ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ
Северо-Кавказский филиал ордена Трудового Красного Знамени
федерального государственного бюджетного образовательного учреждения
высшего образования
«Московский технический университет связи и информатики»

Методические указания
по выполнению практических работ и по дисциплине
«Архитектура информационных систем»

Ростов-на-Дону
2019

Рыбалко И.П., Гладыщук С.В. Методические указания по выполнению практических работ и по дисциплине: «Архитектура информационных систем» – Ростов-на-Дону: СКФ МТУСИ, 2019. – 43 с.

В книге даны рекомендации по выполнению практических работ по дисциплине: «Архитектура информационных систем».

Для каждого занятия приведены краткие теоретические сведения, описание методики расчёта, порядок выбора и выполнения индивидуального задания, определены требования для оформления отчёта и её защите.

Методические указания соответствуют направлению подготовки 09.03.01 Информатика и вычислительная техника, профили: Вычислительные машины, комплексы, системы и сети; Программное обеспечение и интеллектуальные системы.

Рецензент: Заведующий кафедрой ИВТ д.т.н. профессор Соколов С.В.

Составители: Рыбалко И.П., доц. каф. «ИТСС», Гладыщук С.В.
Набор и вёрстка: Гладыщук С.В.

Рассмотрены и одобрены на заседании кафедры «ИТСС». Протокол № 1 от «26» августа 2019 г.

СОДЕРЖАНИЕ

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №1

Ярусно-параллельные формы вычислительного процесса.

1. ЦЕЛЬ ЗАНЯТИЯ: практическое закрепление знаний о преобразовании схем алгоритмов к ЯПФ с целью их распараллеливания.

2. ОСНОВНЫЕ СВЕДЕНИЯ.

Отличительной особенностью многопроцессорных и многомашинных ВС является возможность параллельного выполнения независимых ветвей программы. Это свойство системы связано, с одной стороны, с наличием нескольких независимых вычислителей, а с другой - со свойствами самих алгоритмов, допускающих во многих случаях выделения параллельных ветвей.

Существует много различных способов отражения структуры вычислительного процесса, определяемого заданной программой для вычислительной системы. В частности, можно отметить различного рода блок-схемы, динамические диаграммы, граф-схемы и т.п. Для вычислительных систем, в которых могут быть организованы параллельные процессы, наиболее удобным оказывается использование граф-схем.

Всякую программу можно представить как некоторую последовательность работ (операторов), в результате чего получаем функциональный граф.

Для того чтобы связать функциональную граф-схему с реализацией ее в вычислительной системе, необходимо сопоставить каждой вершине функциональной граф-схемы некоторый вес, выражающий в условных единицах время выполнения данной работы с помощью вычислительных средств (процессора, ЭВМ) определенной производительности. Этот вес будем писать рядом с вершиной функциональной граф-схемы, которая теперь будет называется временной функциональной граф-схемой.

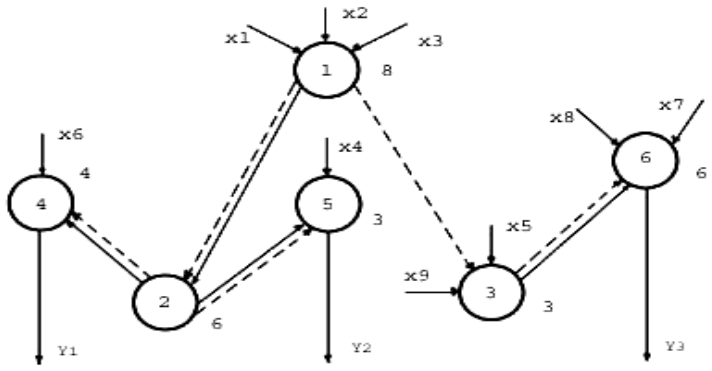


Рисунок 1 - Пример временной функциональной граф-схемы

На рисунке 1 показан пример временной функциональной граф-схемы. Для отображения разветвлений в программах вводятся связи по управлению: сплошные дуги соответствуют функциональным связям между вершинами графа, а пунктирные - связям по управлению. Величины X_i, i=1,2,...,9 являются исходными данными, а величины Y_i, i=1,2,3 есть результаты, которые должны быть

получены в процессе реализации программы.

Как показывают связи по управлению, в качестве результата работы ВС может получиться либо Y_1, либо Y_2, либо Y_3. Число типов процессоров равно двум.

Пусть имеется две работы, которые являются непосредственно функционально связанными или связаны по управлению. Ту работу, в которой в качестве идентификатора операнда используется идентификатор результата другой работы, будем называть зависимой. Если некоторая работа G_j зависит от G_Y, то это означает, что при реализации программы работа G_Y должна быть выполнена раньше работы G_j.

Упорядочим, теперь вершины графа программы. Множеством первого яруса или просто первым ярусом называются те работы, которые не зависят ни от каких других работ. Множеством второго яруса или вторым ярусом называют те работы, которые зависят, по крайней мере, от одной работы, входящей в первый ярус, и не зависят ни от какой другой работы. И вообще, множеством f-го яруса или f-м ярусом называются те работы, которые зависят, по крайней мере, от одной работы (f-1)-го яруса и не зависят ни от одной работы, других ярусов с номерами, меньшими (f-1). Граф, соответствующий такому упорядочению работ, называется ярусно-параллельным графом, а относительно программы, сопоставляемой этому графу, говорят, что она представлена в ярусно-параллельной форме (ЯПФ).

На рисунке 2 показана ярусно-параллельная форма программы, представленной на рисунке 1. В дальнейшем формально будем рассматривать ЯПФ, содержащие только функциональные связи, так как это по существу не влияет на методику решения рассматриваемых ниже задач распределения загрузки.

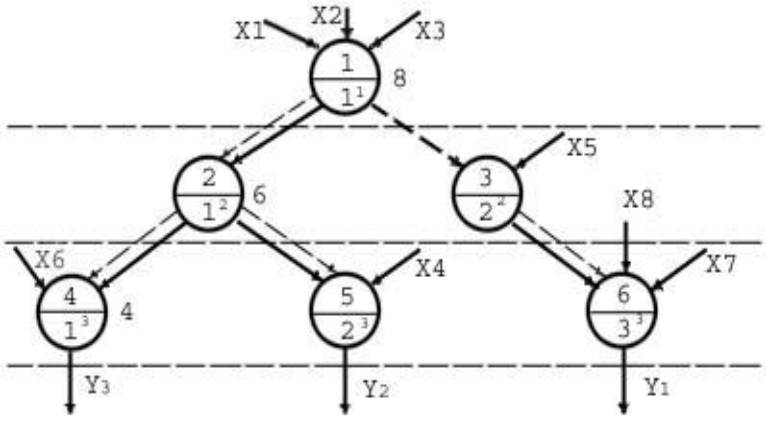


Рисунок 2 - Ярусно-параллельная форма программы

Для оценки структур ЯПФ вводятся специальные показатели. Эти показатели служат для классификации ЯПФ по их структурным особенностям, для оценки согласованности структуры решаемой задачи со структурой самой многопроцессорной вычислительной системы. С помощью показателей удобно сравнивать и оценивать структуры ЯПФ, соответствующие различным вариантам организации решения одной и той же задачи. Особый интерес представляет использование числовых характеристик на этапе исследования способности задачи к рас-

параллеливанию и выбора алгоритма решения, ЯЛФ программы которого характеризуются наилучшими показателями.

