице маршрутизации (рисунок 2).

```
program StaticNotificationForwarding()
begin
  initialize T_B
  loop
    m ← wait for and return next message
    forwardNotification(m)
  end
end

procedure forwardNotification(Message m)
begin
  if m is "forward(n)" message from neighbor U then
    send "notify(n)" to all local clients in F_B^L(n)
    send "forward(n)" to all neighbors in F_B^N(n) \ {U}
  fi
  if m is "pub(n)" from client X then
    send "notify(n)" to all local clients in F_B^L(n)
    send "forward(n)" to all neighbors in F_B^N(n)
  fi
end
```

Рисунок 2 – Алгоритм статического перенаправления уведомлений

Согласно этому алгоритму входящие сообщения обрабатываются последовательно в порядке FIFO независимо от отправителя. Брокеры распространяют уведомления друг другу с помощью отправки и приёма сообщений *forward*(n). Брокер получает *pub*(n) сообщения от клиентов и посылает *notify*(n) сообщения своим клиентам [4].

Алгоритм работает следующим образом.

Если брокер получает сообщение *pub*(n) от одного из своих локальных клиентов, то он посылает *notify*(n) сообщение всем своим локальным клиентам, которые есть в $F_B^L(n)$, и *forward*(n) сообщение всем своим соседним брокерам, которые есть в $F_B^N(n)$.

Если брокер получает *forward*(n) сообщение от одного из своих соседей U, то он посылает *notify*(n) сообщение всем своим локальным клиентам, которые есть в $F_B^L(n)$, и *forward*(n) сообщение всем своим соседям, которые есть в $F_B^N(n) \setminus \{U\}$. Например, в ситуации, изображенной на рисунке 3, B_1 направляет уведомление, полученное от X_1, своим локальным клиентам X_2 и соседям B_2.

С учетом понятий конфигурации маршрутизации и перенаправления уведомлений, основанных на таблицах маршрутизации, необходимо определить условия, которые должны выполняться, чтобы можно было гарантировать, что система публикации/подписки будет вести себя корректно.

Статическая система публикации/ подписки*: *допустимая маршрутная конфигурация. В статических системах публикации/подписки набор участвующих брокеров,

связи между ними и подписки их локальных клиентов всё время их работы постоянны. В таких системах маршрутная конфигурация должна гарантировать, что уведомление, опубликованное для произвольного клиента, будет доставлено любому клиенту с соответствующей подпиской. Допустимая маршрутная конфигурация, представленная ниже, удовлетворяет этим условиям. Неформально брокер должен различать две группы уведомлений: множество уведомлений $v_{B_Y}(\{Y\})$, которые направляются локальным клиентам, и множество уведомлений $v_{B_i}(\{B_j\})$, которые направляются соседним брокерам.

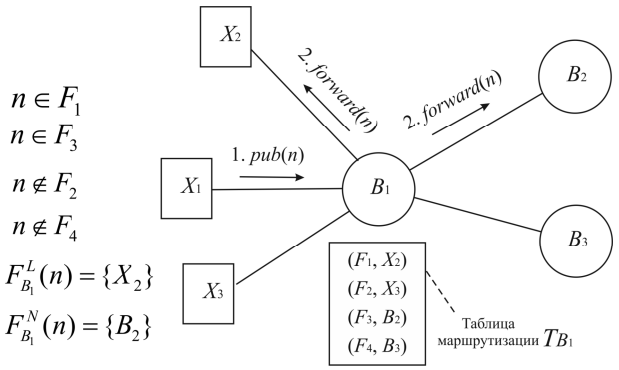


Рисунок 3 – Диаграмма перенаправления уведомлений

Рассмотрим $v_{B_Y}(\{Y\})$. Система является корректной [7], если удовлетворяет условию, что брокер доставляет точно те уведомления одному из локальных клиентов, которые ему интересны. Это означает, что $v_{B_Y}(\{Y\})$ всегда должно быть эквивалентно $\cup_{F \in S_Y} N(F)$, где S_Y – множество активных подписок Y (т. е. все фильтры, которые Y подписал и пока не аннулировал подписку). Если $v_{B_Y}(\{Y\})$ будет содержать больше уведомлений, то будет нарушено требование *безопасности* системы. Если $v_{B_Y}(\{Y\})$ будет содержать меньше уведомлений, эти уведомления никогда не будут доставлены, и в этом случае нарушается требование *живучести* системы.

Рассмотрим $v_{B_i}(\{B_j\})$, которое, по меньшей мере, должно содержать все уведомления, интересные локальным клиентам и соседним брокерам. Но, так как топология связей брокеров ациклична, то этот набор должен дополнительно содержать и такие уведомления, в которых заинтересованы клиенты, связанные с соседними брокерами. Если это не так, то некоторые из этих клиентов не получат всех интересующих их уведомлений. В отличие от $v_{B_Y}(\{Y\})$ множество $v_{B_i}(\{B_j\})$ может содержать больше уведомлений, чем это необходимо, потому что финальная фильтрация достигается через фильтрацию $v_{B_Y}(\{Y\})$. Конечно, этот избыток должен быть как можно меньше, чтобы не была превышена пропускная способность сети. Эти требования необходимы и достаточны для того, чтобы статическая маршрутизация уведомлений поддерживала корректное функционирование системы.

Определим I_B как множество всех уведомлений, которые интересны любому из локальных клиентов брокера B:

$$I_B = \cup_{X \in L_B} \cup_{F \in S_X} N(F). \quad (5)$$

Заметим, что I_B меняется каждый раз, когда клиент подписывается на определённое сообщение или аннулирует его. Рассмотрим некоего брокера B_i и одного из его соседних брокеров B_j. Если между B_i и B_j нет связи, то G разделяется на два несвязанных подграфа: один содержит B_i, а другой B_j. Допустим, что $G_{B_i, B_j} = (V_{B_i, B_j}, E_{B_i, B_j})$ есть подграф, содержащий все вершины, связанные с B_i. Множество всех уведомлений, которые интересны хотя бы одному потребителю любого брокера из V_{B_i, B_j}, обозначим как

$$\eta_{B_i, B_j} = \cup_{B \in V_{B_i, B_j}} I_B. \quad (6)$$

Допустимая маршрутная конфигурация может быть определена следующим образом.

Определение 1. Допустимая маршрутная конфигурация. Маршрутная конфигурация допустима, если

$$e(B_i, B_j) \in E \Rightarrow v_{B_i}(\{B_j\}) \supseteq \eta_{B_j, B_i}$$

(удалённая допустимость),

$$Y \in L_{B_Y} \Rightarrow v_{B_Y}(\{Y\}) = \cup_{F \in S_Y} N(F)$$

(локальная допустимость).

Удалённая допустимость формализует требование, предъявляемое к $v_{B_i}(\{B_j\})$, тогда удалённая маршрутная конфигурация брокера B_i гарантирует, что те уведомления, которые будут посланы через соединение B_i к B_j, представляют интерес хотя бы одному локальному клиенту любого брокера в $V_{Bj, Bi}$ (рисунок 4).

Локальная допустимость формализует требование, предъявляемое к $v_{B_Y}(\{Y\})$, тогда локальная маршрутная конфигурация гарантирует доставку локальному клиенту Y только тех уведомлений, на которые у него есть подписка в S_Y (рисунок 5).

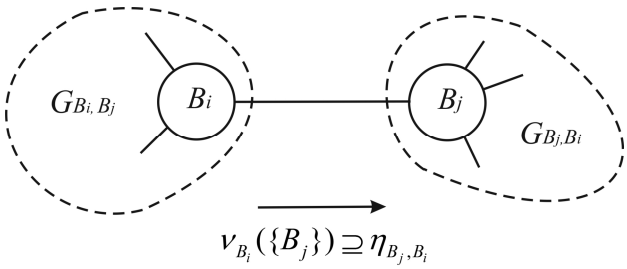


Рисунок 4 – Диаграмма удалённой допустимости маршрутной конфигурации

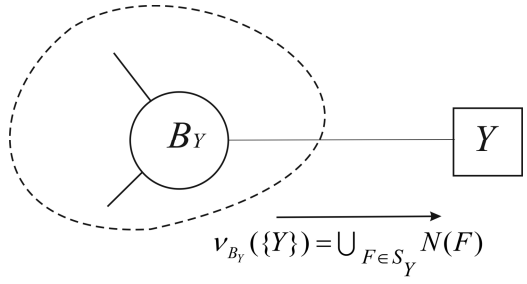


Рисунок 5 – Диаграмма локальной допустимости маршрутной конфигурации

Допустимая маршрутная конфигурация называется *совершенной*, если в удалённой допустимости подмножество отношений заменяется равенством. Совершенная маршрутная конфигурация гарантирует, что ни одно из уведомлений не будет доставлено напрасно. Это, с одной стороны, минимизирует использование пропускной способности сети, а с другой – может привести к уменьшению таблиц маршрутизации.

Докажем, что если допустимая маршрутная конфигурация используется в сочетании с алгоритмом, приведенным на рисунке 2, то система публикации/подписки будет удовлетворять требованиям безопасности и живучести, сформулированным в [7]. Представим доказательство в виде последовательности шагов разных уровней, например <1>2 есть второй шаг доказательства на уровне 1. Каждый шаг содержит утверждение, которое может быть доказано на нижних уровнях с помощью дополнительных шагов. Представление доказательства в таком структурированном виде дает возможность читать только его верхние уровни, опускаясь на подуровни, если это необходимо.

Докажем, что удовлетворяется требование безопасности системы. Это может быть сделано с помощью проверки трёх его составляющих [7], что и приведено в леммах 1 – 4. Требование живучести системы проверено в лемме 5. Вместе эти результаты сведены в теорему о корректности системы.

Лемма 1. $\Box[Notify(Y, n) \Rightarrow \circ\Box \neg Notify(Y, n)]$.

Докажем утверждение, что всегда, если клиент будет уведомлён о n, то он больше никогда не будет уведомлён о n снова. Действительно, каждое уведомление n может быть опубликовано только один раз, так как уведомления уникальны. Из алгоритма, представленного на рисунке 2 (пересылка уведомлений), ацикличности графа G и в связи с предположением о надёжности канала следует также, что n может быть доставлено клиенту только один раз.

Лемма 2. Если маршрутная конфигурация допустима, то

$\Box[Notify(Y, n) \Rightarrow [\exists F \in S_Y.\ n \in N(F)]]$.

Докажем, что допустимая маршрутная конфигурация подразумевает, что только соответствующие уведомления доставляются клиенту. В допустимой маршрутной конфигурации в силу свойства локальной допустимости, если $Y \in L_{B_Y}$, то $\nu_{B_Y}(\{Y\}) = \cup_{F \in S_Y} N(F)$. Это утверждение вместе с предположением о надёжности канала и алгоритмом на рисунке 2 гарантирует доставку только соответствующих уведомлений. Приведем подробное доказательство.

Предположим, что:

1) маршрутная конфигурация допустима;

2) $Notify(Y, n)$.

Докажем, что $\exists\ F \in S_Y.\ n \in N(F)$. Допустим, что B_Y – это брокер, обслуживающий Y. Тогда

<1>1. B_Y посылает сообщение *notify*(n) к Y. Это справедливо в соответствии с предположением 2 и алгоритмом (рисунок 2, строки 13 – 17).

<1>2. $Y \in F^L_{B_Y}(n)$. Это следует из шага <1>1 и алгоритма (рисунок 2, строки 13 – 17).

<1>3. $n \in \nu_{B_Y}(\{Y\})$. Это следует из шага <1>2 и определения $F^L_{B_Y}$.

<1>4. $\nu_{B_Y}(\{Y\}) = \cup_{F \in S_Y} N(F)$. Это следует из определения допустимой маршрутной конфигурации (локальная допустимость) и предположения 1.

<1>5. Исходное утверждение справедливо, так как из <1>3 и <1>4 следует существование фильтра $F \in S_Y$, который соответствует n.

Лемма 3. $\Box[Notify(Y, n) \Rightarrow [\exists X.\ n \in P_X]]$.

Докажем, что если клиент уведомлён об n, то n было опубликовано ранее одним из других клиентов. Это следует из алгоритма и предположения о надёжности канала. Доказательство проводится в обратном порядке: от доставки n к её публикации в соответствии с топологией сети.

Предположим, что $Notify(Y, n)$. Докажем, что $\exists X.\ n \in P_X$.

<1>1. B_Y посылает сообщение *notify*(n) к Y. Это справедливо в соответствии с приведенным предположением и моделью системы.

<1>2. B_Y получает или сообщение *pub*(*n*), или *forward*(*n*). Это следует из шага <1>1 и алгоритма (рисунок 2, строки 12 – 16).

<1>3. Рассмотрим случай, когда B_Y получает сообщение *pub*(*n*) от клиента $X \in L_{B_Y}$.

<2>1. $n \in P_X$,что справедливо в соответствии с предположением и моделью системы.

<2>2. <1>3 следует из шага <2>1.

<1>4. Рассмотрим случай, когда B_Y получает сообщение *forward*(*n*) от соседа.

<2>1. Сообщение должно исходить от какого-либо брокера *B*. Это следует из надёжности каналов, алгоритма (строки 14 – 18) и топологии сети.

<2>2. Брокер *B* получает *pub*(*n*) сообщение от клиента $X \in L_B$. Это следует из шага <2>1, надёжности каналов и алгоритма (рисунок 2, строка 16).

<2>3. <1>4 справедливо, так как из шага <2>2 и модели системы следует, что $n \in P_X$.

<1>5. Исходное утверждение справедливо, что следует из <1>2 в соответствии с <1>3 и <1>4, охватывающими все случаи.

Лемма 4. Если маршрутная конфигурация допустима, то статическая пересылка уведомлений удовлетворяет требованию безопасности.

Доказательство. Леммы 1, 2 и 3 доказывают, что все три требования безопасности удовлетворены, и, следовательно, общее требование безопасности удовлетворяется в полном объеме.

Лемма 5. Если маршрутная конфигурация допустима, то статическая пересылка уведомлений удовлетворяет требованию живучести.

Чтобы доказать, что подписки являются статическими, достаточно показать, что из утверждения $[Pub(X, n) \wedge \exists F \in S_Y \ . \ n \in N(F)]$ следует $\Diamond Notify(Y, n)$. Это доказывается индукцией по топологии брокерской сети. Во-первых, локальная доставка доказана, так как B_Y посылает сообщение *notify*(*n*) к *Y*, если он получает сообщение *forward*(*n*) или *pub*(*n*), и *Y* имеет подписку, совпадающую с *n*. После этого начальный шаг индукции доказан. Локальная доставка непосредственно следует из локальной допустимости маршрутной конфигурации. С другой стороны, удалённая допустимость подразумевает, что *n* перенаправляется к каждому брокеру *B*, обслуживающему локальных клиентов с соответствующей подпиской, поскольку при удалённой допустимости уведомления пересылаются по всем ссылкам, представляющим подсети брокеров с клиентами, имеющими соответствующие подписки.

Предположим, что:

1) $Pub(X, n)$;

2) $F \in S_Y$. $n \in N(F)$;

3) маршрутная конфигурация допустима.

Докажем, что $\Diamond Notify(Y, n)$.

Пусть B_X и B_Y – брокеры, обслуживающие соответственно *X* и *Y*. Тогда

<1>1. Если B_Y получает *forward*(*n*) или *pub*(*n*) сообщение, то посылает *notify*(*n*) к *Y*.

Докажем, что если B_Y получает *forward*(*n*) или *pub*(*n*) сообщение, то он посылает *notify*(*n*) сообщение к *Y*.

<2>1. B_Y посылает *notify*(n) сообщение к *Y*, если $Y \in F_B^L(n)$. Это справедливо в соответствии с приведенными предположениями и алгоритмом (рисунок 2, строки 13 – 17).

<2>2. $F \in S_Y$. $n \in N(F)$. Это следует из предположения 2.

<2>3. $v_{B_Y}(\{Y\}) = \cup_{G \in S_Y} N(G)$. Это следует из предположения 3.

<2>4. $n \in v_{B_Y}(\{Y\})$. Это следует из <2>2 и <2>3.

<2>5. $Y \in F_B^L(n)$. Это следует из шага <2>4 и определения $F_B^L(n)$.

<2>6. Доказываемое утверждение следует из <2>1 и <2>5.

<1>2. *n* доставляется к *Y*, если $B_X = B_Y$.

<2>1. B_Y получает *pub*(*n*) сообщение от *X*. Это справедливо в соответствии с предположением 1, фактом, что $B_X = B_Y$, и алгоритмом на рисунке 2.

<2>2. Доказываемое утверждение следует из <1>1 и <2>1.

<1>3. Пусть $B_k \neq B_Y$ есть произвольный брокер на пути от B_X к B_Y , а B_l есть следующий брокер на пути от B_k к B_Y. Это справедливо, поскольку пути определены и уникальны, так как граф *G* имеет связанную ациклическую топологию.

<1>4. B_k направляет *n* к B_l.

<2>1. $B_Y \in V_{B_l, B_k}$. Это справедливо в соответствии с фактом, что B_l ближе к B_Y, чем B_k, и определением V_{B_l, B_k}.

<2>2. $\eta_{B_l, B_k} \supseteq I_{B_Y}$. Это следует из определения η_{B_l, B_k} в формуле (6) и шага <2>1.

<2>3. $n \in I_{B_Y}$. Это следует из предположения 2 и определения I_{B_Y}.

<2>4. $v_{B_k}(\{B_l\}) \supseteq \eta_{B_l, B_k}$. Это следует из предположения 3.

<2>5. Доказываемое утверждение следует из <2>2, <2>3 и <2>4.

<1>5. Исходное утверждение справедливо, что следует из шага <1>2 (начальный случай) и <1>4 (шаг индукции).

Теорема 1. Если допустимая маршрутная конфигурация используется в сочетании с алгоритмом, приведенным на рисунке 2, тогда система публикации/подписки удовлетворяет требованиям безопасности и живучести. Доказательство теоремы следует из лемм 4 и 5.

Можно доказать, что оба свойства, которым должна удовлетворять допустимая маршрутная конфигурация, необходимы для удовлетворения безопасности и живучести статической системы публикации/подписки.

Лемма 6. Удалённая допустимость маршрутной конфигурации необходима статической системе публикации/подписки, использующей алгоритм на рисунке 2, для удовлетворения условия живучести.

Пусть свойство 1 (удалённая допустимость) допустимой маршрутной конфигурации нарушается, покажем, что это ведет к нарушению условия живучести [7]. Нарушение удалённой допустимости подразумевает, что есть уведомления, которые не были посланы через какое-то соединение, хотя есть некоторые клиенты в определенной подсети, которые имеют на них подписку. Следовательно, если такое уведомление будет опубликовано, оно не будет доставлено заинтересованным клиентам. Это означает нарушение условия живучести.

Предположим, что $\exists B_i, B_j$ с $e(B_i, B_j) \in E$, для которых $v_{Bi}(\{B_j\}) \supseteq \eta_{Bj,\,Bi}$, не выполняется. Докажем, что условие живучести [7] нарушается.

<1>1. $\exists n$ где $n \in \eta_{Bj,\,Bi} \setminus v_{Bi}(\{B_j\})$. Это следует из предположения.

<1>2. $\exists B \in V_{Bj,\,Bi}.\, n \in I_B$. Это следует из определения $\eta_{Bj,\,Bi}$ (формула 6).

<1>3. $\exists Y \in L_B.\, \exists F \in S_Y.\, n \in N(F)$. Это следует из определения I_B (формула 5).

<1>4. Предположим, n опубликовано локальным клиентом B_i. Докажем, что n не доставлено Y.

<2>1. B_i не направил n к B_j, что следует из шага <1>1.

<2>2. n не перенаправлено B. Это следует из <1>2, <2>1, факта ацикличности топологии и индукции.

<2>3. Доказываемое утверждение следует из <2>2 и <1>3, следовательно, n не доставлено Y.

<1>5. Исходное утверждение вытекает из <1>3 и <1>4, следовательно, условие живучести системы нарушено.

Лемма 7. Локальная допустимость маршрутной конфигурации необходима для статической системы публикации/подписки, использующей алгоритм на рисунке 2, для удовлетворения условиям безопасности и живучести [7].

Предположим, что свойство 2 (локальная допустимость) допустимой маршрутной конфигурации нарушается и покажем, что это приводит к невыполнению условия безопасности или живучести [7]. Нарушение локальной допустимости приводит к тому, что некоторые клиенты не получают некоторые интересующие их уведомления (нарушение живучести), или некоторые клиенты получают не интересующие их уведомления (нарушение безопасности).

Пусть $\exists Y.\, v_{B_Y}(\{Y\}) \neq \cup_{F \in S_Y} N(F)$, где B_Y брокер, обслуживающий Y.

Докажем, что безопасность и живучесть системы нарушаются.

<1>1. Рассмотрим случай $\exists n \in \cup_{F \in S_Y} N(F) \setminus v_{B_Y}(\{Y\})$. Предположим, что есть локальный клиент B_Y, публикующий n.

<2>1. n не доставляется к Y. Это следует из предположения о надёжности канала и алгоритма на рисунке 2.

<2>2. Доказываемое утверждение следует из шага <2>1 и означает, что требование живучести нарушается.

<1>2. Рассмотрим случай $\exists n \in v_{B_Y}(\{Y\}) \setminus \cup_{F \in S_Y} N(F)$. Предположим, что есть локальный клиент B_Y, публикующий n.

<2>1. n доставлено Y. Это следует из предположения о надёжности канала и алгоритма на рисунке 2.

<2>2. Доказываемое утверждение следует из шага <2>1 и означает, что требование безопасности нарушается.

<1>3. Исходное утверждение вытекает из шагов <1>1 и <1>2, охватывающих все случаи, и означает, что требования безопасности и живучести нарушаются в любом случае.

Теорема 2. Допустимая маршрутная конфигурация необходима для статической системы публикации/подписки, использующей алгоритм на рисунке 2, для удовлетворения требованиям безопасности и живучести. Доказательство теоремы следует из лемм 6 и 7.

Следствие. Допустимая маршрутная конфигурация необходима и достаточна для статической системы публикации/подписки, использующей алгоритм, изображённый на рисунке 2, для удовлетворения требования безопасности и живучести [7].

Доказательство следует из теорем 1 и 2.

Следовательно, допустимая маршрутная конфигурация играет основополагающую роль в статической системе публикации/подписки. Она необходима и достаточна для реализации корректной статической системы публикации/подписки [7]. Покажем, что допустимость – слишком сильное требование в динамической системе публикации/подписки, достаточна слабая допустимость маршрутной конфигурации.

Динамическая система публикации/подписки: слабо допустимая маршрутная конфигурация. Динамическая система публикации/подписки должна иметь дело с новыми подписками и отменой существующих подписок. Предположим, что топология брокера статична, в этом случае маршрутная конфигурация не может гарантировать доставку всех уведомлений, соответствующих новой подписке. Для уверенности в доставке всех интересующих уведомлений необходимо обновить локальную маршрутную конфигурацию брокера подписавшегося клиента, а также удалённой части других брокеров. В отличие от новой подписки отменённые подписки не требуют обновления в удаленной части маршрутной конфигурации. Здесь достаточно своевременно изменить локальную часть для того, чтобы предотвратить доставку уведомлений, которые больше не представляют интерес для клиентов, отменивших подписки. Тем не менее, дальнейшая обработка аннулированных подписок желательна по соображениям эффективности: без соответствующей обработки отменённых подписок, набор уведомлений, перенаправляемых брокером своим соседям, будет монотонно возрастать, что приводит к растратам ресурсов сети. Таким образом, разумный алгоритм маршрутизации на основе содержимого должен постоянно обновлять маршрутную конфигурацию в ответ на каждую новую подписку и отмену подписки клиентом.

Обновление локальной конфигурации брокера может осуществляться без какого-либо взаимодействия с другими брокерами. В отличие от этого, обновление удалённой конфигурации брокера требует сложных и распределенных процессов обновления. Они могут проводиться периодически в определенные моменты времени или апериодически – в ответ на сообщения, получаемые от локальных клиентов. Может быть использован гибридный подход, в котором, например, обновления, вызываемые новыми подписками, распространяются немедленно, а обновления, вызываемые отменами подписок, обрабатываются периодически. В работе рассматривается апериодический сценарий, в котором отслеживаются все изменения в маршрутной конфигурации, вызванные новыми подписками или отменой подписок. Изменение локальной конфигурации осуществляется немедленно, а обновления удалённой части инициируются брокером, который обслуживает соответствующего клиента, и проходят через всю топологию брокерской сети.

В общем случае многие процессы обновления, вызванные новой подпиской или отменой подписки, могут осуществляться одновременно, и тогда они, может быть, никогда не завершатся. Поэтому вводится понятие слабо допустимой маршрутной конфигурации, которая налагает менее строгие требования, чем полностью допустимая, но всё же гарантирует, что система является корректной. Слабо допустимая маршрутная конфигурация гарантирует только доставку уведомлений, соответствующих подписке, чей процесс обновления уже завершился.

Допустим, что $\overline{S}_X$ есть подмножество всех активных подписок S_X клиента X, чей процесс обновления завершился. Это означает, что новая подписка добавляется в $\overline{S}_X$, когда процесс её обновления завершится, а если подписка будет отменена, то она удаляется немедленно. Такое определение $\overline{S}_X$ допускает определение «ослабленной» версии I_B и $\eta_{Bj,\,Bi}$:

$$\bar{I}_B = \cup_{X \in L_B} \cup_{F \in \bar{S}_X} N(F), \qquad (7)$$

$$\overline{\eta}_{B_j,\,B_i} = \cup_{B \in V_{B_j,\,B_i}} \bar{I}_B. \qquad (8)$$

Теперь можно ввести определение слабо допустимой маршрутной конфигурации.

Определение 2. Слабо допустимая маршрутная конфигурация. Маршрутная конфигурация слабо допустима, если

$$e(B_i, B_j) \in E \Rightarrow v_{B_i}(\{B_j\}) \supseteq \overline{\eta}_{B_j,\,B_i}$$

(слабая удалённая допустимость),

$$Y \in L_{B_Y} \Rightarrow v_{B_Y}(\{Y\}) = \cup_{F \in S_Y} N(F)$$

(локальная допустимость).

В приведенном определении удалённая допустимость ослаблена, и брокеру требуется лишь направить те уведомления соседям, которые соответствуют подписке, и чей процесс обновления был завершен. Локальная допустимость остается неизменной, так как необходимые изменения в локальной части таблицы маршрутизации осуществляются немедленно. Из определения легко видеть, что допустимая маршрутная конфигурация каждого брокера также слабо допустима, поскольку $\overline{\eta}_{Bj,Bi} \subseteq \eta_{Bj,Bi}$ и $\overline{S}_Y \subseteq S_Y$. Кроме того, маршрутная конфигура-

ция является *слабо совершенной*, если она удовлетворяет первому свойству, в случае, если расширенное отношение заменяется равенством.

Теперь можно доказать, что слабо допустимая маршрутная конфигурация имеет основополагающее значение для обеспечения корректности системы.

Теорема 3. Если алгоритм на рисунке 2 гарантирует, что маршрутная конфигурация слабо допустима, и что каждый процесс обновления заканчивается, то система публикации/подписки удовлетворяет требованиям безопасности и живучести [7].

Доказательство. Это следует из лемм 4 и 5. Так как свойство локальной допустимости не было изменено, а изменения в локальной конфигурации происходят немедленно, то лемма 4 может быть применена непосредственно, подтверждая требование безопасности. Свойство удалённой допустимости изменено таким образом, чтобы рассматривать только те подписки, процесс обновления которых завершен. Поскольку согласно условию теоремы заканчивается каждый процесс обновления, то лемма 5 может быть применена, подтверждая требование живучести системы, если S заменить на $\overline{S}$ и утверждение «допустимый» на «слабо допустимый».

Слабо допустимая маршрутная конфигурация – полезная концепция динамических систем публикации/подписки, поскольку если алгоритм гарантирует, что маршрутная конфигурация слабо допустима, то после завершения процесса обновления соответствующих подписок клиентам гарантируется доставка только интересующих их уведомлений.

Заключение. В работе приведены формальные основы маршрутизации сообщений на базе содержимого, что важно для получения полного представления о поведении систем Публикация/Подписка, использующих данную парадигму. Рассмотрены новые подходы, которые охватывают формализацию конфигурации маршрутизации, её валидность, базу маршрутизации и универсальные критерии корректности системы. Вводится понятие допустимых маршрутных конфигураций, которые гарантируют, что в статической системе публикации/подписки все соответствующие уведомления доставляются потребителю.

В динамической системе публикации/подписки невозможно гарантировать, что конфигурация маршрутизации всегда верна, поэтому введено понятие слабо допустимых конфигураций маршрутизации. Слабо допустимые конфигурации маршрутизации обеспечивают доставку только тех уведомлений, которые соответствуют подписке, и которые уже были включены в настройки маршрутизации.

Библиографический список

1. *Антипов О.В., Пылькин А.Н.* Выбор механизма уведомления в распределённых системах Публикация/Подписка // Математическое и программное обеспечение вычислительных систем: межвуз. сб. науч. тр. – Рязань: РГРТУ, 2011. – С. 187 – 198.

2. *Благодаров А.В., Пылькин А.Н., Скуднев Д.М.* Моделирование процессов функционирования сети Ethernet // Вестник Рязанского государственного радиотехнического университета. – 2007. – № 21. – С. 60 – 64.

3. *Антипов В.А., Гузенко Р.Е.* Трехуровневая метамодель отображения семантики предметной области на структуру XML сообщений // Вестник Рязанского государственного радиотехнического университета. – 2010. – № 31. – С. 43 – 48.

4. *Антипов В.А., Антипов О.В., Чехов А.П.* Построение телемедицинской системы на основе коммуникационной парадигмы Публикация/Подписка // Биомедицинская радиоэлектроника. – М.: Радиотехника. №7. 2012. – С. 64 – 69.

5. *Антипов В.А., Антипов О.В., Богомолов О.М., Кутаков Д.А.* Формирование медицинской виртуальной организации // Биомедицинская радиоэлектроника. – М.: Радиотехника. №7. 2013. – С. 70 – 77.

6. *Пылькин А.Н., Крошилин А.В., Крошилина С.В.* Построение модели оценки состояния здоровья пациента в медицинских экспертных системах // Вестник Рязанского государственного радиотехнического университета. – 2012. – № 41. – С. 64 – 70.

7. *Антипов В.А., Антипов О.В., Пылькин А.Н.* Интеграция распределенных программных приложений на основе коммуникационной парадигмы Публикация/Подписка // Математические и компьютерные методы в технических, гуманитарных и общественных науках: колл. моног. Вып. 3 – М: Приволжский Дом знаний, 2013. – С. 36 – 67.

УДК 004.72:519.2

Д.А. Перепелкин

АЛГОРИТМ ПАРНЫХ ПЕРЕСТАНОВОК МАРШРУТОВ НА БАЗЕ ПРОТОКОЛА OSPF ПРИ ДИНАМИЧЕСКОМ ОТКАЗЕ УЗЛОВ И ЛИНИЙ СВЯЗИ КОРПОРАТИВНОЙ СЕТИ

Предложен алгоритм адаптивной ускоренной маршрутизации на базе протокола OSPF при динамическом отказе узлов и линий связи корпоративной сети, повышающий эффективность ее функционирования.

Ключевые слова: *адаптивная ускоренная маршрутизация, алгоритмы маршрутизации, алгоритм парных перестановок маршрутов, протокол OSPF, динамические изменения, динамические корпоративные сети, программно-конфигурируемые сети.*

Введение. Быстрое развитие сетевых технологий требует обеспечения качественного обслуживания современного трафика. В связи с этим большое значение приобретают вопросы эффективности применяемых процессов маршрутизации. Задача маршрутизации в корпоративных сетях решается при условии, что оптимальный маршрут, обеспечивающий передачу пакета за минимальное время, зависит от топологии сети, пропускной способности и нагрузки на линии связи. Топология сети изменяется в результате отказов узлов и линий связи телекоммуникационной системы.

Применение новых перспективных подходов для решения задачи маршрутизации позволяет повысить эффективность функционирования корпоративной сети за счет уменьшения трудоемкости построения оптимальных маршрутов.

Цель работы – разработка нового эффективного алгоритма поиска оптимальных маршрутов на базе протокола OSPF при динамическом отказе узлов и линий связи корпоративной сети, повышающего эффективность ее функционирования.

Постановка задачи. Для повышения эффективности функционирования корпоративных сетей наиболее важной задачей является выбор алгоритма маршрутизации, который будет обеспечивать поиск оптимальных маршрутов с учетом различных свойств той или иной корпоративной сети. В настоящее время широкое применение получили алгоритмы адаптивной маршрутизации. Эти алгоритмы обеспечивают автоматическое обновление таблиц маршрутизации после изменения конфигурации сети.

Протокол OSPF (Open Shortest Path First) при решении задачи маршрутизации базируется на алгоритме состояния каналов. Характеристики и параметры качества обслуживания протокола OSPF подробно рассматриваются в работах [1-3]. Выбор оптимального маршрута в протоколе OSPF определяется по алгоритму Дейкстры. Трудоемкость построения таблиц маршрутизации с использованием данного алгоритма составляет порядка $O(N^2)$, где N – число маршрутизаторов в корпоративной сети.

Развитие в последнее время технологии программно-конфигурируемых сетей [4, 5] позволяет формулировать различные задачи оптимальной маршрутизации и балансировки сетевого трафика [6, 7].

В работах [8, 9] предложен алгоритм парных переходов, позволяющий за счет сбора дополнительной информации учесть возможные изменения параметров линий связи корпоративной сети и не производить полный пересчет маршрутных таблиц. Это позволило снизить трудоемкость расчета таблиц маршрутизации до величины порядка $O(k \cdot N)$, где k – число фактически выполненных парных переходов.

В работах [10-12] предложены эффективные алгоритмы адаптивной ускоренной маршрутизации при динамических изменениях в структуре корпоративной сети, которые также позволили снизить трудоемкость построения оптимальных маршрутов передачи данных до величины $O(k \cdot N)$. Недостатком предложенных алгоритмов является то, что при каждом динамическом изменении в структуре корпоративной сети после каждого парного перехода необходимо рассчи-

тывать дополнительную информацию для того, чтобы определить оптимальный маршрут до других узлов в сети.

В работах [13-16] предложен алгоритм парных перестановок маршрутов при динамических изменениях параметров линий связи корпоративной сети, что позволило снизить трудоемкость построения оптимальных маршрутов передачи данных до величины $O(N)$. В работе [17] предложен алгоритм парных перестановок маршрутов на базе протокола OSPF при динамическом подключении узлов и линий связи корпоративной сети. Однако в данных работах алгоритм парных перестановок маршрутов не учитывает возможность динамического отказа узлов и линий связи корпоративной сети. В связи с этим трудоемкость построения оптимальных маршрутов передачи данных оказывается выше трудоемкости существующих алгоритмов.

Разработка новых более эффективных алгоритмов адаптивной ускоренной маршрутизации позволяет уменьшить трудоемкость построения таблиц маршрутизации до величины $O(N)$ в корпоративных сетях, использующих в своей работе протокол OSPF.

Разработка алгоритма. Для повышения эффективности функционирования корпоративных сетей на базе протокола OSPF предлагается алгоритм парных перестановок маршрутов, который позволяет уменьшить трудоемкость построения таблиц маршрутизации до величины $O(N)$ при динамическом отказе узлов и линий связи корпоративной сети.

В общем случае для решения данной задачи применяется графовая модель корпоративной сети, в которой множество вершин графа соответствует множеству узлов связи или маршрутизаторов в сети, а множество ребер соответствует возможным каналам связи между этими узлами. Каждое ребро, соответствующее каналу связи, 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.

Представим корпоративную сеть в виде неориентированного взвешенного связного графа $G = (V, E, W)$, где V – множество вершин (узлов связи), $|V| = N$, E – множество ребер (каналов или линий связи), $|E| = M$, W – множество весов ребер (стоимость каналов или линий связи).

Пусть на графе G в некоторый момент времени уже решена задача поиска оптимальных маршрутов до всех узлов множества $V_{\hat{s}} = V \setminus \{v_s\}$ из начального узла v_s, т. е. построено дерево оптимальных маршрутов с корнем в узле v_s. Обозначим это дерево как T_g.