Шириной f-го яруса в графе G ЯПФ программы называется число работ b^f, входящих в этот ярус.

Шириной B графа G ЯПФ называется максимальная ширина яруса в G.

Длиной (трудоемкостью) l^f_{im} i-ой работы в f-м ярусе ЯПФ относительно процессора (ЭВМ) m-го типа называется время ее реализации на процессоре (ЭВМ) этого типа.

Длиной яруса l^f в графе G ЯПФ называется максимальная длина работы, входящей в этот ярус.

Длиной Ткр графа G ЯПФ называется величина наиболее длинного (критического) пути, ведущего к заключительному состоянию этого графа. Длина G определяет минимально-возможное время реализации данного графа.

Дебалансом яруса d^f называется разность между длиной яруса и длиной самой короткой работы, входящей в данный ярус.

Дебалансом графа d ЯЛФ называется сумма всех дебалансов ярусов данного графа.

Заполнением E^f яруса f в графе G называется сумма

$$E^f = \sum_{i=1}^{b^f} (l^f - l^f_{im}), \qquad 1$$

где b^f - число работ в f-м ярусе; l^f_{im} - длина i-й работы находящейся в f-м ярусе для m-го процессора.

Заполнением E графа G ЯПФ называется сумма

$$E = \sum_{f=1}^{N} \sum_{i=1}^{b^f} (l^f - l^f_{im}), \qquad 2$$

где N - число ярусов в G.

Разбросом Q графа G ЯПФ программы называется сумма

$$Q = \sum_{f=1}^{N} (E - b^f) \qquad 3$$

Введенные показатели позволяют произвести классификацию ЯПФ. ЯПФ называется равномерной, если в графе G этой ЯПФ дебалансы всех ярусов равны нулю.

ЯПФ называется абсолютно равномерной, если она равномерна и длины всех ярусов одинаковы.

ЯПФ называется прямоугольной, если разброс в графе этой ЯПФ равен нулю.

ЯПФ называется абсолютно совершенной, если она абсолютно равномерна и прямоугольна.

Пример построения ЯПФ алгоритма и расчет ее характеристик.

Граф программы задачи, подлежащей решению с помощью многопроцессорной системы, задан в табличной форме (таблица 1). В таблице 2 указаны длины работ, которые заданы в условных единицах для случая обслуживания процессором с относительной производительностью S_0 =1.

Таблица 1 – Граф задач многопроцессорной системы

		Выходы работ														
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Входы работ	1															
	2	1			1								1			
	3		1													1
	4															
	5	1														
	6					1			1							
	7															
	8												1			
	9					1		1				1				
	10								1						1	1
	11							1						1		
	12															
	13															
	14													1		
	15													1		

Таблица 2 – Длинна работ

N	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	10	5	7	15	9	8	20	14	11	6	3	18	12	8	11

Необходимо выполнить:

- преобразовать граф в ярусно-параллельную форму;
- рассчитать основные показатели ЯПФ.

1. Преобразование графа в ЯПФ. Пользуясь описанной методикой, представим исходный граф в ярусно-параллельной форме. Для этого выбираем независимые работы. Это работы 1,4,7,12,13. Помещаем их в первый ярус. С этими работами связаны работы 2, 5, 8, 11,14,15, попадающие во второй ярус. Оставшиеся

работы 3,6,9 и 10 располагаются в третьем ярусе. Вводим пустую работу в связь, проходящую от 7-й к 9-й работе через 2-й ярус. Результирующая ярусно-параллельная форма приведена на рисуноке 3.

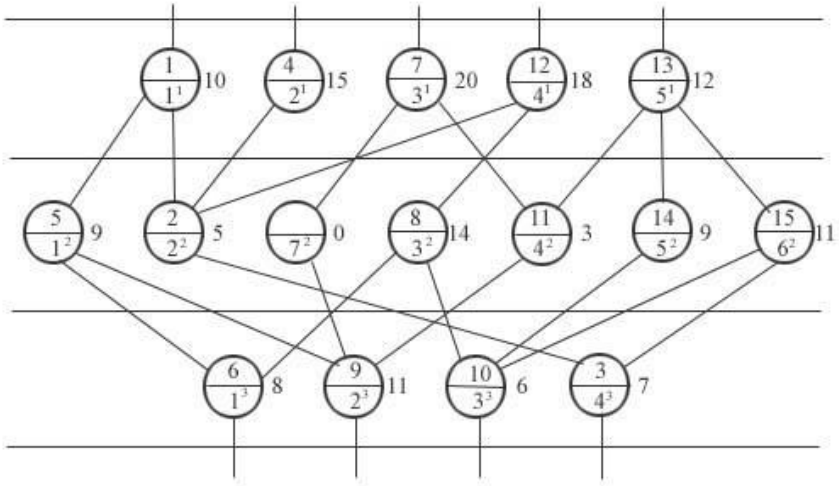


Рисунок 3 - Результирующая ярусно-параллельная форма

Упорядочим обозначения работ в соответствии с принятыми выше. Для этого пронумеруем работы в пределах каждого яруса и проставим полученные номера в нижней части обозначения работы.

Запишем у каждой работы ее длину для случая $S_0 = 1$

2. Расчет основных показателей ЯПФ.

Ширина ярусов равна:

$b^1=5$; $b^2=6$; $b^3=4$.

Ширина графа B=6.

Длины ярусов равны:

$l^1 = 20$; $l^2 = 14$; $l^3 = 11$;

Дебалансы ярусов:

$d^1 = 20 – 10 = 10$;

$d^2 = 14 – 3 = 11$;

$d^3 = 11 – 6 = 5$.

Дебаланс графа равен:

$$d = \sum_{i=1}^{3} d^f = 10+11+5=26$$

Заполнение яруса:

$E^1 = (20 – 10) + (20 – 15) + (2O – 18) + (20 – 12) = 26$;

$E^2 = (14 – 9) + (14 – 5) + (14 – 3) + (14 – 8) + (14 – 11) = 34$;

$E^3 = (11 – 8) + (11 – 6) + (11 – 7) = 12$

Заполнение графа:

E =26 + 34 + 12 = 71

Разброс графа:

Q = (6 – 5) + (6 – 4) = 3.

3. ВАРИАНТЫ ЗАДАНИЯ.

По заданной преподавателем букве выбрать матрицу графа по таблицах 3 – 7. По заданному числу из таблицы 8 выбрать значения трудоемкости работ при обслуживании их процессором с относительной производительностью $S_э$ = 1.

Необходимо выполнить:

- преобразовать граф в ярусно-параллельную форму;
- рассчитать основные показатели ЯПФ.

4. ФОРМА ОТЧЕТА.

Отчет должен включать:

- вариант задания;
- чертеж полученной ЯПФ;
- расчеты основных характеристик ЯПФ.