Рассмотрим множество каналов E графа G. По признаку вхождения каналов в дерево T_g можно разделить исходное множество E на два подмножества: $E_T \in T_g$ и $E_R \notin T_g$, $E_T \cup E_R = E$.

Множество каналов дерева E_T – множество каналов дерева T_g для графа G. Для заданного графа G согласно свойству дерева мощность множества E_T будет равняться мощности множества V минус единица $|E_T| = |V| - 1$.

Множество каналов замены E_R для дерева – множество каналов графа G, не вошедших в дерево T_g. При соответствующих условиях некоторый канал $e_{i,j} \in E_R$, инцидентный узлам v_i и v_j, может перейти в множество каналов дерева E_T, заменив собой некоторый канал $e_{k,p} \in E_T$. При этом инцидентность канала $e_{k,p}$ узлу v_i или v_j является обязательным условием. В свою очередь, канал $e_{i,j}$ перейдет в множество E_R.

Будем называть такие переходы парными переходами и обозначать $e_{i,j} - e_{k,p}$.

В множестве E_R можно выделить два подмножества.

Множество каналов замены E_S для дерева – это такое подмножество множества E_R, элементы-каналы которого участвуют, по крайней мере, в одном отношении парного перехода.

Множество непарных каналов E_P – это такое подмножество множества E_R, элементы-каналы которого не участвуют ни в одном отношении из множества R.

В общем случае множество E_P может быть пустым $|E_P| = 0$. Множество E_S будет пустым только при условии, что исходный связный граф G является деревом, и задача поиска оптимальных маршрутов в этом случае лишена смысла.

Для каждого канала связи $e_{i,j} \in E$ на шкале значений весов определены точка вхождения в дерево $w_{i,j}^{\ t}$ и точка вхождения в множество замены $w_{i,j}^{\ s}$, причем $w_{i,j}^{\ t} \le w_{i,j}^{\ s}$, под которыми понимается максимально возможный вес канала $e_{i,j}$ при его вхождении в множество каналов дерева $E_T \in T_g$ и в множество каналов замены для дерева $E_S \notin T_g$ соответственно.

Обозначим $w_{i,j}$ – вес канала, соединяющего узлы v_i и v_j. Узел v_i располагается ниже по иерархии в дереве оптимальных маршрутов относительно v_j. Множество E_T – множество каналов, каждый элемент которого входит, по крайней мере, в один оптимальный маршрут из начального узла связи, E_R – множество остальных каналов. $E_R \cup E_T = E$, $E_R \cap E_T = \varnothing$. Обозначим V_T – множество узлов, до которых найден оптималь-

ный маршрут из начального узла связи, V_R – множество остальных узлов. $V_R \cup V_T = V$, $V_R \cap V_T = \varnothing$.

Обозначим множество маршрутов до узла v_i из исходного узла v_s через Π_i, где элемент множества $\pi_{i,k} \in \Pi_i$ будет множеством не повторяющихся каналов $e_{i,j} \in E$, образующих вместе маршрут, соединяющий v_s и v_i. Каждому $\pi_{i,k} \in \Pi_i$ поставим в соответствие число, равное сумме весов входящих в него каналов, т. е. длину маршрута $d_{i,k} \in D_i$, где D_i представляет собой множество оценок оптимальных маршрутов до узла v_i из исходного узла v_s. На множестве Π_i задан селектор H, возвращающий оптимальный маршрут из множества Π_i. В том случае, если существует несколько маршрутов в Π_i с минимальной длиной, то выбирается один из них. Оптимальный маршрут до узла v_i будем обозначать $\pi_i = H(\Pi_i)$, оценку его длины – d_i.

Для разработки алгоритма парных перестановок маршрутов на базе протокола OSPF при динамическом отказе узлов и линий связи корпоративной сети сформулируем следующие теоремы.

Теорема 1. При отказе некоторого канала связи $e_{i,j}$, инцидентного узлам связи V_i и V_j, входящего в дерево оптимальных маршрутов, причем $d_i < d_j$, оптимальные маршруты и их оценки до узла V_i окажутся без изменения.

Доказательство. Узлы связи V_i и V_j входят в дерево оптимальных маршрутов. Так как $d_i < d_j$, то $d_i + w_{i,j} = d_j$ и канал $e_{i,j}$ не входит в маршрут π_i к узлу V_i. Поэтому при отказе канала $e_{i,j}$ оптимальные маршруты и их оценки до узла V_i не изменятся. Теорема доказана.

Следствие 1. При отказе некоторого канала связи $e_{i,j}$, инцидентного узлам связи V_i и V_j, входящего в дерево оптимальных маршрутов, причем $d_i < d_j$, необходимо определить новые оптимальные маршруты для множества узлов, инцидентных узлу V_j.

Следствие 2. При отказе некоторого канала связи $e_{i,j}$, инцидентного узлам связи V_i и V_j, входящего в дерево оптимальных маршрутов, причем $d_i < d_j$, необходимо в списках маршрутов до каждого узла исключить те маршруты, которые включали канал $e_{i,j}$.

Теорема 2. При отказе некоторого узла связи V_i для всех узлов, не инцидентных узлу V_i (то есть не имеющих канала связи $e_{i,k}$), оптимальные маршруты и их оценки окажутся без изменения.

Доказательство. Пусть узел связи V_i не входит в оптимальный маршрут к некоторому узлу V_j. Тогда и в маршруте к этому узлу $\pi_{v,j}$ отсутствует канал $e_{i,k}$. При отказе узла V_i будут исключены все каналы $e_{i,k}$, которые не входят в маршрут $\pi_{v,j}$, следовательно, оптимальный маршрут и его оценка окажутся без изменения. Теорема доказана.

Следствие 1. При отказе некоторого узла связи V_i необходимо определить новые оптимальные маршруты для множества узлов, инцидентных узлу V_i, имеющих в своем оптимальном маршруте канал $e_{i,k}$.

Следствие 2. При отказе некоторого узла связи V_i необходимо в списках маршрутов исключить все маршруты до узла V_i, а также те маршруты, которые включали инцидентные узлу V_i каналы (то есть имели каналы $e_{i,k}$).

Использование доказанных выше теорем позволяет разработать алгоритм парных перестановок маршрутов на базе протокола OSPF, уменьшающий размерность задачи поиска оптимальных маршрутов при динамическом отказе узлов и линий связи корпоративной сети.

Рассмотрим работу алгоритма парных перестановок маршрутов на базе протокола OSPF в корпоративных сетях. Укрупненно алгоритм имеет следующий вид.

Шаг 1. Первоначальная инициализация исходных данных. Используя пакет HELLO протокола OSPF, определить веса линий связи $w_{i,j}$.

Шаг 2. Построить дерево оптимальных маршрутов корпоративной сети.

Шаг 3. Для узла связи, являющегося листом дерева, произвести поиск всех парных переходов без ограничений. Эти списки для удобства дальнейшей работы привязываются к узлу, инцидентному рассматриваемому каналу связи и расположенному ниже по иерархии.

Шаг 4. Если узел связи не является листом дерева, то вычислить парные переходы для этого узла и выбрать лучшие значения потенциалов парных переходов для потомков узла и собственных парных переходов. Подобную процедуру выполнить для формирования списков парных переходов в случае динамического отказа узлов и линий связи корпоративной сети.

Шаг 5. Для каждого узла сформировать полный список парных переходов. Число элементов в каждом из этих списков не превышает количество узлов графа. Такое решение позволяет отказаться от предварительной сортировки потенциалов или приращений для парных переходов без значительного усложнения алгоритма обработки изменения.

Шаг 6. Для каждого канала связи корпоративной сети определить точку вхождения в дерево оптимальных маршрутов и точку вхождения в множество замены.

Шаг 7. Для каждого узла связи корпоративной сети сформировать полный список возможных маршрутов, проходящий через каналы, состоящие в отношении парного перехода, включая и каналы, входящие в дерево оптимальных маршрутов.

Шаг 8. Определить, есть ли пакеты на передачу:

а) если да, то перейти к шагу 9;

б) иначе – к шагу 17.

Шаг 9. Используя поля «Время жизни» и «Контрольная сумма заголовка» протокола IP, определить, требуется ли уничтожить (отбросить) данный пакет:

а) если да, то перейти к шагу 23;

б) иначе – к шагу 10.

Шаг 10. Используя поле «Тип сервиса», определить, требуется ли создание виртуального соединения:

а) если да, то перейти к шагу 11;

б) иначе – к шагу 13.

Шаг 11. Организовать виртуальное соединение, послав первый пакет «запрос вызова» адресату:

а) если пакет согласия на соединение от адресата пришел, то перейти к шагу 14;

б) если адресат отклонил вызов, то перейти к пункту 13.

Шаг 12. а) установить флаг передачи;

б) послать адресату пакет ликвидации соединения;

в) получить пакет подтверждения рассоединения от адресата.

Шаг 13. Проверка флага передачи:

а) если флаг установлен, то перейти к шагу 23;

б) иначе – к шагу 8.

Шаг 14. Используя поле «Тип сервиса», определить необходимую таблицу маршрутизации с учетом желаемого уровня качества обслуживания:

а) если таблица 1, то перейти к шагу 15;

б) иначе – к шагу 16.

Шаг 15. а) передать пакет, используя первую таблицу маршрутизации;

б) перейти к шагу 12.

Шаг 16. а) передать пакет, используя вторую таблицу маршрутизации;

б) перейти к шагу 12.

Шаг 17. Анализируя полученную протоколом OSPF информацию, определить, произошел ли отказ узлов или линий связи в структуре корпоративной сети:

а) если да, то перейти к шагу 18;

б) иначе – к шагу 8.

Шаг 18. а) *если произошел отказ канала связи* $e_{i,j}$ *с весом* $w_{i,j}$, *причем узел связи* V_i *расположен ниже по иерархии в дереве оптимальных маршрутов, чем узел* V_j *то:*

1) канал связи $e_{i,j}$ не входит в дерево оптимальных маршрутов. Дерево оптимальных маршрутов не изменится. Если канал связи $e_{i,j}$ не входил в множество каналов замены, то оставить все без изменения. Если канал связи $e_{i,j}$ входил в множество каналов замены, то для узла связи V_j определить новые каналы, входящие в множество каналов замены, и определить для них точки вхождения в дерево оптимальных маршрутов. Исключить возможные маршруты замены, проходящие через канал связи $e_{i,j}$, и подключить новые маршруты, проходящие через новые каналы замены для узла связи V_j;

2) канал связи $e_{i,j}$ входит в дерево оптимальных маршрутов. Дерево оптимальных маршрутов до узла связи V_i и всех инцидентных ему узлов не изменится. Для узла V_j из списка маршрутов замены выбрать маршрут с минимальной оценкой и включить его в дерево оптимальных маршрутов, то есть выполнить парную перестановку. Исключить из множества маршрутов замены все маршруты, проходящие через канал связи $e_{i,j}$. Для всех узлов, инцидентных узлу V_j (кроме узла V_i), определить их новые оценки, выбрать маршрут с минимальной длиной и включить его в дерево оптимальных маршрутов. Определить инцидентные узлу V_j каналы, состоящие в отношении парного перехода, и включить их в множество каналов замены. Для узла связи V_j и всех инцидентных ему узлов (кроме узла V_i) определить новые точки вхождения каналов в множество каналов замены и в дерево оптимальных маршрутов. Переформировать список оптимальных маршрутов и их маршрутов замены для узла V_j и всех инцидентных ему узлов (кроме узла V_i).

б) *если произошел отказ узла связи* V_k, *имеющего каналы связи с узлами* V_i *и* V_j, *причем узел* V_i *расположен ниже по иерархии в дереве оптимальных маршрутов, чем узел* V_j, *то:*

1) для всех узлов V_i, не инцидентных узлу V_k, то есть не имеющих канала связи $e_{i,k}$, дерево оптимальных маршрутов не изменится;

2) для всех узлов V_j, инцидентных узлу V_k, имеющих канал связи $e_{k,j}$, не входящий в дерево оптимальных маршрутов, дерево оптимальных маршрутов не изменится. Исключить канал связи $e_{k,j}$ из множества каналов замены. Исключить все возможные маршруты замены, проходящие через канал связи $e_{k,j}$;

3) для всех узлов V_m, инцидентных узлу V_k, имеющих канал связи $e_{k,m}$, входящий в дерево оптимальных маршрутов, выполнить парные перестановки маршрутов, то есть из списка маршрутов замены выбрать маршрут с мини-

мальной оценкой. Исключить канал связи $e_{k,m}$ из множества каналов графа корпоративной сети. Исключить все возможные маршруты замены, проходящие через канал связи $e_{k,m}$;

4) в списках маршрутов графа корпоративной сети исключить все маршруты до узла V_k, а также маршруты, которые включали инцидентные узлу V_k каналы связи (то есть каналы $e_{i,k}$ и $e_{k,j}$).

Шаг 19. Используя список парных переходов, определить, требуется ли сделать парный переход:

а) если да, то перейти к шагу 20;

б) иначе – к шагу 21.

Шаг 20. Для каждого узла связи, у которого в списках возможных маршрутов произошли изменения, определить путь минимальной длины и поместить его в дерево оптимальных маршрутов.

Шаг 21. Построить новое дерево оптимальных маршрутов с учетом изменений.

Шаг 22. Сформировать таблицы маршрутизации.

Шаг 23. Проверка окончания работы маршрутизатора:

а) если да, то перейти к шагу 24;

б) иначе – сбросить флаг передачи и перейти к шагу 8.

Шаг 24. Завершение работы маршрутизатора.

Пример работы алгоритма. Рассмотрим работу алгоритма парных перестановок маршрутов при динамическом отказе узлов и линий связи на примере графа G корпоративной сети, показанного на рисунке 1, в котором уже решена задача поиска оптимальных маршрутов и построено соответствующее дерево.

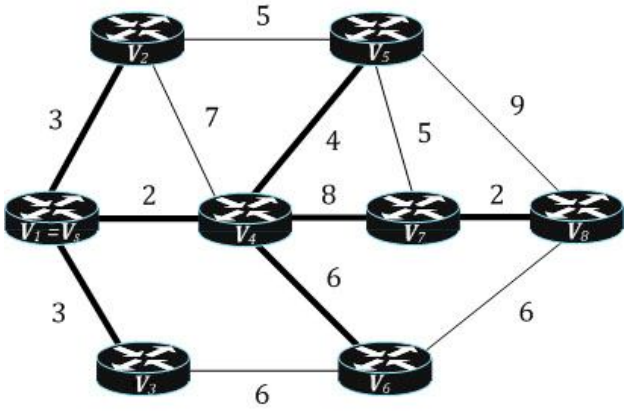


Рисунок 1 – Граф *G* корпоративной сети

Для представленного графа G множество каналов дерева составляет $E_T = \{e_{1,2}; e_{1,3}; e_{1,4}; e_{4,5}; e_{4,6}; e_{4,7}; e_{7,8}\}$; множество каналов замены $E_S = \{e_{2,4}; e_{2,5}; e_{3,6}; e_{5,7}; e_{6,8}\}$; множеством непарных каналов будет $E_P = \{e_{5,8}\}$. Если рассмотреть канал связи $e_{4,5}$, то для него точка вхождения в дерево будет составлять 6, а точка вхождения в множество замены – 13. При этом данный канал связи находится в отношении парного перехода с каналом $e_{2,5}$, который, в свою очередь, находится в отношении парного перехода множества замены с каналом $e_{5,7}$. После попадания $e_{4,5}$ в множество непарных каналов эта парная перестановка примет вид: $e_{2,5} – e_{5,7}$.

После того как сформирован список парных переходов, дополнительно рассчитываем список оптимальных маршрутов и их маршрутов замены из исходного узла связи до каждого узла сети.

Таким образом, списки оптимальных маршрутов и их маршрутов замены до всех узлов графа корпоративной сети из исходного узла v_s будут сформированы следующим образом.

Узел связи V_2

Маршрут $\pi_2 = \{e_{1,2}\}$ с оценкой $d_2 = 3$ – оптимальный маршрут;

маршрут $\pi_2^{(1)} = \{e_{1,4}; e_{4,2}\}$ с оценкой $d_2^{(1)} = 2 + 7 = 9$ – маршрут замены;

маршрут $\pi_2^{(2)} = \{e_{1,4}; e_{4,5}; e_{5,2}\}$ с оценкой $d_2^{(2)} = 2 + 4 + 5 = 11$ – маршрут замены.

Узел связи V_3

Маршрут $\pi_3 = \{e_{1,3}\}$ с оценкой $d_3 = 3$ – оптимальный маршрут;

маршрут $\pi_3^{(1)} = \{e_{1,4}; e_{4,6}; e_{6,3}\}$ с оценкой $d_2^{(1)} = 2 + 6 + 6 = 14$ – маршрут замены.

Узел связи V_4

Маршрут $\pi_4 = \{e_{1,4}\}$ с оценкой $d_4 = 2$ – оптимальный маршрут;

маршрут $\pi_4^{(1)} = \{e_{1,2}; e_{2,4}\}$ с оценкой $d_4^{(1)} = 3 + 7 = 10$ – маршрут замены;

маршрут $\pi_4^{(2)} = \{e_{1,2}; e_{2,5}; e_{5,4}\}$ с оценкой $d_4^{(2)} = 3 + 5 + 4 = 12$ – маршрут замены;

маршрут $\pi_4^{(3)} = \{e_{1,3}; e_{3,6}; e_{6,4}\}$ с оценкой $d_4^{(3)} = 3 + 6 + 6 = 15$ – маршрут замены.

Узел связи V_5

Маршрут $\pi_5 = \{e_{1,4}; e_{4,5}\}$ с оценкой $d_5 = 2 + 4 = 6$ – оптимальный маршрут;

маршрут $\pi_5^{(1)} = \{e_{1,2}; e_{2,5}\}$ с оценкой $d_5^{(1)} = 3 + 5 = 8$ – маршрут замены;

маршрут $\pi_5^{(2)} = \{e_{1,4}; e_{4,7}; e_{7,5}\}$ с оценкой $d_5^{(2)} = 2 + 8 + 5 = 15$ – маршрут замены;

маршрут $\pi_5^{(2)} = \{e_{1,2}; e_{2,4}; e_{4,5}\}$ с оценкой $d_5^{(3)} = 3 + 7 + 4 = 14$ – маршрут замены.