Таблица 3 – Исходные данные

вариант А		Выходы работ														
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Входы работ	1							1								
	2															
	3		1			1										
	4			1											1	
	5															
	6															
	7															
	8		1					1								
	9						1	1								
	10	1		1					1							
	11								1	1						
	12						1									
	13						1		1				1			
	14					1										
	15	1	1	1											1	

Таблица 4 – Исходные данные

вариант Б	Выходы работ														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Входы работ	1											1	1			
	2															
	3		1										1			
	4		1			1	1									
	5															
	6															
	7	1		1												
	8			1	1	1										
	9	1			1						1					
	10		1				1					1				
	11															
	12															
	13	1		1												
	14										1					
	15				1						1					

Таблица 5 – Исходные данные

вариант В		Выходы работ														
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Входы работ	1						1	1	1		1					
	2															
	3	1														1
	4	1				1					1					
	5		1				1	1								
	6															
	7															
	8															
	9															
	10															

	11					1					1					1
	12														1	1
	13														1	
	14		1					1								
	15						1			1						

Таблица 6 – Исходные данные

вариант Г		Выходы работ														
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Входы работ	1						1					1				
	2							1				1				
	3															
	4					1	1						1	1		
	5			1				1								
	6															
	7															
	8		1												1	1
	9						1									
	10	1	1							1						1
	11															
	12															
	13															
	14													1		
	15												1			

Таблица 7 – Исходные данные

вариант Д		Выходы работ														
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Входы работ	1															
	2	1				1										

	3		1		1											
	4	1				1							1			
	5															
	6							1						1		
	7															
	8	1					1									
	9		1						1							
	10						1								1	
	11				1			1								
	12															
	13															
	14												1	1		
	15				1										1	

Таблица 8 – Исходные данные

№ варианта	Длины работ														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
01	21	8	12	15	10	24	5	13	4	18	12	23	7	14	91
02	10	13	25	4	11	20	19	7	9	16	12	2	16	13	22
03	36	24	40	20	17	29	42	12	31	15	22	29	18	36	27
04	20	24	42	9	13	18	36	31	29	17	23	26	34	19	15
05	17	12	31	25	8	21	15	44	14	23	30	28	16	24	28
06	21	13	40	9	8	25	20	13	32	19	27	12	18	37	16
07	58	32	25	44	36	22	29	32	49	17	44	36	25	35	20
08	28	39	19	40	25	32	37	43	22	30	46	41	24	34	26
09	72	60	54	63	70	40	46	50	55	45	40	66	51	36	38
10	65	50	44	46	54	42	52	36	45	55	30	34	42	59	32
11	40	52	35	48	61	57	41	30	52	48	41	34	37	47	59
12	21	17	35	44	26	20	38	45	27	30	40	52	21	24	31

13	32	24	61	52	37	55	43	48	29	32	58	36	27	33	40
14	44	24	76	62	27	51	59	66	31	48	52	36	48	61	42
15	28	33	44	37	58	40	62	37	25	51	47	30	22	42	54

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 2

Определение критического (длиннейшего) пути ЯПФ.

1. ЦЕЛЬ ЗАНЯТИЯ: практическое закрепление знаний об определении нижней границы возможного времени реализации программы.

2. ОСНОВНЫЕ СВЕДЕНИЯ.

Для определения минимально-возможного времени реализации заданного графа на параллельной вычислительной системе, состоящей из процессоров с относительной производительностью *So=1* и числом превышающим ширину графа B, необходимо определить критический (длиннейший) путь от первого до N-го яруса.

Для этого введем в линии связи, транзитивно проходящие через ярус, пустые работы (время реализации которых равно нулю).

Нумеруем работы а пределах каждого яруса *f:* $i^f = 1^f, 2^f, \ldots b^f$

Для определения критического пути воспользуемся алгоритмом нахождения пути в графе с дугами произвольной длины. Преобразуем граф *G* к сетевому, для этого введем нулевой ярус $f = 0$ и поместим туда фиктивную вершину *X*, которую соединим дугами со всеми вершинами первого яруса и присвоим каждой дуге длину равную длительности соответствующей вершины. Так, дуге (X, l^1) присваивается длина l^1, равная трудоемкости этой вершины. Дугам, соединяющим $(f - 1)$-й ярус с *f*-м ярусом, присваивается длины равные трудоемкости соответствующих работ *f*-го яруса. Все вершины *N*-го яруса соединяются с

единственной вершиной *Z* в фиктивном *(N + 1)*-м ярусе, и имеют длину равную нулю.

В результате такого преобразования получим сетевой граф *G`* с истоком *X* и стоком *Z*. Исходная задача сводится к нахождению длиннейшего пути в сети *G`*.

Алгоритм вхождения экстремального пути на сети *G`* заключается в следующем

1. Вершину *X* помечаем индексом $\Lambda_0 = 0$, остальные X_i помечаем индексом $\Lambda_i = 0$.

2. Ищем такую дугу (X_I, X_J), для которой $\Lambda_j - \Lambda_i < l(X_I, X_J)$ и заменяем индекс Λ_j индексом $\Lambda'_j = \Lambda_i + l(X_I, X_J)$. Продолжаем этот процесс до тех пор пока можно увеличить индекс хотя бы одной вершины.

3. Находим длиннейший путь, начиная с вершины *Z*. Для этого находим вершину X_{p1} инцидентную *Z*, для которой выполняется условие $\Lambda_z - \Lambda_{p1} = l(X_{p1}, Z)$. Далее находим X_{p2} инцидентную X_{p1} , для которой выполняется условие $\Lambda_{p1} - \Lambda_{p2} = l(X_{p2}, X_{p1})$ и так до тех пор пока не дойдем до вершины *X*: Экстремальный путь $M_э = (X, X_{pn}, \ldots, X_{p2}, X_{p1}, Z)$ длина которого Λ_z является максимальной.

Пример нахождения критического пути ЯПФ алгоритма

Определим критический путь для ЯПФ алгоритма, который приведён на рисунке 3, предыдущей работы. Построим сеть по данной ЯПФ, как было описано выше. Полученная сеть *G`* представлена на рисунке 1. Согласно алгоритму помечаем вершину X индексом 0, индекс ∞ у остальных вершин не ставим, чтобы рисунок был нагляднее. При изменении индекса старый будем зачеркивать и справа писать новый.

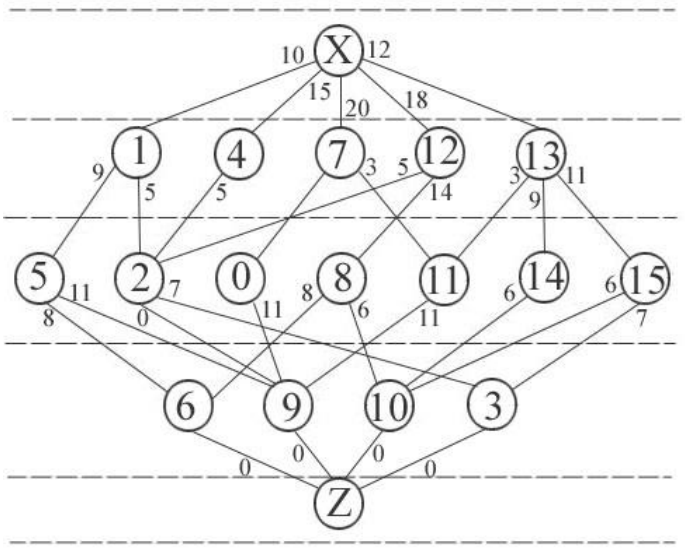


Рисунок 1 - Сеть *G`*

Процесс изменения индексов виден из рисунка 2. Полученный критический путь $M_0 = (X, 12, 8, 6, Z)$, его величина $Tкр = \Lambda z = 40$.

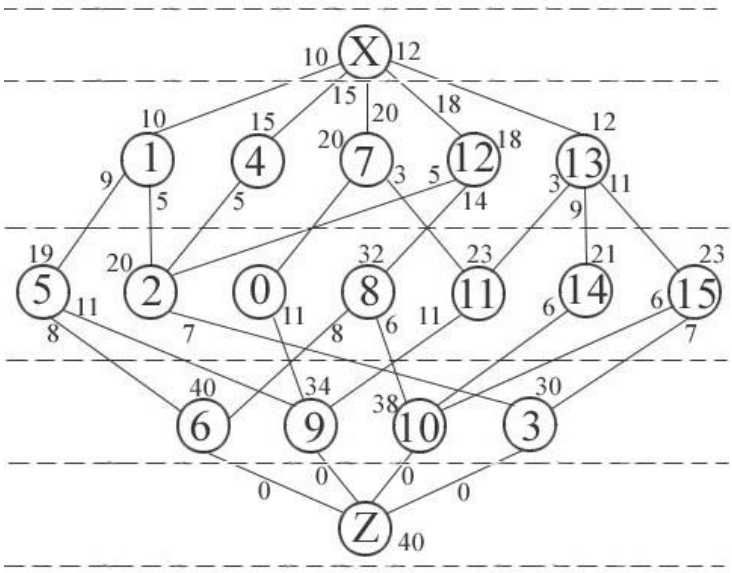


Рисунок 2 - Процесс изменения индексов

3. ВАРИАНТЫ ЗАДАНИЯ.

По заданной преподавателей букве выбрать матрицу графа программы по табл. 3 – 7 (предыдущей работы). По заданной цифре из таблицы 8 (предыдущей работы) выбрать значения трудоемкости работ при обслуживании их процессором с относительной производительностью *Sэ = 1*.

Необходимо выполнить:

1) преобразовать граф в ярусно-параллельную форму;

2) построить по графу ЯПФ *G* сеть *G`*;

3) определить на сети *G`* критический путь M_0 и значение *Ткр* .

4. ФОРМА ОТЧЕТА.

Отчет должен включать:

- вариант задания;
- чертеж полученной ЯПФ;
- чертеж сети *G`*;
- критический путь M_0 и значение *Ткр*.

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №3

Распределение загрузки многопроцессорной системы.

1. ЦЕЛЬ ЗАНЯТИЯ: практическое закрепление знаний о распределении загрузки между процессорами многопроцессорной системы.

2. ОСНОВНЫЕ СВЕДЕНИЯ.

Для характеристики загрузки многопроцессорных вычислительных систем будем использовать следующие показатели:

а) относительная производительность процессоров:

$$S_m = \frac{l^{f_{iэ}}}{l^{f_{im}}},$$

где $l^f_{iэ}$, $i = 1, 2, \ldots, b^f$, $f = 1, 2, \ldots, N$, длина (трудоемкость) i-той работы в f-м ярусе при обслуживании ее с помощью некоторого эталонного процессора, для которого принято $Sэ=1$, Sm – относительная производительность m-го процессора $m = 1, 2, \ldots, n$.

б) $T^f_{реш}$ - длительность обслуживания с помощью n процессоров работ f-го яруса ЯПФ; этот показатель оценивается длительностью работы наиболее загруженного процессора на f-м ярусе;

в) $T_{реш}$ - длительность решения всей задачи (обслуживание всех работ ЯПФ):

$$T_{реш} = \sum_{f=1}^{N} T^f_{реш},$$

г) $T^f\ пр.m.$- длительность простоя m-го процессора при обслуживании операторов f-го яруса;

$T^f\ пр.m. = T^f_{реш} - T^f_{реш.m}$, $m=1, 2, \ldots, n$,

где $T^f_{реш.m}$ - загруженность m-го процессоре на f-м ярусе;

д) суммарные простои процессоров:

$T^f\ пр = \sum_{m=1}^{n} T^f\ пр.m$ - на f-м ярусе;

$Tпр = \sum_{f=1}^{N} T^f\ пр$ - при обслуживании ЯПФ;

е) коэффициент простоя процессоров:

$$Kпр = \frac{Tпр}{Tреш * n} * 100\%;$$

ж) средняя относительная производительность одного процессора системы:

$$Scp = \frac{1}{n} \sum_{m=1}^{n} So;$$

з) повышение производительности системы за счет организации параллельных вычислений на n процессорах:

$$K = \frac{1}{Scp * Tреш} \sum_{f=1}^{N} \sum_{i=1}^{b^f} l^f_{iэ}$$

Пусть на некотором этапе решения задачи, соответствующей обработке *f*-го яруса имеется b^f работ, принадлежащих *f*-му ярусу, и *n* процессоров, входящих в ВС. Будем считать, что каждая из работ может быть выполнена на одном из процессоров. Процессоры могут иметь различную производительность.

Показатели относительной производительности процессоров представим в виде множества $S = \{S_1, S_2, \ldots, S_n\}$. Располагаем процессоры в список по мере убывания их производительности: $П = \{П_1, П_2, \ldots, П_n\}$, где процессор $П_i$ характеризуется:

$$Si\max = \max_{m=1,2,\ldots,n}(Sm, S / Sj\max)_{j=1,2,\ldots,i};$$

Располагаем также в список по мере убывания длины работ, находящихся в рассматриваемом *f*-м ярусе:

$X^f = \{x^f 1, x^f 2, \ldots, x^f b\}$, где работа $x^f i$ характеризуется длиной:

$$l^f i\max э = \max_{i=1,2,\ldots,b^f}(l^f iэ, L^f / l^f j\max)_{j=1,2,..,i}$$

$$L^f = \{l^f{}_{1э}, l^f{}_{2э}, \ldots, l^f{}_{bэ}\}$$

В процессе распределения самая длинная работа ставится для реализации на самый производительный процессор, и далее работы распределяются согласно соответствующим позициям обоих списков слева направо.

Если $n>b^f$, то неиспользованными остаются процессоры, имеющие наименьшую производительность.

В результате использования данной методики обеспечивается распределение, близкое к оптимальному.

Для получения точного решения эта задача сводится к распределительной задаче линейного программирования, которая решается известными методами.

Пример распределения загрузки.

В качестве примера распределим загрузку для реализации алгоритма, заданного ЯПФ (рисунок 1). Алгоритм будет реализовываться на вычислительной системе состоящей из шести параллельно работающих процессоров, относительная производительность которых, по сравнению с эталонным, производительность которого принята за единицу, приведена в таблица 1.