Узел связи V_6

Маршрут $\pi_6 = \{e_{1,4}; e_{4,6}\}$ с оценкой $d_6 = 2 + 6 = 8$ – оптимальный маршрут;

маршрут $\pi_6^{(1)} = \{e_{1,3}; e_{3,6}\}$ с оценкой $d_6^{(1)} = 3 + 6 = 9$ – маршрут замены.

Узел связи V_7

Маршрут $\pi_7 = \{e_{1,4}; e_{4,7}\}$ с оценкой $d_7 = 2 + 8 = 10$ – оптимальный маршрут;

маршрут $\pi_7^{(1)} = \{e_{1,4}; e_{4,5}; e_{5,7}\}$ с оценкой $d_7^{(1)} = 2 + 4 + 5 = 11$ – маршрут замены;

маршрут $\pi_7^{(2)} = \{e_{1,2}; e_{2,5}; e_{5,7}\}$ с оценкой $d_7^{(2)} = 3 + 5 + 5 = 13$ – маршрут замены.

Узел связи V_8

Маршрут $\pi_8 = \{e_{1,4}; e_{4,7}; e_{7,8}\}$ с оценкой $d_8 = 2 + 8 + 2 = 12$ – оптимальный маршрут;

маршрут $\pi_8^{(1)} = \{e_{1,4}; e_{4,6}; e_{6,8}\}$ с оценкой $d_8^{(1)} = 2 + 6 + 6 = 14$ – маршрут замены;

маршрут $\pi_8^{(2)} = \{e_{1,3}; e_{3,6}; e_{6,8}\}$ с оценкой $d_8^{(2)} = 3 + 6 + 6 = 15$ – маршрут замены;

маршрут $\pi_8^{(3)} = \{e_{1,4}; e_{4,5}; e_{5,7}; e_{7,8}\}$ с оценкой $d_8^{(3)} = 2 + 4 + 5 + 2 = 13$ – маршрут замены.

При динамическом отказе канала связи $e_{4,7}$ граф G корпоративной сети примет вид, показанный на рисунке 2.

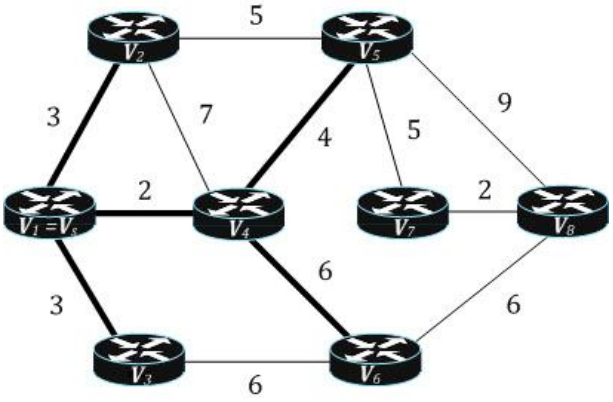


Рисунок 2 – Отказ канала связи $e_{4,7}$

На данном рисунке жирными линиями обозначено дерево оптимальных маршрутов, которое не требует изменения.

То есть при отказе канала связи $e_{4,7}$ дерево оптимальных маршрутов до узлов V_2, V_3, V_4, V_5 и V_6 согласно теореме 1 не изменится. Для узлов V_7 и V_8 необходимо пересчитать точки вхождения в дерево и в множество замены, а также переформировать списки оптимальных маршрутов и их маршрутов замены с учетом отказанного канала связи $e_{4,7}$. Оптимальный маршрут до узла V_7 составит $\pi_7 = \{e_{1,4}; e_{4,5}; e_{5,7}\}$ с оценкой $d_7 = 2 + 4 + 5 = 11$. Оптимальный маршрут до узла V_8 составит $\pi_8 = \{e_{1,4}; e_{4,5}; e_{5,7}; e_{7,8}\}$ с оценкой $d_8 = 2 + 4 + 5 + 2 = 13$.

Таким образом, для графовой модели корпоративной сети, представленной на рисунке 2, множество каналов дерева составит $E_T = \{e_{1,2}; e_{1,3}; e_{1,4}; e_{4,5}; e_{4,6}; e_{5,7}; e_{7,8}\}$; множество каналов замены $E_S = \{e_{2,4}; e_{2,5}; e_{3,6}; e_{6,8}\}$; множество непарных каналов $E_P = \{e_{5,8}\}$.

При динамическом отказе узла связи V_7 граф G корпоративной сети примет вид, показанный на рисунке 3. На данном рисунке жирными линиями обозначено дерево оптимальных маршрутов, которое не требует изменения.

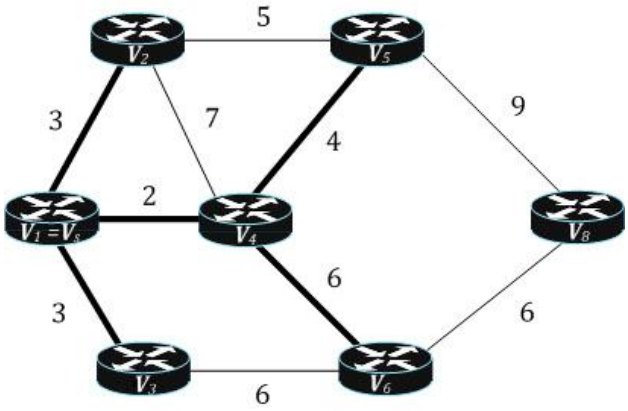


Рисунок 3 – Отказ узла связи V_7

То есть при отказе узла связи V_7 дерево оптимальных маршрутов до узлов V_2, V_3, V_4, V_5 и V_6 согласно теореме 2 не изменится. Для узла V_8 необходимо пересчитать точки вхождения в дерево и в множество замены, а также переформировать списки оптимальных маршрутов и их маршрутов замены с учетом отказавшего узла связи V_7 . Оптимальный маршрут до узла V_8 составит $\pi_8 = \{e_{1,4}; e_{4,6}; e_{6,8}\}$ с оценкой $d_8 = 2 + 6 + 6 = 14$.

Таким образом, для графовой модели корпоративной сети, представленной на рисунке 3, множество каналов дерева составит $E_T = \{e_{1,2}; e_{1,3}; e_{1,4}; e_{4,5}; e_{4,6}; e_{6,8}\}$; множество каналов замены $E_S = \{e_{2,4}; e_{2,5}; e_{3,6}; e_{5,8}\}$.

Результаты моделирования. Для подтверждения правильности алгоритма парных перестановок маршрутов на базе протокола OSPF при динамическом отказе узлов и линий связи корпоративной сети разработано программное обеспечение моделирования процессов маршрутизации.

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

Также при проектировании и создании программных средств рассматривался подход, предложенный в работе [18].

На рисунках 4 – 6 представлены результаты моделирования алгоритма парных перестановок маршрутов на базе протокола OSPF.

Для каждого испытания на множестве обработанных изменений выбирались минимальное, максимальное и среднее значения размерности задачи, выраженные через количество узлов связи, для которых необходим поиск оптимальных маршрутов. Для каждого эксперимента были найдены значения оценок математического ожидания и среднего квадратичного отклонения числа изменений. Для алгоритма парных перестановок маршрутов на базе протокола OSPF определялось число фактически выполненных

перестановок маршрутов при динамическом отказе узлов и линий связи корпоративной сети.

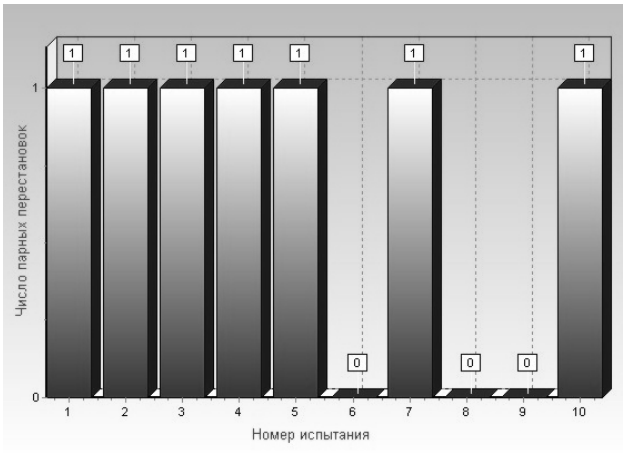


Рисунок 4 – Число изменений корпоративной сети из 10 узлов

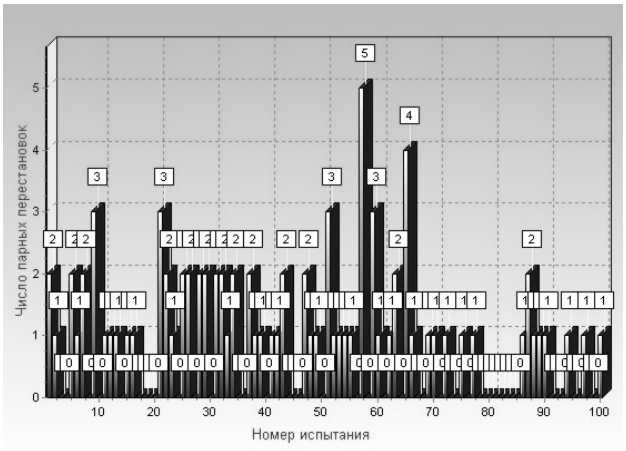


Рисунок 5 – Число изменений корпоративной сети из 100 узлов

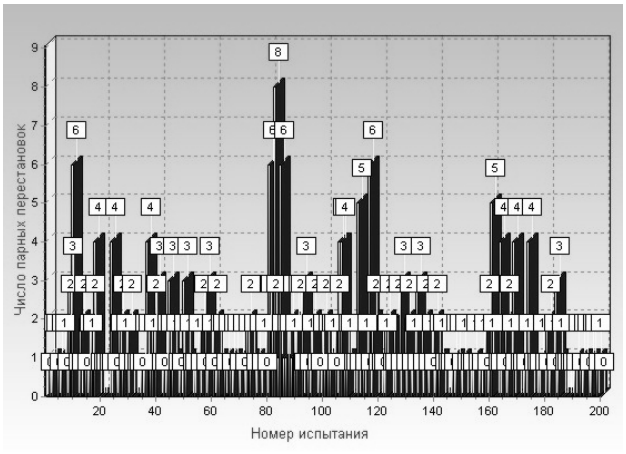


Рисунок 6 – Число изменений корпоративной сети из 200 узлов

Были проведены исследования графовых моделей корпоративных сетей, состоящих из 10, 100 и 200 узлов связи. Исследование разработанного алгоритма парных перестановок маршрутов на базе протокола OSPF показало, что максимальное значение числа изменений существенно меньше размерности для каждой из рассмотренных моделей сети, а значение оценки математического ожидания не превышает единицы. Более того, обнаружена тенденция уменьшения значения оценки математического ожидания числа изменений дерева оптимальных маршрутов с увеличением количества узлов связи в сети.

В таблице для различного числа узлов связи (N) графа корпоративной сети приведены обобщенные статистические характеристики числа изменений дерева маршрутов: минимальное (*Min*) и максимальное (*Max*) значения, значения оценок математического ожидания (МО) и среднего квадратичного отклонения (СКО).

N	*Min*	*Max*	МО	СКО
10	0	0,1	0,07	0,0251
100	0	0,05	0,0087	0,0080
200	0	0,04	0,0054	0,0061

При динамическом отказе узлов и линий связи корпоративной сети разработанный алгоритм парных перестановок маршрутов на базе протокола OSPF в отличие от алгоритма Дейкстры позволяет производить перестроение таблиц маршрутизации не полностью, а только той ее части, в которой произошли изменения. Для этого необходимо один раз просмотреть списки оптимальных маршрутов и их маршрутов замены только для тех узлов сети, оценки длин которых изменились при динамическом отказе узлов и линий связи корпоративной сети, и на основе предварительно собранной информации о списках маршрутов выполнить изменения для вычисления новых оптимальных маршрутов. При этом трудоемкость изменения дерева оптимальных маршрутов является линейной функцией от числа узлов сети и определяется выражением $O(N)$. Если рассмотреть полученные результаты при моделировании предложенного алгоритма парных перестановок маршрутов и сравнить их с алгоритмом Дейкстры, то видно:

1) для корпоративной сети из 10 узлов связи максимальное число парных перестановок маршрутов составляет одно изменение (= 1), при этом для алгоритма Дейкстры при каждом отказе узлов и линий связи трудоемкость построения таблицы маршрутизации составляет $O(N^2)$, т. е. порядка $10^2 = 100$ элементарных операций;

2) для корпоративной сети из 100 узлов связи максимальное число парных перестановок маршрутов составляет пять изменений (= 5), при этом для алгоритма Дейкстры при каждом отказе узлов и линий связи трудоемкость построения таблицы маршрутизации составляет $O(N^2)$, т.е. порядка $100^2 = 10\,000$ элементарных операций;

3) для корпоративной сети из 200 узлов связи максимальное число парных перестановок маршрутов составляет восемь изменений (= 8), при этом для алгоритма Дейкстры при каждом отказе узлов и линий связи трудоемкость построения таблицы маршрутизации составляет $O(N^2)$, т. е. порядка $200^2 = 40\,000$ элементарных операций.

На основе этого можно сделать вывод, что предложенный алгоритм парных перестановок маршрутов на базе протокола OSPF является эффективным при поиске оптимальных маршрутов в условиях динамических изменений в структуре корпоративной сети за счет использования дополнительной информации о возможных маршрутах замены.

Заключение. Разработанный алгоритм парных перестановок маршрутов на базе протокола OSPF позволяет повысить эффективность функционирования корпоративной сети за счет уменьшения трудоемкости построения таблиц маршрутизации до величины порядка $O(N)$ при динамическом отказе узлов и линий связи в ее структуре.

Работа выполнена при финансовой поддержке гранта Президента РФ для молодых ученых кандидатов наук МК-819.2014.9.

Библиографический список

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

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

3. *Корячко В.П., Перепелкин Д.А.* Анализ и проектирование маршрутов передачи данных в корпоративных сетях. – М.: Горячая линия – Телеком, 2012. – 235 с.

4. *Turner J.* Openflow: Enabling innovation in campus networks // SIGCOMM Computer Communication Review. – 2008. – Vol. 38, No 2. – P. 69-74.

5. *Шибанов А.П., Корячко В.П., Ижванов Ю.Л.* Моделирование агрегированного телекоммуникационного канала с технологией открытых потоков // Радиотехника. – 2012. – № 3. – С. 109-112.

6. *Ижванов Ю.Л.* Динамическая оценка состояния компьютерных сетей на основе метода сетевой томографии и задачи балансировки трафика // Информатизация образования и науки. – 2013. – № 3 (19). – С. 35-40.

7. *Корячко В.П., Перепелкин Д.А., Иванчикова М.А.* Алгоритм адаптивной маршрутизации в корпоративных сетях нескольких провайдеров связи // Вестник Рязанского государственного радиотехнического университета. – 2013. – № 2 (44). – С. 52-56.

8. *Уваров Д.В., Перепелкин А.И., Корячко В.П.* Построение дерева кратчайших путей в графе на основе данных о парных переходах // Системы управления и информационные технологии. – 2004. – № 4 (16). – С. 93-96.

9. *Корячко В.П., Перепелкин Д.А.* Построение дерева оптимальных маршрутов корпоративной сети в условии динамического изменения нагрузки на ее линиях связи // Информационные технологии: межвуз. сб. науч. тр. РГРТУ. – 2011. – С. 7-18.

10. *Перепелкин Д.А., Перепелкин А.И.* Разработка алгоритмов адаптивной маршрутизации в корпоративных вычислительных сетях // Вестник Рязанского государственного радиотехнического университета. – 2006. – № 19. – С. 114-116.

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

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

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

14. *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. – 2010. – № 33. – С. 49-55.

15. *Перепелкин Д.А., Перепелкин А.И.* Повышение качества функционирования корпоративных сетей на базе протокола OSPF // Качество. Инновации. Образование. – 2010. – № 12. – С. 51-56.

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

17. *Перепелкин Д.А.* Алгоритм парных перестановок маршрутов на базе протокола OSPF при динамическом подключении узлов и линий связи корпоративной сети // Вестник Рязанского государственного радиотехнического университета. – 2013. – № 4-1 (46). – С. 67-75.

18. *Пруцков А.В., Цыбулько Д.М.* Проблемно-ориентированное объектное программирование // Вестник Рязанского государственного радиотехнического университета. – 2013. – № 3 (45). – С. 57-62.

УДК 004.4

А.В. Пруцков, Д.М. Цыбулько

ПРИМЕНЕНИЕ ПРОБЛЕМНО-ОРИЕНТИРОВАННОГО ОБЪЕКТНОГО ПРОГРАММИРОВАНИЯ ДЛЯ ОПИСАНИЯ ПОРЯДКА РАБОТЫ ИНТЕЛЛЕКТУАЛЬНЫХ И ИНФОРМАЦИОННЫХ СИСТЕМ

Предложено описывать порядок действия систем различного назначения с помощью разработанного авторами проблемно-ориентированного объектного программирования (ПООП). Такой подход позволяет использовать ПООП специалистам, которые не являются профессиональными программистами. Приведены примеры описаний порядка работы системы логического вывода знаний и системы управления информационным табло. Показана область применения ПООП, а также возможность реализации управляющих конструкций языков программирования единственной конструкцией ПООП. Сформулированы принципы ПООП, определяющие направление его развития и эволюции.

***Ключевые слова**: языки программирования, объектно-ориентированное программирование, проблемно-ориентированное программирование.*

Введение. В работе [1] был предложен новый подход к программированию, названный проблемно-ориентированным объектным программированием (ПООП). Новизна подхода заключается в том, что специалист в некоторой предметной области записывает решение задачи этой предметной области с помощью объектов различных типов, выполняющих действия, специфичные для данной предметной области и описанные с помощью единственной конструкции. Структура и описание этой конструкции также приведены в работе [1]. Описание объектов, значений их параметров и последовательности передачи управления между объектами составляют программу на одноименном языке ПООП. Областью применения ПООП являются пользовательские сценарии, определяющие порядок работы некоторых систем. В работе [1] был приведен сценарий проверки знаний правил образования количественных числительных естественных языков для разработанного Интернет-приложения для обработки количественных числительных [2].