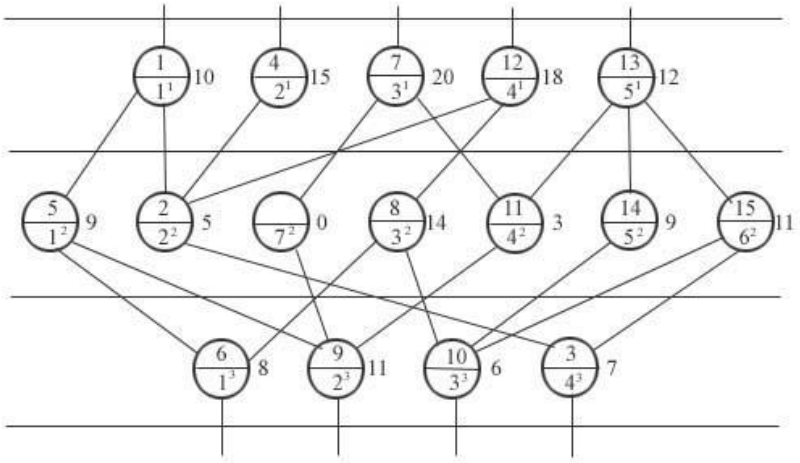


Рисунок 1 – Алгоритм ЯПФ

Таблица 1 – Эталон производительности

N процессора		$П_2$	$П_3$	$П_4$	$П_5$	$П_6$	$П_1$
Sm		1.6	1.3	1	0.8	0.6	0.3
f=1	i^1	3^1	4^1	2^1	5^1	1^1	
	l^1im	12.5	13.8	15	15	16.7	
f=2	i^2	3^2	6^2	1^2	5^2	2^2	4^2
	l^2im	8.8	8.5	9	11.3	8.4	10
f=3	i^3	2^3	1^3	4^3	3^3		
	l^3im	6.9	6.2	7	7.5		

Рассчитаем длины работ с учетом их реализации на процессоре *m*-го типа по формуле:

$$l^f im = \frac{l^f iэ}{Sm},$$

где $l^f iэ$ – время реализации работы на процессоре, производительность которого принята за эталонную Sэ = 1. Используя методику решения данной задачи, изложенную выше, заполним табл. 3.1. Наиболее трудоемкая работа оказывается распределенной на наиболее производительный процессор.

Теперь длины ярусов $l^1 = T^1реш = 18.7$, $l^2 = T^2реш = 11.3$, $l^3 = T^3реш = 7.5$, где $T^fреш$ – время решения на *f*-м ярусе, *f* = 1,2,3.

В результате имеем:

а) время решения задачи в системе:

$$Треш = \sum_{f=1}^{3} T^f реш = 18,7 + 11,3 + 7,5 = 35,5;$$

б) суммарные простои процессоров:

$$Тпр = \sum_{f=1}^{N} T^f пр = 27,2 + 11,8 + 17,4 = 56,4;$$

в) коэффициент простоя процессоров:

$$Кпр = \frac{Тпр}{Треш * n} * 100\% = \frac{56.4 * 100\%}{35.5 * 8} = 27\%;$$

г) повышение производительности системы по сравнению со случаем использования одного процессора:

$$Scp = \frac{1,6 + 1,3 + 0,8 + 0,6 + 0,3}{6} = \frac{5,6}{6} = 0,93$$

$$Кпр = \frac{158}{0,93 * 35,5} = 4,8$$

3. ВАРИАНТЫ ЗАДАНИЯ

По заданной преподавателем первой букве выбрать матрицу графа программы по табл. 1.3. - 1.7. По заданной цифре из табл. 1.8 выбрать значения трудоемкости работ при обслуживании их процессором с относительной производительностью *Sэ* = 1. По заданной второй букве выбрать относительные производительности процессоров из таблицы 2.

Таблица 2 - Относительные производительности процессоров

N варианта	Относительная производительность *Sm*					
	S_1	S_2	S_3	S_4	S_5	S_6
Е	0.3	1.2	1	0.8	1.6	0.7
Ж	1.4	0.5	1.2	0.6	0.9	1.8
З	0.9	1.8	0.6	1.2	1	0.4
И	1	1.5	0.8	1.2	0.6	0.4
К	1.2	0.6	1.5	0.8	1.1	0.5

Необходимо выполнить:
1) преобразовать граф в ярусно-параллельную форму;
2) построить по графу ЯПФ G сеть G`;
3) распределить работы по процессорам в каждом ярусе;
4) вычислить основные характеристики распределения.

4. ФОРМА ОТЧЕТА.

Отчет должен включать:
- вариант задания;
- чертеж полученной ЯПФ;
- чертеж сети G`;
- таблицу распределении работ;
- расчет характеристик распределения.

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №4

Методы устранения избыточности кодирования.

1. Метод Шеннона-Фано.

Рассмотрим метод эффективного кодирования, предложенный независимо друг от друга Фано и Шенноном. Получаемый по этому методу код называют кодом Шеннона-Фано или кодом Фано.

Таблица 1 - Построение кода по методу Шеннона-Фано.

Символ источника	Вероятность появления $P(a_i)$	Разбиение на подгруппы	Кодовые слова длиной L_i
a_1	0,4	I	1
a_2	0,2	I	01
a_3	0,2	I	001
a_4	0,1	I	0001
a_5	0,05	I	00001
a_6	0,05	II	00000

Кодирование по методу Шеннона-Фано, таблица 1. Пусть задан алфавит источника и требуется построить код более экономный, чем равномерный код. Алгоритм построения кода заключается в следующем:

1. Расположить все имеющиеся символы в порядке убывания вероятности.
2. Разбить символы на две группы (верхнюю и нижнюю), так чтобы суммарные вероятности появления символов в группах были как можно ближе. Для первой (верхней) группы в качестве первого символа кодовых слов будем использовать, например, 1 , а для второй (нижней) группы - 0 .
3. Далее каждую группу снова надо разделить на две части (верхнюю и нижнюю) с как можно более близкими суммарными вероятностями. В качестве второго символа кодовых слов для первых подгрупп используем –1 , для вторых подгрупп –0.
4. Процесс повторяется, пока в подгруппах не останется по одному символу источника.

Для рассмотренного 6-символьного алфавита наилучший равномерный код имеет длину 3 двоичных символа

$2^2<6<2^3$, $n=3$

Среднее число двоичных символов кода Шеннона-Фано, приходящихся на один символ алфавита источника, равно:

$\tilde{n} = L_i * P(a_i) = 1*0,4 + 2*0,2 + 3*0,2 + 4*0,1 + 5(0,05+0,05) = 2,3$ *бит*

Энтропия источника равна:

$$H(A) = -\sum_{i=1}^{6} P(a_i) * \log_2 P(a_i) = 2,22 \qquad \textit{бит}$$

В ряде случаев по методу Шеннона-Фано выгодно кодировать не отдельные символы источника, а блоки из нескольких символов.

Если применить этот метод к кодированию трехсимвольных блоков источника, среднее число двоичных символов, приходящихся на один символ источника, будет еще ближе к энтропии алфавита источника.