Однако ПООП имеет более обширную область применения, чем только пользовательские сценарии.

Целью работы является показать новые области применения ПООП для описания порядка работы продукционной системы и системы вывода информации различного типа согласно сценарию. Достижение этой цели покажет значимость ПООП и возможность ее использования для описания принципа работы различных систем.

Предметно-ориентированный и проблемно-ориентированный подходы к решению задач. В любой предметной области можно выделить предметы (объекты) и проблемы (задачи). В зависимости от того, что рассматривается в первую очередь: предметы или проблемы, различают предметно-ориентированный и проблемно-ориентированный подходы к решению задач. При предметно-ориентированном подходе более глубоко исследуются предметы предметной области, а при проблемно-ориентированном подходе – проблемы.

Рассмотрим предметную область «Компания, предоставляющая услуги IP-телефонии». При решении задачи уведомления абонентов, имеющих задолженность по оплате услуг, используется предметно-ориентированный подход. Сначала необходимо сформировать базу данных (БД), содержащую данные об абонентах, предоставленных им услугах и датах поступления от них платежей. Формируется запрос на формирование списка должников, которые уведомляются в установленном порядке. При решении задачи установления соединения между абонентами используется проблемно-ориентированный подход. Для установления соединения необходимы только телефонные номера или другие иденти-

фикаторы абонентов, но не их фамилии, имена, отчества и другие данные из БД.

Проблемно-ориентированный и предметно-ориентированный подходы не являются конкурирующими. Они являются двумя различными взглядами на одну предметную область.

Точнее было бы называть предметные области проблемными, так как любая предметная область исследуется прежде всего для решения задач в ней. Предметы не нужны, если их нельзя использовать для решения задач. Однако термин «предметная область» является устоявшимся, поэтому мы будем использовать его, подразумевая под ним проблемную область.

В ПООП, как следует из его названия, используется проблемно-ориентированный подход: в первую очередь исследуются проблемы предметной области, а предметы исследуются поверхностно (в некоторых случаях нужно только знать, что они существуют в данной предметной области). Этим и ограничивается область применения ПООП.

Представление управляющих конструкций языка программирования единственной конструкцией ПООП. Чтобы показать связь ПООП с другими языками программирования, опишем управляющие конструкции (условия и циклы) языка программирования единственной конструкцией ПООП. В качестве исходного языка программирования будем использовать язык С.

Условный оператор обеспечивает выполнение оператора (группы операторов) в одной из двух ветвей в зависимости от истинности или ложности условия. Данная конструкция языка С имеет вид:

```
if (<условие>) {
  <операторы>;
}
else {
  <операторы>;
}
```

Условный оператор можно представить объектом типа ControlCondition, который имеет единственное поле

Condition – условие;

и две ветви:

iftrue (обозначается T) – выполнить команды, если условие истинно;

iffalse (F) – выполнить команды, если условие ложно.

Общий вид объекта типа ControlCondition:

```
C1 = ControlCondition {
  Condition: <условие>
  iftrue
    call <объект 1>;
  iffalse
    call <объект 2>;
}
```

Цикл с предусловием языка С выполняется пока истинно условие, указанное перед его началом:

```
while(<условие>) {
  <операторы>;
}
```

На языке ПООП цикл с предусловием можно представить с помощью объектов типа ControlCondition и типа RunBody, который выполняет тело цикла (первым выполняется объект C1):

```
C1 = ControlCondition {
  Condition: <условие>
  iftrue
    call B1;
  iffalse
    call E;
}

B1 = RunBody {
  ifnocondition
    call C1;
}
```

Здесь и далее объект E – это следующий выполняемый (после цикла) объект.

Цикл с постусловием языка С выполняется пока истинно условие, которое указано в конце цикла:

```
do {
    <операторы>;
}
while(<условие>)
```

На языке ПООП цикл с постусловием можно описать так (первым выполняется объект B2):

```
B2 = RunBody {
  ifnocondition
    call C2;
}

C2 = ControlCondition {
  Condition: <условие>
  iftrue
    call B2;
  iffalse
    call E;
}
```

С помощью языка ПООП можно также реализовать цикл с параметром for языка С. Заголовок цикла включает переменную (или переменные), которая изменяет своё значение от заданного начального до конечного значения, изменяясь заданной операцией, и для каждого значения этой переменной тело цикла выполняется один раз:

```
for (<инициализация>; <условие>; <операция>) {
    <операторы>;
}
```

На языке ПООП данную конструкцию можно представить следующим образом (сначала выполняется объект I1):

```
I1 = Initialize {
  Variable: i
  OriginValue: 0
  ifnocondition
    call C1;
}

C3 = ControlCondition {
  Condition: i < 10
  iftrue
    call B3;
  iffalse
    call E;
}

B3 = RunBody {
  ifnocondition
    call F1;
}

F1 = Operate {
  Variable: i
  ifnocondition
    call C1;
}
```

Последовательность вызовов объектов I1, C3, B3, F1 также соответствует циклу с предусловием:

```
I1;
while (C3) {
  B3;
  F1;
}
```

что говорит об эквивалентности этих управляющих конструкций.

Таким образом, перечисленные конструкции языка С были представлены единственной конструкцией ПООП четырех типов ControlCondition, RunBody, Initialize и Operate. Представление конструкций языка программирования конструкцией ПООП наглядно демонстрирует порядок работы условного оператора и операторов цикла языка программирования.

ПООП отличается от языков программирования еще одним аспектом. Порядок выполнения операторов в языках программирования определяется списком операторов – программой. В ПООП порядок выполнения указывается явно: в каждом объекте указывается, какому объекту передается управление, поэтому программа, как список операторов, не нужна.

Применение ПООП для представления и вывода знаний в продукционной модели. В работе [1] было высказано предположение, что объекты ПООП могут использоваться для представления знаний в продукционной модели.

Продукция вида ЕСЛИ-ТО представляет знание и соответствующее ему действие: проверить высказывание из условной части ЕСЛИ продукции, и если оно истинно, то считать высказывание из заключительной части ТО истинным. Следовательно, ПООП можно использовать для представления знаний в виде продукций.

В работе [3] приведены продукции (мы их обозначим R1, R2, ..., R10), используемые для принятия решения о финансировании проектов. Первая продукция (R1) имеет вид:

Если "конкурентоспособность проекта" = "очень высокая" И "актуальность и новизна проекта" = "очень высокая" И "социально-экономическая значимость проекта для города и области" = "очень высокая" И "финансовый уровень предприятия-заявителя" = "очень высокий", ТО "Принять проект к реализации".

Объект типа ControlCondition, описывающий приведенную выше продукцию, имеет вид:

```
R1 = ControlCondition {
  Condition: ("конкурентоспособность проекта" = "очень высокая") AND ("актуальность и новизна проекта" = "очень высокая") AND ("социально-экономическая значимость проекта для города и области" = "очень высокая") AND ("финансовый уровень предприятия-заявителя" = "очень высокий")
  iftrue
    call A;
  iffalse
    call R2;
}
```

Объект ApproveProject (обозначается A), обрабатывающий принятие проекта к реализации, может заносить данные проекта в БД проектов и/или рассылать соответствующие уведомления заинтересованным лицам. Также для примера из работы [3] будут определены объект ApproveBut (B), обрабатывающий принятие проект к реализации при наличии средств, объект DeferProject (D), обрабатывающий отложение проекта, объект RejectProject (J), обрабатывающий отклонение проекта, и объект ImpossibleToDecide (I), выполняющийся, если решение по проекту не может быть принято.

Инженер по знаниям описывает продукции с помощью ПООП (рисунок 1, на примере из работы [3]). Такое описание представляет собой алгоритм логического вывода новых знаний в продукционной модели. Подчеркнем, что новая модель представления знаний не предлагается. Всё изложение ведётся в терминах продукционной модели представления знаний.

Применение ПООП в прикладных задачах. Некоторые регионы России активно создают и развивают туристическую инфраструктуру, в том числе с использованием информационных технологий. В наиболее популярных туристиче-

ских местах устанавливаются городские туристические табло – экраны различных размеров и выполненных по различных технологиям. На табло выводится информация следующих типов:

- информация с веб-сервисов (прогноз погоды, последние новости);
- городская информация (объявления, анонсы культурных и спортивных событий, сообщения гражданской обороны);
- видео- и графические изображения (информация об истории города и достопримечательностях).

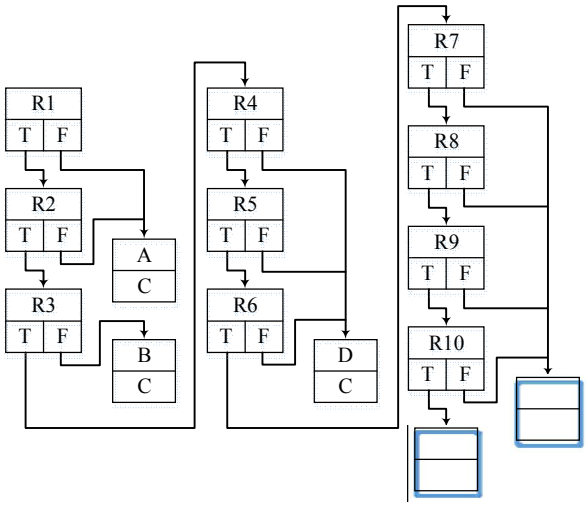


Рисунок 1 – Сценарий логического вывода

Информация на табло выводится согласно сценарию. Для написания такого сценария можно использовать ПООП.

Структура типов объектов, использующихся для решения задачи, проста. Например, объект M1 типа ShowMedia выводит на табло заданный медиафайл:

```
M1 = ShowMedia {
  MediaFile: D:\Video\CityViews.avi
  ifnocondition
    call E;
}
```

Все типы объектов для вывода информации на табло используют безусловную передачу управления ifnocondition (обозначается на схеме C).

Также в сценариях используется объект типа CheckTimer, который имеет поле

TimeLimit – временной интервал;

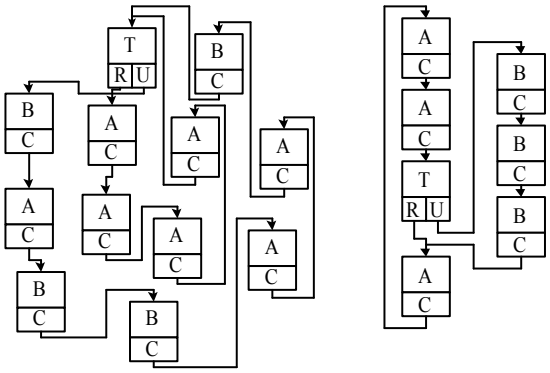
и условия:

iftimeisup (U) – выполнить команду, если временной интервал истёк;

iftimerunsout (R) – выполнить команду, если временной интервал не истёк.

Объект типа CheckTimer при первом выполнении запускает таймер. Если значение таймера не превысило временной интервал, выполняется команда из ветви iftimerunsout, иначе выполняются команды из ветви iftimeisup, а значение таймера обнуляется, и он запускается вновь.

Все типы объектов для вывода информации можно разделить на первостепенные (обозначаются A), которые необходимо выводить на табло чаще, и второстепенные (B). В сочетании с типом объекта CheckTimer можно создавать сценарии вывода информации различных конфигураций (рисунок 2, а, б).



а **б**

Рисунок 2 – Сценарии вывода информации

Принципы ПООП.

1. ПООП – это универсальный язык программирования, понятный непрограммистам.
2. ПООП никогда не станет сложнее.
3. ПООП не надо обучаться, с ним достаточно познакомиться.
4. Единственная забота пользователя при ПООП – составить последовательность действий для решения задачи.

Поясним перечисленные принципы.

Принцип 1. В отличие от известных языков программирования ПООП использует действия, понятные пользователю. Например, учитель немецкого языка поймет смысл действий «Задать Вопрос», «Выдать Подсказку», «Показать Результат» (см. пример в работе [1]), а инженер по знаниям без труда составит из объектов типа ControlCondition базу знаний. ПООП подстраивается под пользователя, а не наоборот. В случае с языками программирования пользователь должен изучать конструкции языка и принципы составления программ.

Принцип 2. Во время развития и эволюции многие языки программирования утратили первоначальные принципы, сформулированные при их создании. Например, язык Basic в современной версии уже никто не назовет языком для обучения программированию, каким он был первоначально, он стал сложнее. Мы, авторы ПООП, будем приветствовать все идеи развития ПООП, которые не противоречат данным прин-

ципам. Однако мы будем противодействовать любым попыткам усложнить ПООП и изменить его ориентацию с непрограммистов на программистов.

Принцип 3. Связан с принципами 1 и 2. Чтобы составить программу с помощью ПООП, необходимо познакомиться с типами объектов, их полями и условиями, а также способами связи между ними. ПООП говорит с пользователем на его языке.

Принцип 4. После анализа предметной области аналитики выделили типы объектов и способы передачи данных между ними (скрытые от пользователя), а программисты реализовали их в программном коде. Специалисту в этой предметной области остаётся только составить программу для решения своих задач.

Заключение. В статье представлены следующие результаты.

1. Показана связь между ПООП и языками программирования; управляющие конструкции языков программирования можно описать единственной конструкцией ПООП 4 типов (2 из которых используются только в операторе с параметром).

2. С помощью ПООП описан порядок работы различных систем:

- продукционной системы представления знаний;
- системы вывода информации различного типа.

ПООП может использоваться не только для описания пользовательских сценариев, но и для описания порядка работы более сложных систем.

3. Сформулированы принципы ПООП, определяющие направление его развития и эволюции.

Можно предположить, что наряду с ПООП существует предметно-ориентированное объектное программирование для непрограммистов, в котором данные первичны, а команды вторичны. Инструкции предметно-ориентированного объектного программирования аналогичны вызовам методов объектов объектно-ориентированного программирования, например:

```
data.process();
```

тогда как в языке ПООП типы объектов в общем случае имеют вид

```
process(data);
```

В дальнейшем планируется использовать ПООП для описания алгоритмов работы систем адаптивной ускоренной маршрутизации в динамических корпоративных сетях [4-5] и систем автоматизированного обучения и проверки знаний [6-8].

Библиографический список

1. *Пруцков А.В., Цыбулько Д.М.* Проблемно-ориентированное объектное программирование // Вестник Рязанского государственного радиотехнического университета. – 2013. – № 3 (45). – С. 57-62.

2. *Пруцков А.В., Цыбулько Д.М.* Интернет-приложение метода обработки количественных числительных естественных языков // Вестник Рязанского государственного радиотехнического университета. – 2012. – № 41. – С. 70-74.

3. *Гусева М.В., Демидова Л.А.* Классификация инвестиционных проектов на основе систем нечеткого вывода и мультимножеств // Вестник Рязанского государственного радиотехнического университета. – 2006. – № 19. – С. 157-165.

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

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

6. *Пруцков А.В.* Применение информационных ресурсов для автоматизации обучения и проверки знаний // Информационные ресурсы России. – 2005. – № 1. – С. 18-20.

7. *Пруцков А.В.* Статический и динамический подходы к проектированию подсистем проверки знаний автоматизированных обучающих систем // Информационные ресурсы России. – 2006. – № 1. – С. 27-29.

8. *Пруцков А.В., Розанов А.К.* Программное обеспечение методов обработки форм слов и числительных // Вестник Рязанского государственного радиотехнического университета. – 2011. – № 38. – С. 78-82.

УДК 004.75

Е.В. Селиванов, И.Ю. Каширин

ОБЛАЧНЫЕ ТЕХНОЛОГИИ КАК НОВАЯ СТУПЕНЬ ЭВОЛЮЦИИ ИНФОРМАЦИОННЫХ СЕРВИСОВ ГЛОБАЛЬНЫХ СЕТЕЙ

Рассмотрена история развития информационных сервисов глобальных сетей и проведён её анализ. Исследован современный рынок веб-приложений. Выделены основные типы облачных систем. Определено место облачных технологий в современном мире. Рассмотрены перспективные направления развития интернет-технологий.

Ключевые слова: *облачные технологии, интернет-сервисы, распределённые приложения.*

Введение. Информационные технологии в современном мире стремительно развиваются. Сетевые интернет-технологии сейчас являются безусловными лидерами по темпам своего роста. Специалисту в любой области науки нужно иметь представление об истории развития тех или иных сервисов, перспективах их дальнейшего существования и эволюции, постоянно находиться в курсе последних событий, если он хочет уследить за постоянно изменяющимся рынком веб-приложений и ориентироваться в нём. Современные облачные сервисы, хотя и появились относительно недавно, но уже стали доступны не только для узкого круга потребителей, но и для широких масс. Такие сервисы перестали требовать специальных знаний для своего использования, стали проще и гораздо распространеннее, всё больше заменяя собой обыкновенные приложения, подчас совершенно незаметно для пользователя.

История развития информационных сервисов от компонентной технологии до современных облачных инструментов представляет большой научный интерес, так как её анализ позволяет увидеть и понять основные тенденции в области веб-приложений, а также спрогнозировать основные направления развития этой отрасли в будущем. Кроме того, наглядный обзор эволюционного отбора технологий, рассмотрение удачных решений и провалов на пути развития этой области помогают понять на первый взгляд неочевидные вещи, предостерегают от «изобретения велосипеда» и указывают на разработки, которые будут востребованы в ближайшее время.

Идеи сервис-ориентированных архитектур, а затем и облачных технологий на удивление быстро овладели умами большого количества людей, хотя ещё не были в достаточной мере осознаны. В настоящее время облачные технологии, являясь новой ступенью эволюции информационных сервисов глобальных сетей, занимают всё больше пространства, заявляя свои права на те части рынка, которые были заняты стандартными «настольными» приложениями. Глобальная сеть развивается, этим существенно способствуя росту всевозможных видов распределённых приложений в частности и совершенствованию сервис-ориентированной архитектуры в целом. Несомненно, у этой области технологий большое будущее.