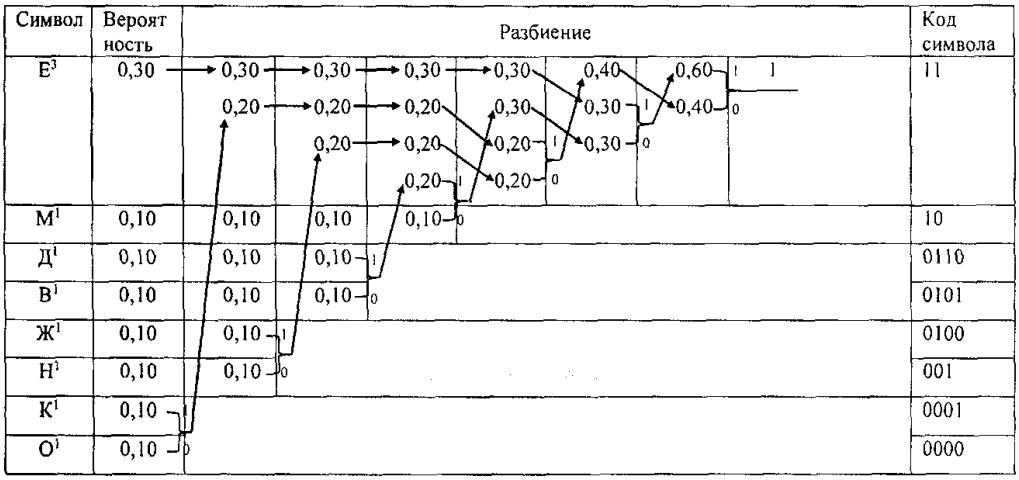
Кодирование по методу Хаффмана. Более экономный код, чем метод Шеннона-Фано, он нашел широкое применение на практике, например, в факсимильных устройствах. Построение кода Хаффмана основывается на преобразовании, которое называется сжатием алфавита.

2. Метод Хаффмана.

Алгоритм кодирования:

1. Расположить символы исходного алфавита А в порядке убывания вероятности.
2. Два наименее вероятных символа алфавита А будем считать одним символом нового сжатого алфавита A_1.
3. Располагаем символы алфавита A_1 в порядке убывания вероятности. И снова подвергаем его сжатию, как в пункте 2.
4. Повторяем процедуру, пока не придем к алфавиту, содержащему всего два символа.
5. Припишем символам последнего алфавита кодовые обозначения 0 (например - верхнему) и 1 (в нашем примере – нижнему). Это старшие символы будущих кодовых слов.
6. В предпоследнем алфавите кодовые обозначения получаются следующим образом:
   - Символ, который сохранился в последнем алфавите, имеет то же кодовое обозначение.
   - Символам, которые слились в последнем алфавите, приписывают справа 0 (в нашем примере – верхнему символу) и 1 (нижнему символу).
7. Повторяем процедуру, последовательно возвращаясь к исходному алфавиту.

Таблица 2. Построение кода по методу Хаффмана.



В данном случае среднее число двоичных символов кода, приходящихся на один символ источника, получилось таким же, как в коде Шеннона-Фано. В общем случае математиками доказано, что код Хаффмана является самым экономным в том смысле, что никакой другой метод кодирования алфавита не позволяет получить среднее число двоичных символов на один символ алфавита меньше, чем в случае кода Хаффмана таблица 2.

3. Метод Лемпеля-Зива (LZ77).

Алгори́тм Ле́мпеля — Зи́ва — Ве́лча (Lempel-Ziv-Welch, LZW) — это универсальный алгоритм сжатия данных без потерь, созданный Авраамом Лемпелем (англ. Abraham Lempel), Яковом Зивом (англ. Jacob Ziv) и Терри Велчем(англ. Terry Welch). Он был опубликован Велчем в 1984 году, в качестве улучшенной реализации алгоритма LZ78, опубликованного Лемпелем и Зивом в 1978 году. Алгоритм разработан так, чтобы его можно было быстро реализовать, но он не обязательно оптимален, поскольку он не проводит никакого анализа входных данных.

Акроним «LZW» указывает на фамилии изобретателей алгоритма: Лемпель, Зив и Велч, но многие[кто?] утверждают, что, поскольку патент принадлежал Зиву, то метод должен называться алгоритмом Зива — Лемпеля — Велча.

Данный алгоритм при сжатии (кодировании) динамически создаёт таблицу преобразования строк: определённым последовательностям символов (словам) ставятся в соответствие группы бит фиксированной длины (обычно 12-битные). Таблица инициализируется всеми 1-символьными строками (в случае 8-битных символов — это 256 записей). По мере кодирования алгоритм просматривает текст символ за символом, и сохраняет каждую новую, уникальную 2-символьную строку в таблицу в виде пары код/символ, где код ссылается на соответствующий первый символ. После того как новая 2-символьная строка сохранена в таблице, на выход передаётся код первого символа. Когда на входе читается очередной символ, для него по таблице находится уже встречавшаяся строка максимальной длины, после чего в таблице сохраняется код этой строки со следую-

щим символом на входе; на выход выдаётся код этой строки, а следующий символ используется в качестве начала следующей строки.

Алгоритму декодирования на входе требуется только закодированный текст, поскольку он может воссоздать соответствующую таблицу преобразования непосредственно по закодированному тексту.

Алгоритм[править | править вики-текст]

Инициализация словаря всеми возможными односимвольными фразами. Инициализация входной фразы W первым символом сообщения.

Найти в словаре строку W наибольшей длины, которая совпадает с последними принятыми символами.

Считать очередной символ K из кодируемого сообщения.

Если КОНЕЦ_СООБЩЕНИЯ, то выдать код для W, иначе.

Если фраза WK уже есть в словаре, присвоить входной фразе W значение WK и перейти к Шагу 3, иначе выдать код W, добавить WK в словарь, присвоить входной фразе W значение K и перейти к Шагу 3.

Конец.

Пример[править | править вики-текст]

Данный пример показывает алгоритм LZW в действии, показывая состояние выходных данных и словаря на каждой стадии, как при кодировании, так и при раскодировании сообщения. С тем чтобы сделать изложение проще, мы ограничимся простым алфавитом — только заглавные буквы, без знаков препинания и пробелов. Сообщение, которое нужно сжать, выглядит следующим образом:

Кодирование[править | править вики-текст]

Без использования алгоритма LZW, при передаче сообщения, как оно есть — 25 символов по 5 бит на каждый — оно займёт 125 бит. Сравним это с тем, что получается при использовании LZW:

Символ: Битовый код: Новая запись словаря:
(на выходе)

T 20 = 10100
O 15 = 01111 27: TO
B 2 = 00010 28: OB
E 5 = 00101 29: BE
O 15 = 01111 30: EO
R 18 = 10010 31: OR <--- со следующего символа начинаем использовать 6-битные группы
N 14 = 001110 32: RN
O 15 = 001111 33: NO
T 20 = 010100 34: OT
TO 27 = 011011 35: TT
BE 29 = 011101 36: TOB
OR 31 = 011111 37: BEO
TOB 36 = 100100 38: ORT
EO 30 = 011110 39: TOBE
RN 32 = 100000 40: EOR

OT 34 = 100010 41: RNO
0 = 000000 42: OT#

Общая длина = 6*5 + 11*6 = 96 бит.

Таким образом, используя LZW, мы сократили сообщение на 29 бит из 125 — это почти 22 %. Если сообщение будет длиннее, то элементы словаря будут представлять всё более и более длинные части текста, благодаря чему повторяющиеся слова будут представлены очень компактно.

Декодирование[править | править вики-текст]

Теперь представим, что мы получили закодированное сообщение, приведённое выше, и нам нужно его декодировать. Прежде всего, нам нужно знать начальный словарь, а последующие записи словаря мы можем реконструировать уже на ходу, поскольку они являются просто конкатенацией предыдущих записей.