Цель настоящей работы – провести обзор облачных технологий как основной ступени эволюции информационных сервисов (ИС) глобальных сетей (ГС), провести анализ истории развития этой технологии, определить её место в современном мире и раскрыть перспективные направления развития веб-приложений.

Компонентные технологии. История развития ИС ГС берёт своё начало в 90-х годах на заре развития компонентных технологий, таких как COM (Component Object Model), CORBA (Object Request Broker Architecture) [10]. Эти технологии давали разработчикам функции для многократного применения двоичных объектов и программное обеспечение для совместного использования кода на одном компьютере. Осознание потенциала технологии повторного использования кода, а также развитие локальных и глобальных сетей в дальнейшем привели к усовершенствованию компонентных моделей для взаимодействия объектов между различными компьютерами. Таким образом, в середине 90-х годов появились расширенная компонентная модель от компании Microsoft – DCOM (Distributed COM), протокол IIOP (Inter-ORB

Protocol) CORBA [10] от компании OMG, а также Java RMI (Remote Method Invocation) от компании Sun. Эти стандарты позволяли программному обеспечению, работающему на одном компьютере, использовать код, расположенный на другом (удалённом) компьютере [10]. На рисунке 1 представлен общий вид архитектуры CORBA/IIOP.

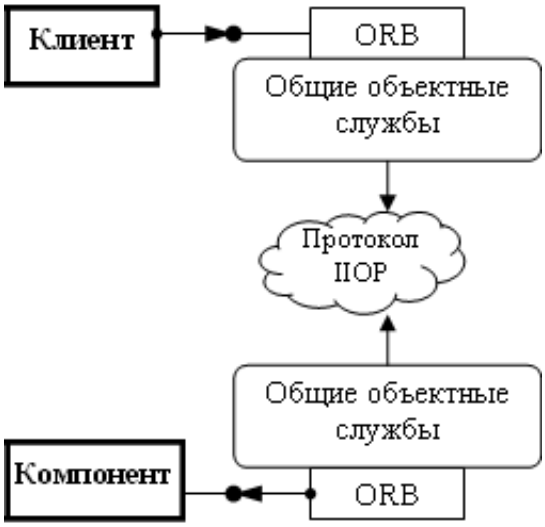


Рисунок 1 – Общий вид архитектуры CORBA/IIOP

Представленная архитектура по своему строению схожа с архитектурой COM/DCOM.

Однако перечисленные технологии должным образом не функционировали на пространстве сети интернет, оставаясь в большей части в рамках локальных вычислительных сетей. Во многом такое положение вещей было связано со следующими недостатками:

- зависимость от платформы и/или языка программирования, например COM-технология целиком ориентирована на платформу Windows, а RMI зависит от многих особенностей языка Java и не поддерживается в других известных языках программирования;

- ориентированность на передачу двоичных данных, так как компонентные технологии работают с бинарными объектами, обмен информацией между программами затрудняет даже наличие брандмауэров в сети;

- трудность масштабирования – технологии COM и CORBA позволяют спроектировать компоненты, способные обслуживать большое количество клиентов, но это требует от разработчика серьезного опыта и внимания при том, что типичный компонент не стоит этого;

- отсутствие универсального стандарта представления данных;

- сложность самих стандартов - COM и CORBA включают такие встроенные средства, как инструменты безопасности, шифрования, транзакции, что повышает вероятность возникновения проблем несовместимости.

Информационные сервисы. В качестве следующей ступени развития технологии в начале 2000-х годов выступают ИС. При разработке стандартов для них были учтены все самые слабые места предыдущих технологий и предложены новые способы решения накопившихся в этой области проблем.

Проблема совместимости была решена фундаментально с помощью перехода к текстовому формату представления данных: «Язык XML (Extensible Markup Language) - фундамент, на котором строятся веб-сервисы» [2]. XML взят за основу в ИС для передачи данных, описания веб-службы и типов данных. Таким образом, взаимодействие между различными сервисами более не являлось сложной задачей, что позволило добиться истинной кроссплатформенности веб-приложений.

На смену жёсткому бинарному представлению данных приходит гораздо более гибкий текстовой формат описания WSDL: «Язык описания веб-сервисов (Web Services Description Language, WSDL) [9] - это формат XML-схем, определяющий расширенную структуру описания интерфейсов веб-сервисов. WSDL первоначально был разработан компаниями Microsoft и IBM. Это общий способ представления передаваемых в сообщениях типов данных, указывающий действия, которые должны быть выполнены с данным сообщением, согласно которому сообщения привязываются к сетевым транспортам» [2]. Весь сервис полностью описывается с помощью текста, и это не только шаг к глобальной кроссплатформенности, но и компромисс в области человеко-машинного взаимодействия: язык XML может быть как понятен человеку, так и проинтерпретирован программными приложениями.

Использование текстового описания веб-служб способствует развитию протокола SOAP (Simple Object Access Protocol – простой протокол доступа к данным) [1], предоставляющего механизм подключения к сервисам. Его спецификация описывает структуру сообщений, используемых для обмена форматированными XML-данными в распределённой вычислительной среде, и является расширением протокола XML-RPC: первоначально SOAP использовался только для вызова удалённых процедур (RPC), но в связи с развитием веб-технологий стал использоваться также и для обмена XML-данными [1]. На основе SOAP могут возводиться более сложные протоколы и способы взаимодействия, так как он обеспечивает транспорт базового уровня.

В общем виде ИС для пользователя или потребителя выглядит как «чёрный ящик» и действует по такому же принципу: пользователь знает, что делает сервис, но не знает, как. На вход нужно подать исходные данные в известном заранее формате, а на выходе пользователь получает результат. Такое положение вещей оправданно при быстром росте технологий, когда уследить и угнаться за всем сразу невозможно, и в целях экономии своего времени переложить часть большой задачи на сторонний сервис – это нормальное и естественное решение при разработке программного обеспечения, такое же, как распределение труда в обществе. К тому же подобные технические решения помогают снять проблему повторного использования кода.

Однако каким образом сторонний пользователь узнает о существовании сервиса с нужным ему функционалом? Эту проблему почти до конца 2000-х годов решали специализированные реестры ИС – UDDI.

UDDI регистрирует и публикует определения веб-сервисов. Структура UDDI определяет модель данных в программных интерфейсах XML и SOAP для регистрации и обнаружения коммерческой информации, включая веб-сервисы [2]. Инициативу создания UDDI реестров поддержали большие компании, обладающие обширными ресурсами, такие как IBM, Microsoft, SAP, HP и другие. Ими были созданы публичные, общедоступные реестры, содержащие описание веб-сервисов, их адреса в сети, интерфейсы, WSDL-описания и всё, что нужно было для доступа к ним. Стандарт постепенно совершенствовался, базы данных UDDI росли, пополняясь новыми данными, и вопрос поиска ИС на тот момент казался решённым.

На рисунке 2 представлена общая схема взаимодействия веб-службы.

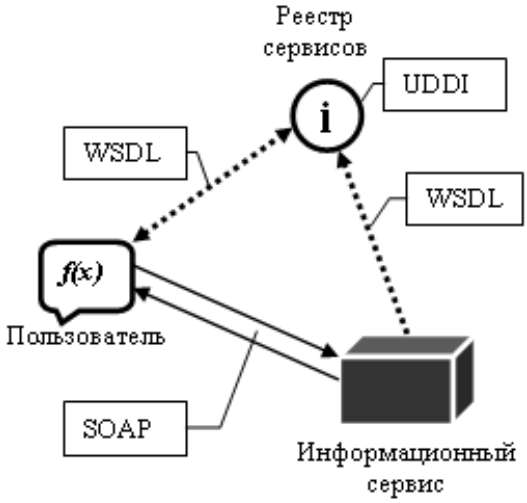


Рисунок 2 – Общая схема взаимодействия веб-службы

Согласно стандарту WSDL-описание сервиса размещается в реестре UDDI, чтобы пользователь мог найти нужную ему функцию и узнал о существовании веб-службы. По запросу реестр сервисов возвращает пользователю WSDL-описание сервиса, включающее его адрес, формат и типы данных для взаимодействия, список предоставляемых им функций. Далее пользователь напрямую обращается к ИС с помощью протокола SOAP, передавая ему исходные данные. Обработав данные, сервис возвращает пользователю результат.

Таким образом, потребитель может найти различные сервисы с нужными ему функциями и объединить их в нужной последовательности для решения своей задачи. На этом основывается понятие сервис-ориентированной архитектуры (SOA). Такая архитектура в том или ином виде активно используется в современных проектах. По такому же принципу построены и облачные технологии, однако в подавляющем большинстве случаев сервис-ориентированная архитектура представлена в них не в своём каноничном виде. Облачные технологии в этом смысле – новая ступень развития ИС ГС.

Работа с информационными сервисами. Одним из главных преимуществ сервис-ориентированного подхода к проектированию реальных систем является возможность использования совершенно разных языков программирования без ущерба для конструкции разрабатываемой системы, благодаря тому, что сервисы взаимодействуют на основе платформенно-независимого и языково-независимого интерфейса. Однако при выборе языка программирования необходимо учитывать не только уровень знания языка разработчиком, но и наличие библиотек/классов/средств, реализующих работу с веб-сервисами для данного языка программирования, их функциональность [8].

В таблице приводятся результаты проведённого исследования по поддержке работы с ИС в различных языках программирования. Наиболее популярные языки программирования оцениваются по наличию стандартных и сторонних библиотек/классов/средств, реализующих работу с общепринятым протоколом SOAP, а также возможности самостоятельной разработки таких средств и сложность доработки компонентов под конкретную задачу, стоящую перед программистом.

Наличие и отсутствие средств работы с ИС для каждого языка помечаются знаками «+» и «-» соответственно. Стандартная поддержка засчитывается, если в языке изначально присутствуют необходимые для разработки средства,

сторонняя – в том случае, если существуют библиотеки или классы, разработанные третьими лицами. Самостоятельная поддержка определяется принципиальной возможностью разработки своего средства взаимодействия с ИС. Сложность доработки компонентов имеет три градации – просто, средне и сложно. Этот критерий был оценен по наличию, полноте и степени локализации документации компонентов, предназначенных для коммуникации с ИС, а также по минимально необходимому для работы объёму кода.

Поддержка работы с ИС в различных языках программирования

Поддержка работы с ИС	Язык программирования			
	C++	C#	PHP	Java
Стандартная	-	+/-	+	+/-
Сторонняя	+	+	+	+
Самостоятельная	+	+	+	+
Доработка	сложн.	прост.	прост.	средн.

В каждом из рассмотренных языков программирования есть принципиальная возможность самостоятельного создания средств для работы с ИС. Это объясняется наличием во всех языках классов, реализующих передачу данных по сети.

Также для всех языков есть сторонние разработки, обеспечивающие SOAP-коммуникацию, что обусловлено широким распространением веб-сервисов и их востребованностью. Однако функциональность этих компонентов остаётся под вопросом и, планируя разработку с их использованием, необходимо это учитывать, а также определить сложность доработки кода под свои задачи.

Стандартная или изначальная поддержка работы с ИС посредством SOAP/WSDL в той или иной форме присутствует во всех рассматриваемых языках программирования, кроме C++. Однако C# и Java помечены знаком «+/-», потому как рабочие классы для них генерируются по WSDL-описанию сервиса с помощью среды разработки.

Для создания веб-сервисов и взаимодействия с ними в PHP существуют стандартные классы, реализующие полный необходимый функционал. В том числе предоставляется возможность парсинга, заранее неизвестного программе WSDL. Такая особенность открывает большие возможности для динамического связывания самых разных сервисов ГС. Язык PHP имеет большой объём документации на русском языке, прост, интуитивен и в итоге очень удобен для работы с ИС.

Столь богатое разнообразие технических средств и реализаций для сервис-ориентированной архитектуры не останавливает её развитие, напротив, в процессе работы выявляются проблемы и возникают новые решения, эта область постоянно улучшается и совершенствуется.

Эволюция SOA. При всех своих преимуществах каноничная архитектура также имеет свои недостатки: проблема повышения безопасности как отдельных компонентов сервис-ориентированных систем, так и самих этих систем (например, безопасного обмена данными); проблема поиска ИС; проблема разработки стандартизованного способа организации приложения на уровне модели бизнес-процесса и связанная с ней проблема методик анализа, разработки, мониторинга бизнес-процессов, использующих в своей работе ИС, а также другие проблемы [7].

К концу 2000-х годов один за другим UDDI реестры всех крупных компаний были закрыты и прекратили своё существование. Оставшиеся реестры, которые продолжают функционирование сейчас, работают нестабильно, многих сервисов, содержащихся в этих реестрах, уже не существует, а также в них почти не публикуются новые сервисы. Современные веб-сервисы стремятся к самодостаточности. Они публикуют всю мета-информацию о себе по своему адресу, не дублируя её в UDDI. Стандарт UDDI и UDDI сервисы используются только в очень узких областях. К примеру, они используются в BizTalk (Enterprise Service Bus Toolkit от Microsoft) [5].

В настоящее время всё более чёткой становится тенденция, когда многие крупные компании, такие как Google, не только не используют громоздкий UDDI, но также постепенно отходят от каноничного протокола SOAP и стандарта WSDL, разрабатывая собственные, более простые интерфейсы или API. «API — интерфейс прикладного программирования или интерфейс программирования приложений), т.е. набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением (библиотекой, сервисом) для использования во внешних программных продуктах. API определяет функциональность, которую предоставляет программа (модуль, библиотека), при этом API позволяет абстрагироваться от того, как именно эта функциональность реализована» [3]. Описание сервиса, его интерфейса и методы взаимодействия при этом публикуются в виде текста на ресурсе, посвящённом веб-службе. Также, идя в ногу со временем, некоторые компании переходят на AJAX взаимодействие с потребителем. Например, Google уже с 2007 года полностью отказывается от поискового WSDL-интерфейса «SOAP search API» в пользу «AJAX search API»,

в чём можно убедиться, посетив страницу для разработчиков «Google Developers».

Среди прочих инноваций в проектировании веб-приложений на сегодняшний момент можно выделить использование для передачи данных формата JSON вместо более громоздкого XML.

Формат JSON (англ. JavaScript Object Notation) также является текстовым, но в отличие от XML более лаконичен и вследствие этого существенно легче воспринимается людьми. Несмотря на происхождение от JavaScript, формат считается языконезависимым и может использоваться практически с любым языком программирования. К тому же для многих языков уже существует готовый код для создания и обработки данных в формате JSON. Применительно к веб-приложениям JSON хорошо зарекомендовал себя в задачах обмена данными как между браузером и сервером (AJAX), так и между самими серверами через программные HTTP-интерфейсы [6].

Облачные технологии. Развитие и совершенствование сервис-ориентированных архитектур в конце 2000-х приводит к возникновению качественно новой ступени эволюции ИС, которая получила наименование «облачные технологии».

«Облака» основываются на многих удачных решениях SOA, таких как универсальный доступ по сети, где услуги доступны пользователям по сети вне зависимости от используемого устройства доступа; слабосвязанные компоненты структуры; масштабируемость, когда услуги могут быть предоставлены, расширены, сужены по желанию без необходимости взаимодействия с поставщиком. При этом облачные технологии делают web-приложения дружественнее и понятнее для конечного пользователя. Это достигается с помощью новых технических решений, являющихся по своей сути надстройками над каноничной сервис-ориентированной архитектурой. Эти решения реализуют такие функции, как динамическое распределение мощностей и учёт их потребления, что является основными чертами облачной технологии.

Кроме того, в «облаке» интегрированы такие концепции, как:

- транзакционные вычисления, когда один или большее количество поступающих наборов данных обрабатываются совместно в рамках единой транзакции и устанавливают взаимосвязи с другими данными, уже введенными в систему;

- концепция Grid Computing, когда операции по обработке данных разбиваются на небольшие последовательности элементарных операций, которые могут выполняться изолированно;

- распределённые масштабируемые вычисления;

- распределённые хранилища данных;

- концепция виртуализации, позволяющая перемещать работающие экземпляры виртуальных серверов с одного физического сервера на другой, при этом с точки зрения пользователя или приложения, которые работают на виртуальном сервере, нет никаких возможностей определить, является ли сервер, на котором они работают, виртуальным или физическим [4];

- концепция Web 2.0, реализующая интерфейсы и взаимодействие между пользователями.

Механика сложных процессов скрыта от пользователя. Вместо множества технических подробностей «облако» обыкновенно предлагает потребителю удобный интерфейс, позволяющий решать именно те задачи, которые в данный момент нужно решить. Пользователь при этом контролирует лишь основные параметры услуги, имеющие для него реальное значение – такие, как скорость доступа, объём данных и т.п. Техническую сторону распределения данных полностью берёт на себя поставщик.

Ещё одним привлекательным для пользователя свойством является вычислительная эластичность, позволяющая гибко реагировать на изменения вычислительных и/или функциональных потребностей с минимальными эксплуатационными затратами и обращениями к провайдеру, а в некоторых случаях вообще без прямого участия человека. Это в свою очередь позволяет значительно уменьшить расходы на инфраструктуру информационных технологий, так как оплата услуги происходит только по факту пользования ею. К тому же «облачные провайдеры» предлагают комплексные услуги, уже готовые для включения в бизнес потребителя, что значительно снижает порог входа на рынок. Наиболее известными из этих услуг на данный момент являются: SaaS – программное обеспечение как услуга, PaaS – платформа как услуга и IaaS – инфраструктура как услуга.

Таким образом, облачные технологии привлекают к себе всё больше внимания и пользователи всё чаще делают выбор в пользу современных web-приложений. По оценке аналитической фирмы IDC (International Data Corporation), специализирующейся на исследованиях рынка информационных технологий, расходы на рынке облачных сервисов в 2009 году составили в целом примерно 16,5 миллиарда долларов или 4 % всего рынка информационных технологий [11].

На рисунке 3 представлена круговая диаграмма, иллюстрирующая процентное соотно-

шение расходов в сфере облачных технологий по состоянию за 2009 год.

Рисунок 3 – Диаграмма распределения расходов в сфере облачных технологий за 2009 год

Как можно видеть, облачные приложения доминируют в 2009 году, и это неудивительно, так как первая волна облачных сервисов состояла из приложений, предоставляемых в рамках модели SaaS (англ. Software as a Service), или «программное обеспечение как услуга». Необходимо отметить, что сервис-ориентированный подход наиболее прозрачен именно в этой модели. В ней используется одно из важнейших преимуществ сервисной архитектуры – удобство и простота технологии сервис-ориентированной интеграции. Модули, реализующие отдельные функции, являются автономными: из них можно собирать необходимую функциональную экосистему, комбинируя слабосвязанные сервисы в наборы, строить процессы высокого уровня, соответствующие конкретным требованиям бизнеса, и управлять ими посредством тонких клиентов или браузера.

Однако по прогнозам IDC уже в 2014 году доля облачных приложений на рынке уменьшится и модель PaaS (англ. Platform as a Service), или «платформа как услуга», станет популярнее, чем раньше. При этом расходы на рынке облачных технологий в целом вырастут до 55,5 миллиарда долларов при росте 27 % за год. Эта цифра составит 12 % от всего рынка информационных технологий [11].

На рисунке 4 представлена круговая диаграмма прогноза процентного соотношения расходов в сфере облачных технологий для 2014 года.

Такой прогноз говорит о том, что потребителям становится интересна возможность использования облачной инфраструктуры для размещения базового программного обеспечения с последующим размещением на нём приложений. Пользователи начинают больше доверять облачным технологиям, видят их преимущества и постепенно начинают отказываться не только от привычных «настольных» приложений, но и от базового программного обеспечения в пользу новых интернет-сервисов. Сфера облачных технологий постоянно совершенствуется и неуклонно растёт, охватывая новые области применения.

Рисунок 4 – Диаграмма прогноза расходов в сфере облачных технологий для 2014 года

Завершая обзор, нужно отметить, что столь обширный анализ современного рынка web-приложений в технической статье обоснован тем, что без знания основных тенденций мира информационных технологий эффективность работы специалиста падает, так как он не сориентирован относительно тех средств и решений, которые будут востребованы в ближайшем будущем.

Заключение. При сохранении темпов роста публичные «облака» будут доминировать в новой эре информационных технологий. Уже сегодня они достигают столь высокого уровня дружественности к конечному пользователю, что человек может не заметить, что работает «в облаке». Например, такими сервисами, как Gmail, Yahoomail, Webmail, Hotmail, огромное количество людей пользуются ежедневно, но для того чтобы продолжать работу, им нет никакой необходимости знать, что они работают с облачными технологиями. «Облака» незаметно охватывают всё большее пространство, заменяя собой традиционные информационные технологии. В этом легко убедится, взглянув на сервисы Google Docs, Zoho Writer, Google Picasa, Lunapic и др. Все они успешно выполняют функции своих «настольных» приложений-прообразов, предоставляя возможность работы с документами и изображениями посредством браузера, выступающего в роли тонкого клиента. Сфера развлечений, основывающаяся на облачных технологи-

ях, не нуждается в представлении: это множество различных музыкальных и видео-сервисов, игровые платформы, разнообразные творческие приложения.

Большие компании, такие как Google, Microsoft, Amazon и др., активно поддерживают и развивают идеи облачных приложений, создавая новые смелые решения в этой области. Примерами могут служить операционные системы Crome OS и Windows Azure, полностью существующие в «облаке», а также разнообразные инфраструктурные услуги от Amazon, первопроходца и лидера в сфере облачных технологий.

Проанализировав тенденции развития сферы web-сервисов, можно сделать вывод, что пользователи, активно пользуясь облачными приложениями, постепенно понимают все преимущества данной технологии и всё больше доверяют ей. Становятся востребованными облачные платформы PaaS, что в свою очередь может указывать на укрепление позиций облачной инфраструктуры (IaaS) в будущем.

ИС ГС прошли долгий путь от компонентных технологий до «облаков», и каждое событие на этом пути имеет своё обоснованное объяснение. Специалисту в области web-технологий необходимо знание истории этого пути, чтобы увидеть и понять основные тенденции сферы WWW, а также спрогнозировать основные направления развития этой отрасли в будущем.

Библиографический список

1. *Бекет Г.* Java SOAP. – Изд.: Лори, 2012. – 456 с.

2. *Ньюкомер Э.* Веб-сервисы: XML, WSDL, SOAP и UDDI – СПб.: Питер, 2003. – 256 с.

3. *Петин В. А.* API Яндекс, Google и других популярных веб-сервисов. Готовые решения для вашего сайта. – СПб.: БХВ-Петербург, 2012. – 480 с.

4. *Риз Дж.* Облачные вычисления - СПб.: БХВ-Петербург, 2011. – 288 с.

5. *Селиванов Е.В.* Реестры информационных сервисов глобальной сети Интернет // Новые информационные технологии в научных исследованиях: материалы XVII Всероссийской научно-технической конференции студентов, молодых ученых и специалистов. – Рязань: РГРТУ, 2012. – 250 с.

6. *Селиванов Е.В.* Технологии обмена данными информационных сервисов глобальных сетей // Математическое и программное обеспечение вычислительных систем: межвуз. сб. науч. тр. / под ред. А.Н. Пылькина. – Рязань: РГРТУ, 2012. – 240 с.

7. *Селиванов Е.В., Каширин И.Ю.* Тенденции развития сервис-ориентированных информационных систем // Современные тенденции в образовании и науке: сб. науч. тр. по материалам Международной научно-практической конференции 28 декабря 2012 г. – Тамбов: Изд-во ТРОО «Бизнес-Наука-Общество», 2013. – 163 с.

8. *Селиванов Е.В.* Поддержка работы с информационными сервисами в языках программирования // Математическое и программное обеспечение вычислительных систем: межвуз. сб. науч. тр. / под ред. А.Н. Пылькина. – Рязань: РГРТУ, 2013. – 152 с.

9. *Каширин И.Ю., Минашкин С.А.* Онтологии для представления знаний в интерактивных сервисах информационных сетей // Вестник Рязанского государственного радиотехнического университета. – 2012. – № 39-2. – С. 72-75.

10. *Эммерих В.* Конструирование распределённых объектов. Методы и средства программирования интероперабельных объектов в архитектурах OMG/CORBA, Microsoft/COM и Java/RMI. – М.: Мир, 2002. – 512 с.

11. *Frank Gens.* IDC's Public IT Cloud Services Forecast: New Numbers, Same Disruptive Story // IDC Exchange Blog. 2010. URL: http://blogs.idc.com/ie/?p=922. (Дата обращения: 23.10.2013).

УДК 62.681.5

О.В. Фалеев

СИНТЕЗ АГРЕГАТНОГО КОМПЛЕКСА ТЕХНИЧЕСКИХ И ПРОГРАММНЫХ СРЕДСТВ

Предложен метод решения задачи синтеза агрегатного комплекса технических и программных средств, учитывающий функциональную зависимость стоимости модулей комплекса от объема их производства.

Ключевые слова: *метод агрегатного построения, синтез агрегатного комплекса средств, задача разбиения множеств.*

Введение. В практике проектирования сложных систем [1, 2, 3] широко используется метод агрегатного построения [4, 5]. Процессам проектирования сложных агрегативных систем предшествует создание агрегатной базы, называемой также комплексом агрегатных средств

[6]. Однако в литературе не нашли должного отражения методы создания средств агрегатного комплекса.

При решении задачи создания агрегатного комплекса средств одним из основных условий является возможность его оптимизации по критерию минимума обобщенной стоимости. Из математической модели модуля [7] вытекает возможность сведения задачи синтеза агрегатного комплекса средств (АКС) к известной задаче разбиения множеств. Однако существующие методики решения этой задачи не учитывают функциональной зависимости стоимости модуля от объема его производства и не рассматривают возможность разбиения на ряд подмножеств с перекрываемыми функциями. *Цель работы* – формализованное решение задачи создания агрегатного комплекса технических и программных средств с учетом зависимости стоимости от объемов производства.

Теоретическая часть. Рассмотрим множество областей применения семейства систем автоматизации испытаний (САИ)

$$O = \left\{ O_i, \overline{1,I} \right\}, \quad (1)$$

в которых реализуется множество преобразований

$$\Pi = \left\{ \Pi^h, h = \overline{1,H} \right\} \quad (2)$$

при помощи множества процедур обработки

$$P = \left\{ p_j, j = \overline{1,J} \right\}. \quad (3)$$

Распределение процедур обработки между областями задается матрицей

$$A = \left\| a_{ij} \right\|, \quad (4)$$

где

$$a_{ij} = \begin{cases} 1, \text{ если } p_j \neq \emptyset \text{ в } O_i, \\ 0, \text{ в противном случае.} \end{cases} \quad (5)$$

Повторяемость процедуры обработки p_j на множестве областей $O = \{O_i\}$ характеризуется вектором−столбцом повторяемости

$$A_j = \left\| \begin{matrix} a_{1j} \\ \vdots \\ a_{ij} \\ \vdots \\ a_{Ij} \end{matrix} \right\|, \quad (6)$$

а количественное значение повторяемости процедуры p_j определяется суммой компонент вектора A_j

$$q_j = \sum_{i=1}^{I} a_{ij}. \quad (7)$$

Повторяемость q_j процедуры обработки p_j зависит от повторяемости преобразований Π^h на множестве областей $O = \{O_i\}$, которая характеризуется вектором

$$T = (t_1, ..., t_i, ..., t_I). \quad (8)$$

В этом случае повторяемость процедуры обработки p_j равна

$$q_j = \sum a_{ij} \cdot t_j. \quad (9)$$

Реализация процедуры обработки p_j требует затрат на разработку c_j, изготовление d_j и эксплуатацию v_j. Высокая повторяемость процедур обработки p_j позволяет использовать для ее реализации высокопроизводительные технологические процессы, что значительно уменьшает трудоемкость изготовления этих средств.

Формализация задачи синтеза. Синтезируемый агрегатный комплекс средств (АКС) $M(P) = \left\{ M_k, k = \overline{1,K} \right\}$ должен обладать свойствами функциональной полноты

$$\bigcup_k M_k = P, \; M_k \in M(P). \quad (10)$$

При этом каждый модуль M_k характеризуется множеством индексов J_k, образованным индексами процедур p_j, реализуемых модулем M_k:

$$J_k = \left\{ j : j \in J; p_j \in M_k \right\}. \quad (11)$$

Тогда

$$M_k = \bigcup_{j \in J_k} p_j, M_k \subset P. \quad (12)$$

Каждый модуль M_k характеризуется вектором

$$L_k = (L_{k1}, ..., L_{kj}, ..., L_{kJ}), \quad (13)$$

где

$$L_{kj} = \begin{cases} 1, \text{ если } p_j \in M_k, \\ 0, \text{если } p_j \notin M_k. \end{cases}. \quad (14)$$

Подмножество процедур p_j, реализуемых модулем M_k, образует подматрицу B_k матрицы A

$$B_k = \left\| a_{ij}^k \right\| = L_k \cdot \left\| a_{ij} \right\|, \quad (15)$$

где

$$a_{ij}^k = \begin{cases} 1, \text{ если } p_j \in M_k \text{ и } p_j \neq \emptyset \text{ в } O_i, \\ 0, \text{ в противном случае.} \end{cases} \quad (16)$$

При разбиении множества P на подмножества M_k повторяемость процедуры p_j определяется повторяемостью модуля M_k, являющегося, в свою очередь, объединением повторяемости каждой процедуры $p_j \in M_k$:

$$W_k = \sum_{i=1}^{I} a_{ij}, \; j \in J_k, \quad (17)$$

или с учетом повторяемости области O_i:

$$W_k = \sum_{i=1}^{I} a_{ij}^k \cdot t_i, \; j \in J_k. \quad (18)$$

Обобщенная стоимость модуля M_k с учетом его повторяемости W_k равна

$$C_k = \sum_{i=1}^{I} \sum_{j=1}^{J} a_{ij}^k \left\{ t_i \left[d_j(W_k) + v_j \right] + c_j \right\}, \; j \in J_k. \quad (19)$$

Обобщенная стоимость множества модулей $M(P) = \left\{ M_k, k = \overline{1,K} \right\}$ равна

$$C = \sum_{k=1}^{K} C_k = \sum_{i=1}^{I} \sum_{j=1}^{J} \sum_{k=1}^{K} a_{ij}^k \left\{ t_i \left[d_j(W_k) + v_j \right] + c_j \right\}. \quad (20)$$

Выражение (20) является целевой функцией задачи разбиения множества процедур $P = \left\{ p_j, j = \overline{1,J} \right\}$ на множество подмножеств, каждое из которых представляет собой модуль, $M(P) = \left\{ M_k, k = \overline{1,K} \right\}$. Каждое из подмножеств возможных разбиений $M_u(P)$ характеризуется вектором

$$\Theta^u = \left\{ L_k^u, u = \overline{1,U} \right\}, \Theta^u \subseteq \Theta. \quad (21)$$

Из всего множества векторов возможных вариантов разбиения выделим подмножество векторов допустимых вариантов Θ^0. Правило отбора векторов из множества Θ в подмножество векторов допустимых вариантов Θ^0 сужает количество вариантов. Это правило представляет собой ограничения на информационную и параметрическую совместимость различных вариантов разбиения и ограничения на объем модуля и число связей межмодульного интерфейса. Информационная совместимость внутримодульного интерфейса описывается следующим выражением:

$$\left\{ \left\{ \bigcup_{j \in J_k} \overrightarrow{X^j} \right\} \setminus \left\{ \bigcup_{l \in J_k} \overrightarrow{Y^l} \right\} \cup L_k \right\} = \emptyset, j \neq l, \quad (22)$$

где $\overrightarrow{X^j}$ – вектор входных переменных процедуры p_j, $\overrightarrow{Y^l}$ – вектор выходных переменных процедуры p_l, L_k – внешний интерфейс модуля:

$$L_k = \left\{ \overrightarrow{X^k} \cup \overrightarrow{Y^k} \right\}. \quad (23)$$

Здесь $\overrightarrow{X^k}$ и $\overrightarrow{Y^k}$ – векторы входных и выходных переменных модуля соответственно.

Пусть процедура p_j требует для своей реализации ψ_j элементов, а максимальное число элементов модуля M_k равно β_k. Тогда ограничение на число элементов модуля имеет вид

$$\sum_{j \in J_k} \psi_j \leq \beta_k. \quad (24)$$

Если каждый вариант реализации процедуры p_j характеризуется определенным значением параметра g_i по i–му входу и g_i^* по i^*–му выходу объекта, выполняющего эту процедуру, то параметрическая совместимость множества модулей, образующих АКС, оценивается следующим выражением:

$$g_i = g_i^*. \quad (25)$$

Выразим интерфейс L_k через соответствующие переменные процедур p_j, образующих модуль M_k. Для этого рассмотрим матрицу смежности множества процедур $P = \left\{ p_j \right\}$:

$$\varphi = \left\| \gamma_{ij} \right\|; \; j, j' \in J, \quad (26)$$

где

$$\gamma_{jj'} = \begin{cases} 1, \text{ если } f_j' \text{ выполняется после } f_j \\ 0 \text{ в противном случае.} \end{cases} \quad (27)$$

Множество входных переменных X^k модуля M_k образуется множеством входных переменных процедур $p_j \in M_k$, которое не содержится во множестве выходных и промежуточных переменных этих процедур:

$$X^k = \left\{ \bigcup_{j \in J_k} X^j \right\} \setminus \left\{ Y^j \bigcap_{j, j' \in J_k} X^{j'} \right\} \quad (28)$$

или

$$X^k = \left(1 - \gamma_{jj'}\right) \cdot \left\{ \cup X^j \right\}; \; j, j' \in J_k. \quad (29)$$

Аналогично получаем выражение для Y^k:

$$Y^k = \left(1-\gamma_{jj'}\right)\cdot\left\{\bigcup_{j'} Y^{j'}\right\};\ j\in J_k,\ j'\notin J_k. \quad (30)$$

В соответствии с (23) имеем

$$L_k=\left\{\left\{\cup X^j\left(1-\gamma_{jj'}\right)\right\}\cup\left\{Y^{j'}\left(1-\gamma_{j'l}\right)\right\}\right\};\ j,j'\in J_k,\ l\notin J_k. \quad (31)$$

Если $B_j^{вх}$ – число связей входной переменной X^j, $B_j^{вых}$ – число связей выходной переменной Y^j, а максимальное число внешних связей модуля M_k ограничено величиной ω_k, то ограничение на число внешних связей имеет вид

$$\sum_{\substack{j,j'\in J_k\\ l\in J_k}}\left[B_j^{вх}\left(1-\gamma_{jj'}\right)+B_j^{вых}\left(1-\gamma_{j'l}\right)\right]\le\omega_k. \quad (32)$$

Задача распределения множества процедур $P=\{p_j\}$ между множеством модулей $M(P)=\{M_k\}$ формулируется следующим образом:

Найти такой вектор Θ^u, характеризующий вариант разбиения множества процедур на множество подмножеств, каждое из которых представляет собой модуль, который бы минимизировал целевую функцию

$$C=min\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K}a_{ij}^k\left\{t_i\left[d_j\left(W_k\right)+v_j\right]+c_j\right\} \quad (33)$$

при ограничениях:

$$W_k=\sum_{i=1}^{I}a_{ij},\ j\in J_k, \quad (34)$$

$$J_k=\left\{j: j\in J;\ p_j\in M_k\right\},\ J_k\subset J, \quad (35)$$

$$\sum_{j\in J_k}\psi_j\le\beta_k, \quad (36)$$

$$\left\{\left\{\bigcup_{j\in J_k}\overrightarrow{X^j}\right\}\setminus\left\{\bigcup_{l\in J_k}\overrightarrow{Y^l}\right\}\cup L_k\right\}=\emptyset,\ j\ne l, \quad (37)$$

$$L_k=\left\{\left\{\cup X^j\left(1-\gamma_{jj'}\right)\right\}\cup\left\{Y^{j'}\left(1-\gamma_{j'l}\right)\right\}\right\};\quad j,j'\in J_k,\ l\notin J_k \quad (38)$$

$$\left\{\bigcup_{j\in J}p_j\right\}\setminus\left\{\bigcup_{k\in K}M_k\right\}=0, \quad (39)$$

$$\sum_{\substack{j,j'\in J_k\\ l\in J_k}}\left[B_j^{вх}\left(1-\gamma_{jj'}\right)+B_j^{вых}\left(1-\gamma_{j'l}\right)\right]\le\omega_k, \quad (40)$$

$$g_l=g_{l^*},\ l,l^*\in L_k. \quad (41)$$

Разбиение множества процедур $P=\{p_j\}$ на множество подмножеств, каждое из которых представляет собой модуль, может осуществляться либо с нулевым пересечением модулей (разные модули не могут содержать одинаковые процедуры обработки), либо с ненулевым пересечением модулей (разные модули могут иметь в своем составе одинаковые процедуры обработки).