Данные: На выходе: Новая запись:
Полная: Частичная:
10100 = 20 T 27: T?
01111 = 15 O 27: TO 28: O?
00010 = 2 B 28: OB 29: B?
00101 = 5 E 29: BE 30: E?
01111 = 15 O 30: EO 31: O?
10010 = 18 R 31: OR 32: R? <---- начинаем использовать 6-битные группы
001110 = 14 N 32: RN 33: N?
001111 = 15 O 33: NO 34: O?
010100 = 20 T 34: OT 35: T?
011011 = 27 TO 35: TT 36: TO? <---- для 37, добавляем только первый элемент
011101 = 29 BE 36: TOB 37: BE? следующего слова словаря
011111 = 31 OR 37: BEO 38: OR?
100100 = 36 TOB 38: ORT 39: TOB?
011110 = 30 EO 39: TOBE 40: EO?
100000 = 32 RN 40: EOR 41: RN?
100010 = 34 OT 41: RNO 42: OT?
000000 = 0 #

Единственная небольшая трудность может возникнуть, если новое слово словаря пересылается немедленно. В приведённом выше примере декодирования, когда декодер встречает первый символ, T, он знает, что слово 27 начинается с T, но чем оно заканчивается? Проиллюстрируем проблему следующим примером. Мы декодируем сообщение ABABA:

Данные: На выходе: Новая запись:
Полная: Частичная:
.
.
.
011101 = 29 AB 46: (word) 47: AB?

101111 = 47 AB? <--- что нам с этим делать?

На первый взгляд, для декодера это неразрешимая ситуация. Мы знаем наперёд, что словом 47 должно быть ABA, но как декодер узнает об этом? Заметим, что слово 47 состоит из слова 29 плюс символ, идущий следующим. Таким образом, слово 47 заканчивается на «символ, идущий следующим». Но, поскольку это слово посылается немедленно, то оно должно начинаться с «символа, идущего следующим», и поэтому оно заканчивается тем же символом, что и начинается, в данном случае — A. Этот трюк позволяет декодеру определить, что слово 47 — это ABA.

В общем случае, такая ситуация появляется, когда кодируется последовательность вида cScSc, где c — это один символ, а S — строка, причём слово cS уже есть в словаре.

Сжатие информации

Метод сложения.

Используя таблицы выполнить сжатие своих инициалов одним из методов:

- сложением по модулю два;
- сложением по правилам двоичной логики.

Разрешается придумать любую комбинацию бит длинной не менее 24-х бит.

Метод замены символов.

Сжать информацию, считаем, что каждый символ содержит восемь бит. Сделать выводы.

Условие задачи:

1 2 3 4 5 6 7
1 2 3 4 5 8 6
2 1 3 4 5 2 4
2 1 3 4 5 2 9
4 2 9 4 5 2 9
4 2 9 4 5 2 9
4 2 9 4 5 2 9
5 2 9 4 5 2 1

Разрешается придумать комбинацию аналогичной размерности.

Скрэмблирование. Задание №2.

Для полученной комбинации реализовывать процедуру скрэмблирования с соотношением: $B_i = A_i \ B_{i-3} \ B_{i-5}$.

где B_i - двоичная цифра результирующего кода, полученная на i-м такте работы скрэмблера,

A_i - двоичная цифра исходного кода, поступающая на i-м такте на вход скрэмблера,

B_{i-3} и B_{i-5} - двоичные цифры результирующего кода, полученные на предыдущих тактах работы скрэмблера, соответственно на 3 и на 5 тактов ранее

текущего такта, и объеденённые операцией исключающего ИЛИ (сложение по модулю 2).

Сделать вывод.

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №5.

Кодирование источника сообщений.

Сжатие информации

Метод сложения.

Используя таблицы выполнить сжатие своих инициалов одним из методов:

- сложением по модулю два;
- сложением по правилам двоичной логики.

Разрешается придумать любую комбинацию бит длинной не менее 24-х бит.

Метод замены символов.

Сжать информацию, считаем, что каждый символ содержит восемь бит. Сделать выводы.

Условие задачи:

1 2 3 4 5 6 7
1 2 3 4 5 8 6
2 1 3 4 5 2 4
2 1 3 4 5 2 9
4 2 9 4 5 2 9
4 2 9 4 5 2 9
4 2 9 4 5 2 9
5 2 9 4 5 2 1

Разрешается придумать комбинацию аналогичной размерности.

Скремблирование

Для полученной комбинации реализовывать процедуру скремблирования с соотношением: $B_i = A_i \; B_{i-3} \; B_{i-5}$.

где B_i - двоичная цифра результирующего кода, полученная на i-м такте работы скрэмблера,

A_i - двоичная цифра исходного кода, поступающая на i-м такте на вход скрэмблера,

B_{i-3} и B_{i-5} - двоичные цифры результирующего кода, полученные на предыдущих тактах работы скремблера, соответственно на 3 и на 5 тактов ранее текущего такта, и объединённые операцией исключающего ИЛИ (сложение по модулю 2).

Сделать вывод.

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 6

Расчёт параметров циклического кода.

Задача № 1.

Для циклического кода с $d_0 = 3$ составить кодовую комбинацию для передачи символа, соответствующего двум последним цифрам номера военного билета.

Требуется:

1. Определить необходимое число информационных k и проверочных r элементов.
2. Выбрать образующий полином и составить требуемую кодовую комбинацию.
3. Доказать правильность построения КК.
4. Проверить выполнение свойства линейного кода.
5. Определить класс полученного циклического кода (полный или укороченный).
6. Ввести ошибку в разряд j исходной КК и доказать возможность ее исправления на приеме.

Вариант задач

Решение:

1. Кодовая комбинация, соответствующая последним цифрам номера студенческого билета

$$5(10) = 101\ (2)$$

Число информационных элементов k =5, для нахождения числа проверочных элементов кода воспользуемся формулой

$r \geq \log_2(n+1)$ или $2^r \geq k + r + 1 \Rightarrow 2^r \geq r + 4$, отсюда r = 3.

$$n = k + r = 3 + 3 = 6$$

2. По таблице образующих полиномов для r = 3 находим

$$Pr(x) = x^3+x+1.$$

Для кода 101 = $A(x) = x^2+1$ $B(x) = A(x) \cdot x^3 \oplus R(x)$ – этот полином представляет собой разрешающую КК циклического кода:

$$A(x) \cdot x^3 = x^5 + x^3$$

Разделим $A(x) \cdot x^3$ на Pr(x), чтобы получить остаток R(x), который определяет код проверочных символов

$$\begin{array}{l} x^5 + x^3 \quad | \underline{x^3 + x + 1} \\ \underline{x^5 + x^3 + x^2} \quad x^2 \\ \quad x^2 \end{array}$$

$R(x) = x^2 \Rightarrow r = 100$

Искомую комбинацию находим как сумму $A(x) \cdot x^3$ и $R(x) - x^5 + x^3 + x^2 \Rightarrow$

$\underbrace{101}\ \underbrace{100}$

$$\underbrace{k \quad r}_{n}$$

3.Синдромом циклического кода является результат деления кодовой комбинации на образующий полином, если остаток отделения равен «0», значит циклическая комбинация составлена верно

$$x^5 + x^3 + x^2 \mid \underline{x^3 + x + 1}$$
$$\underline{x^5 + x^3 + x^2} \quad x^2$$
$$0$$
$$R(x) = 0$$

4. Для проверки свойства линейности кода формируем путем циклического сдвига две разрешенные комбинации и складываем их по mod 2

$$101\ 100$$
$$\oplus \quad \underline{010\ 110}$$
$$111\ \ 010 \Rightarrow x^5 + x^4 + x^3 + x \text{ и разделим на Pr(x)}$$
$$x^5 + x^4 + x^3 + x \mid \underline{x^3 + x + 1}$$
$$\underline{x^5 + x^3 + x^2} \mid x^2 + x$$
$$x^4 + x^2 + x$$
$$\underline{x^4 + x^2 + x}$$
$$0$$

R(x) = 0, свойство линейности выполняется, код линейный.