В 1−м случае для всех элементов множества справедливо соотношение, описываемое квантором всеобщности

$$\forall M_k\left\{M_L\cap M_\beta=\emptyset;\ M_L,M_\beta\in M(P)\right\},\ L\ne\beta. \quad (42)$$

Тогда сформулированная выше задача сводится к задаче разбиения множества, которая может решаться как оптимизационная задача на графах и сетях.

Однако глобальный экстремум задачи в случае зависимости стоимости производства модулей от объема их производства следует искать для общего случая, когда допускается ненулевое пересечение модулей, т.е. для всех элементов множества $M(P)=\{M_k\}$ справедливо соотношение, описываемое при помощи квантора существования

$$\exists M_k\left\{M_L\cap M_\beta\ne\emptyset;\ M_L,M_\beta\in M(P)\right\};\ L\ne\beta. \quad (43)$$

Повторяемость процедуры в этом случае равна

$$q_j=\sum_{i=1}^{I}\sum_{j=1}^{J}a_{ij}^k\cdot t_i. \quad (44)$$

Задача разбиения множества процедур на множество подмножеств, каждое из которых представляет собой модуль, разбивается на решение двух последовательно выполняемых задач: комбинаторной задачи формирования множества возможных вариантов разбиения и задачи анализа этих вариантов в соответствии с целевой функцией оптимизации. Если множество процедур обработки включает в себя *n* элементов, то мощность множества подмножеств *n*-элементного множества равна 2^n. Число элементов в этих подмножествах изменяется от 0 до *n*. Среднее число элементов в подмножествах составляет *n*/2. В этом случае размерность задачи равна $n\cdot2^{n-1}$. Так как для вычисления целевой функции (20) одного варианта необходимо выполнить 4 арифметические операции, которые повторяются $\left(n^2\cdot2^n\right)$ раз, то сложность задачи составляет $O\left(n^3\cdot2^{2(n+1)}\right)$.

Снижение размерности задачи. Одним из путей снижения размерности задачи, сформулированной выше, является сужение области изменения повторяемости W_k модуля M_k. Так как затраты на изготовление d_j являются нелинейной выпуклой функцией повторяемости q_j процедуры p_j, то зависимость их от q_j определя-

ется конкретными условиями ее реализации. Нелинейность и выпуклость этой зависимости не позволяют использовать существующие методы решения задач нелинейного программирования [8]. В то же время зависимость затрат на изготовление d_j от повторяемости q_j носит эмпирический характер, поэтому для получения аналитической зависимости обобщенной стоимости от повторяемости предлагается использовать метод кусочно-линейной аппроксимации переменной q_j.

Весь интервал определения переменной q_j в зависимости от точности аппроксимации разбивается на N линейных участков. Рассмотрим интервал $\left[q_j^n, q_j^{n+1}\right]$, на котором d_j изменяется линейно от d_j^n до d_j^{n+1}. Уравнение отрезка прямой, проходящей через две данные точки [9], на этом интервале имеет вид

$$\frac{d_j - d_j^n}{d_j^{n+1} - d_j^n} = \frac{q_j - q_j^n}{q_j^{n+1} - q_j^n}. \quad (45)$$

После преобразования получаем

$$d_j = \frac{d_j^{n+1} - d_j^n}{q_j^{n+1} - q_j^n} q_j + \frac{d_j^n \cdot q_j^n - d_j^{n+1} \cdot q_j^n}{q_j^{n+1} - q_j^n} \quad (46)$$

или

$$d_j = \eta_j \cdot q_j + \theta_j^n \quad . \quad (47)$$

Здесь $n = \overline{1, N}$ – индекс участка аппроксимации, а

$$\begin{cases} \theta_j^n = \dfrac{d_j^n \cdot q_j^n - d_j^{n+1} \cdot q_j^n}{q_j^{n+1} - q_j^n} \\ \eta_j = \dfrac{d_j^{n+1} - d_j^n}{q_j^{n+1} - q_j^n}. \end{cases} \quad (48)$$

Зависимость d_j от q_j аппроксимируется отрезками, уравнения которых имеют вид

$$d_j = \begin{cases} d_j^1, \text{ если } 0 < q \le q_j^1, \\ \eta_j^n \cdot q_j^n, \text{ если } q_j^n < q_j \le q_j^{n+1}, . \\ d_j^N, \text{ если } q_j > q_j^N. \end{cases} \quad (49)$$

Обобщенная стоимость реализации процедуры обработки p_j равна

$$\xi_j = c_j + d\left(q_j\right) + v_j, \quad (50)$$

где $d\left(q_j\right)$ определяется по формуле (49).

С учетом аппроксимации стоимость реализации средств, выполняющих процедуру p_j, равна

$$Д_j = \eta_j \left(q_j\right)^2 + \theta_j^n \cdot q_j. \quad (51)$$

Если d_j представляет собой убывающую функцию переменной q_j, то кривая, описываемая формулой (51), представляет собой параболу, обращенную вогнутостью вниз. Вершина параболы лежит в точке A с координатами

$$\begin{cases} q_j^{an} = -\dfrac{\theta_j^n}{2\eta_j^n}, \\ Д_j^{an} = -\dfrac{\left(\theta_j^n\right)^2}{4\eta_j^n}. \end{cases} \quad (52)$$

С точки зрения критерия минимума обобщенной стоимости представляет интерес область изменения переменной q_j, в которой $Д_j$ убывает при увеличении q_j. Ниспадающий участок кривой лежит правее оси параболы, проходящей через точку, в которой $q_j = -\dfrac{\theta_j^n}{2\eta_j^n}$, до пересечения с осью q_j, описываемой уравнением $q_j = 0$. Отсюда правая граница интервала изменения q_j, в котором $Д_j$ убывает, лежит в точке

$$R_j^{bn} = -\frac{\theta_j^n}{4\eta_j^n}. \quad (53)$$

Если область $\left[q_j^{an}, q_j^{bn}\right]$ пересекается с областью $\left[q_j^n, q_j^{n+1}\right]$, то область пересечения принадлежит области допустимого увеличения повторяемости q_j, в противном случае – не принадлежит:

$$q_{j\,don} = \bigcup_{n=1}^{N-1} \left\{ \left[q_j^{an}, q_j^{bn}\right] \cap \left[q_j^n, q_j^{n+1}\right] \right\}. \quad (54)$$

Это выражение является дополнительным ограничением задачи, сформулированной выше. С точки зрения увеличения повторяемости при объединении процедур p_j в модули M_k представляет интерес суммарная повторяемость несовпадения процедур, так как объединение процедур с максимальной повторяемостью несовпадения дает максимальную повторяемость модуля. Эта характеристика процедур описывается матрицей

$$L = \left\| l_{\alpha\beta} \right\|; \alpha, \beta = \overline{1, J}, \quad (55)$$

где каждый элемент $l_{\alpha\beta}$ равен

$$l_{\alpha\beta} = t_j \cdot \sum_{i=1}^{I} d_{i\alpha} \oplus d_{i\beta} \,. \qquad (56)$$

Здесь $\oplus$ – знак суммирования по $\mathrm{mod}\,2$.

Наибольший выигрыш в стоимости реализации даст объединение процедуры p_α с процедурой p_β, если $\left(l_{\alpha\beta} + q_\alpha + q_\beta\right)$ близко к $\max\limits_n q_{j\,доп}$ или

$$\left(\max_n q_{j\,доп} - l_{\alpha\beta} - q_\alpha - q_\beta\right) \to \min . \qquad (57)$$

В случае если разные процедуры p_j имеют различные допустимые области $q_{j\,доп}$, то условие максимального выигрыша от объединения процедур принимает вид

$$\left[\min_{j\in J_k} \max_n q_{j\,доп} - \sum_{i=1}^{I} t_i \cdot \left(d_{i\alpha} \oplus d_{i\beta}\right) - q_\varepsilon - q_\beta\right] \to \min \cdot \qquad (58)$$

При объединении нескольких (>2) процедур с одинаковой стоимостью реализации условие (58) преобразуется в следующее выражение:

$$\left[\min_{j\in J_k} \max q_{j\,доп} - \sum_{\substack{i=1\\ n}}^{I} t_j \cdot \left(\bigcup_{j\in J_k} d_{ij}\right)\right] \to \min . \qquad (59)$$

Увеличение повторяемости процедур за счет объединения последних дает эффект только тогда, когда выигрыш за счет уменьшения стоимости при увеличении объема реализации перекрывает дополнительные затраты на эксплуатацию. Пусть суммарные затраты на производство и эксплуатацию процедуры p_j равны

$$\rho_j = \eta_j^n \cdot (q_j)^2 + \theta_j^n \cdot q_j + v_j \cdot q_j = \eta_j^n \cdot (q_j)^2 + \left(\theta_j^n + v_j\right) q_j \,. \qquad (60)$$

Функция ρ_j убывает в той области аргумента q_j, где первая производная отрицательна:

$$\frac{d\rho_j}{dq_j} = 2\eta_j^n \cdot q_j + \left(\theta_j^n + v_j\right) < 0 \,. \qquad (61)$$

Отсюда видно, что ρ_j убывает, если

$$q_j < -\frac{\theta_j^n + v_j}{2\eta_j^n} \,. \qquad (62)$$

Так как $\rho_j = 0$ при $q_j = -\dfrac{\theta_j^n + v_j}{2\eta_j^n}$, то область убывания суммарных затрат на реализацию и эксплуатацию лежит в пределах изменения от $\left[-\dfrac{\theta_j^n + v_j}{2\eta_j^n}\right]$ до $\left[-\dfrac{\theta_j^n + v_j}{4\eta_j^n}\right]$. Область допустимого увеличения повторяемости равна

$$q_{j\,доп} = \bigcup_n \left\{\left[\left(-\frac{\theta_j^n + v_j}{2\eta_j^n}\right), \left(-\frac{\theta_j^n + v_j}{4\eta_j^n}\right) \cap \left[q_j^n, q_j^{n+1}\right]\right]\right\} . \qquad (63)$$

Пример. Рассмотрим использование предложенной выше методики на примере выбора набора конструктивных модулей, проектируемых на базе функционального набора. В результате анализа предметной области автоматизации испытаний сложных объектов, в частности изделий РКТ, выявлены десять областей применения для систем автоматизации, создаваемых на основе агрегатного комплекса. Распределение выявленных функций, их частота повторения для каждой из областей, а также стоимость изготовления при единичном производстве приведены в таблице 1. По данным этой таблицы определены необходимые повторяемости изготовления этих модулей.

Таблица 1 – Распределение функций между областями

	o_1	o_2	o_3	o_4	o_5	o_6	o_7	o_8	o_9	o_{10}	d_j
p_1	1	1	1	1	1	5	4	6	7	9	4
p_2	1	0	1	0	1	0	1	0	0	0	32
p_3	0	1	0	1	1	2	1	3	3	1	10
p_4	1	0	1	1	1	0	0	0	1	1	1
p_5	1	1	1	1	1	1	1	0	1	0	20
p_6	1	1	0	0	1	1	0	1	1	1	4
p_7	1	1	1	1	1	1	1	1	1	1	4
p_8	1	1	1	1	1	1	1	1	1	1	3
p_9	0	0	0	0	1	1	1	1	1	1	3
p_{10}	0	1	0	1	1	3	2	5	1	1	2
p_{11}	1	0	1	1	1	2	0	2	1	0	1
p_{12}	1	1	1	1	1	1	1	0	1	0	12
p_{13}	1	1	1	1	1	1	1	0	1	0	10
p_{14}	0	1	0	1	1	2	1	2	3	1	4
p_{15}	0	0	0	0	1	1	0	1	1	1	1
p_{16}	0	1	0	1	1	1	0	0	0	0	1
p_{17}	0	0	0	1	1	1	0	0	0	0	1
p_{18}	1	1	1	1	1	5	4	6	7	9	1
p_{19}	1	1	1	1	1	5	4	5	7	9	1
p_{20}	0	0	0	0	0	1	1	1	1	1	40
p_{21}	0	0	0	0	0	0	0	0	0	1	10
t_i	10	7	15	8	12	6	4	5	6	2	-

Зависимость стоимости изготовления от объема производства в ОКБ «Спектр», г.Рязань (место работы автора), с учетом ее кусочно–линейной аппроксимации с погрешностью не более 1 % приведена в таблице 2. Все стоимости приведены к стоимости изготовления одного модуля.

По формулам (53) и (62) найдем пересечение множества участков с убыванием стоимости при увеличении повторяемости функции с множеством участков изменения повторяемости с учетом стоимости эксплуатации.

$$q_{j\,доп}^1 = [152;304] \cap [0;17] = \varnothing$$

$$q^{2}_{j\,\text{доп}} = [80;160] \cap [17;34] = \varnothing$$
$$q^{3}_{j\,\text{доп}} = [70;140] \cap [34;51] = \varnothing$$
$$q^{4}_{j\,\text{доп}} = [93;186] \cap [51;68] = \varnothing$$
$$q^{5}_{j\,\text{доп}} = [54;108] \cap [68;85] = [68;85]$$
$$q^{6}_{j\,\text{доп}} = [69;138] \cap [85;102] = [85;102]$$
$$q^{7}_{j\,\text{доп}} = [63;127] \cap [102;136] = [102;127]$$
$$q^{8}_{j\,\text{доп}} = [44;89] \cap [136;153] = \varnothing$$
$$q^{9}_{j\,\text{доп}} = [88;176] \cap [153;187] = [153;176]$$
$$q^{10}_{j\,\text{доп}} = [283;566] \cap [187;194] = \varnothing$$

Таблица 2 – Зависимость стоимости изготовления от объема производства

№ п/п	q_j	d_j
1	$0 \le q_1 < 17$	$d_1 = -18 \cdot 10^{-4} q_1 + 1,0$
2	$17 \le q_2 < 34$	$d_2 = -35 \cdot 10^{-4} q_2 + 1,03$
3	$34 \le q_3 < 51$	$d_3 = -41 \cdot 10^{-4} q_3 + 1,05$
4	$51 \le q_4 < 68$	$d_4 = -29 \cdot 10^{-4} q_4 + 0,99$
5	$68 \le q_5 < 85$	$d_5 = -59 \cdot 10^{-4} q_5 + 1,19$
6	$85 \le q_6 < 102$	$d_6 = -41 \cdot 10^{-4} q_6 + 1,04$
7	$102 \le q_7 < 136$	$d_7 = -47 \cdot 10^{-4} q_7 + 1,10$
8	$136 \le q_8 < 153$	$d_8 = -41 \cdot 10^{-4} q_8 + 0,63$
9	$153 \le q_9 < 187$	$d_9 = -24 \cdot 10^{-4} q_9 + 0,75$
10	$187 \le q_{10} < 194$	$d_{10} = -6 \cdot 10^{-4} q_{10} + 0,58$
11	$194 \le q_{11}$	$d_{11} = 0,30$

Таким образом, область допустимого увеличения повторяемости модулей представляет собой объединение двух областей: $[68;127] \cup [153;176]$. В основе принципа объединения в модули лежит выражение (59), которое требует достижения максимальной повторяемости модуля, принадлежащей области допустимого увеличения повторяемости, которая равна max q_j.=176.

Задача разбиения на конструктивные модули относится к комбинаторным задачам. Ее решение дало следующие результаты (указаны индексы процедур):

M1={1,18,19} с повторяемостью 188 (выше допустимой, но минимально требуемая);

M2={2} с повторяемостью 41 (ограничение на количество элементов в модуле);

M3={20} с повторяемостью 78;

M4={4,5,8,15} с повторяемостью 68;

M5={9,11,12,16,21} с повторяемостью 94;

M6={6,7,13,14,15,17} с повторяемостью 92;

M7={3,5,10} с повторяемостью 123;

M8={6,7,8,12,16,17} с повторяемостью 75;

M9={4,9,13,14,21} с повторяемостью 75.

Общая стоимость изготовления агрегатного комплекса составила 29,63 условных единиц, приведенных к стоимости изготовления одного типового элемента замены, при этом выигрыш составил 40, 84 условных единиц.

Заключение. В статье предложен метод синтеза агрегатного комплекса средств, учитывающий функциональную зависимость стоимости модуля от объема его производства.

Библиографический список

1. *Корячко В.П., Скворцов С.В., Таганов А.И., Шибанов А.П.* Эволюция автоматизированного проектирования электронно − вычислительных средств // Радиотехника. − 2012. − №3. − С.97−103.

2. *Лазутин Ю.Д., Корячко В.П., Сускин В.В.* Технология электронных средств. − М.: МГТУ им.Баумана. − 2013. −286 с.

3. *Корячко В.П., Курейчик В.М., Норенков И.П.* Теоретические основы САПР. − М.: Энергоатомиздат. − 1987. −400 с.

4. *Бусленко Н.П., Калашников В.В., Коваленко И.Н.* Лекции по теории сложных систем. – М.: Сов. радио, 1973. −439 с.

5. *Гоев Н.В.* Синтез автоматизированных систем контроля на основе аппаратно − программных модулей // Вестник Рязанского государственного радиотехнического университета. − 2010. − №32. – С.98-102.

6. *Лазарев И.А.* Композиционное проектирование сложных систем. – М.: Радио и связь, 1986. − 312 с.

7. *Фалеев О.В.* Математическая модель модуля агрегатного комплекса технических и программных средств. // Вестник Рязанского государственного радиотехнического университета. − 2013. − №3 (45). – С.79−82.

8. *Мухачева Э.А., Рубинштейн Г.Ш.* Математическое программирование. – Новосибирск: Наука, Сиб. отд. – 1977. – 320 с.

9. *Корн Г. и Корн Т.* Справочник по математике для научных работников и инженеров. – М.: Наука – 1970. – 720 с.