5. Для определения класса кода (полный или укороченный) воспользуемся формулой

$$r = \log_2 (n+1)$$

$3 \neq \log_2 7$, т.к. равенство не выполняется, код является укороченным.

6. Введем ошибку в разряд r – 1 = 2. Кодовая комбинация с ошибкой 111100 $\Rightarrow x^5 + x^4 + x^3 + x^2$, многочлен ошибки x^4 и разделим поочередно на Pr(x)

$$x^5 + x^4 + x^3 + x^2 \mid \underline{x^3 + x + 1} \qquad x^4 \mid \underline{x^3 + x + 1}$$
$$\oplus \ \underline{x^5 + x^3 + x^2} \qquad x^2+x+1 \qquad \oplus \ \underline{x^4 + x^2 + 1} \quad \underline{x}$$
$$x^4 + x^3 \qquad\qquad x^2 + 1$$
$$\oplus \ \underline{x^4 + x^2 + x}$$
$$x^3 + x^2 + x$$
$$\oplus \ \underline{x^3 + x + 1}$$
$$x^2 + 1$$

R(x) = $x^2 + 1$. R(x) = $x^2 + 1$.

Остатки совпадают, значит, ошибка действительно находится во втором разряде.

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 7

Расчёт топологии ИС

1. Структурные матрицы.

Существует единый подход к количественным оценкам элементов графа. Рассмотрим матрицы, характеризующие только ребра сети.

1. Матрица смежности: $R = \|r_{ij}\|$, $r_{ij} = \begin{cases} 1, \text{если есть ребро,} \\ 0, \text{если нет ребра.} \end{cases}$

2. Матрица длин ребер: $L = \|l_{ij}\|$, $l_{ij} = \begin{cases} 0, \text{если } i = j, \\ l_{ij}, \text{если есть ребро,} \\ \infty, \text{если нет ребра.} \end{cases}$

3. Матрица пропускных способностей: $C = \|c_{ij}\|$, c_{ij} - максимальное число бит/сек., которое может быть пропущено по каналам данного ребра, или пропускная нагрузка в Эрлангах.

Чаще вместо этого показателя используется такой показатель, как емкость ребра, определяемая как число стандартных каналов (КТЧ или ОЦК).

$$V = \|v_{ij}\|, \; r_{ij} = \begin{cases} 0, \text{если } i = j \text{ или при отсутствии каналов,} \\ \text{число каналов между } i\text{-м и } j\text{-м узлом.} \end{cases}$$

Матрица емкости сети отражает потенциальные возможности сети, так как не отражает свойств узлов, способов кроссировки, направления каналов и т. д. Это учитывается в следующей матрице.

4. Матрица прямых каналов: $U = \|u_{ij}\|$, $u_{ij} = \begin{cases} 0, \text{если } i = j \text{ или при отсутствии каналов,} \\ \text{число каналов, начинающихся в } i\text{-м узле} \\ \text{и заканчивающихся в } j\text{-м узле.} \end{cases}$

5. Матрица надежности: $P = \|p_{ij}\|$, $p_{ij} = 1 - q_{ij}$, где p_{ij} - вероятность нахождения ребра в работоспособном состоянии; q_{ij} - вероятность неработоспособного состояния ребра.

6. Матрица стоимости: $Ц = \|ц_{ij}\|$, $ц_{ij}$ - стоимость ребра между узлами a_i и a_j; $ц_{ii}$ - стоимость узла a_i.

2. Оценка ребер и путей в сети.

Из вышеперечисленных типовых матриц часто строят матрицы, отражающие возможности ребра по доставке информации.

1. Матрица длин каналов: $\lambda = \|l_{ij} \cdot v_{ij}\|$, где вхождения измеряются в канало-километрах.

2. Мощность сети: $G = \|l_{ij} \cdot c_{ij}\|$, где вхождения измеряются в бито-километрах или эрланго-километрах.

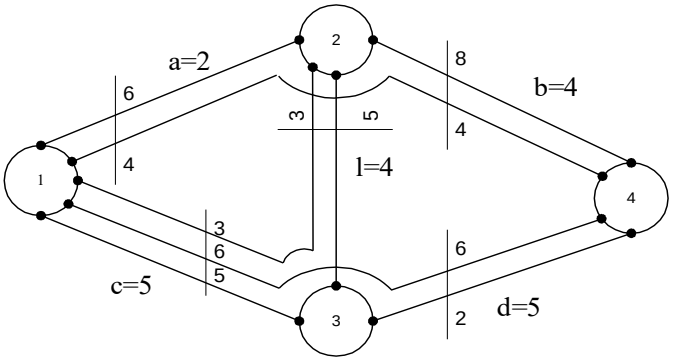
Для пути часто рассматриваются следующие характеристики:

1. Ранг пути $r(\mu_{st})$ - число входящих в данный путь ребер.

2. Длина пути – сумма длин ребер.

3. Пропускная способность пути – минимальная пропускная способность ребра пути.
4. Емкость пути – максимальное число каналов, которое может быть получено в данном пути.

Задача №1. Дана структура



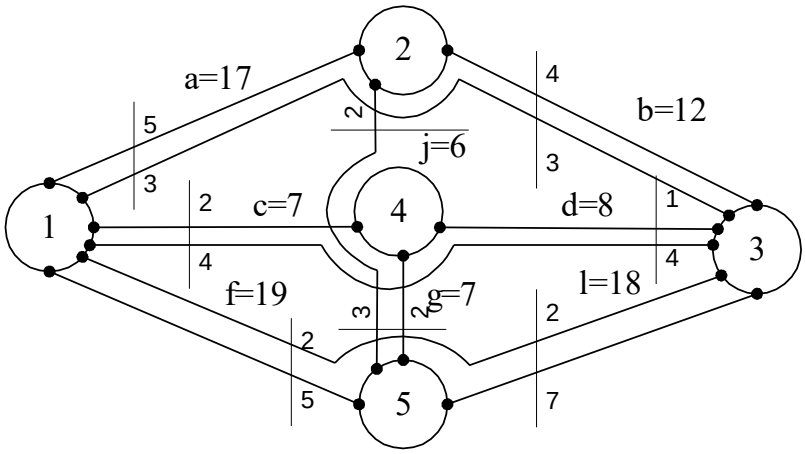
Построить матрицы **R**, **L**, **V**, **U**, **λ** Определить ранги, длину, емкость путей (любых).

Решение.

$$R=\begin{bmatrix} - & 1 & 1 & 0 \\ 1 & - & 1 & 1 \\ 1 & 1 & - & 1 \\ 0 & 1 & 1 & - \end{bmatrix},\qquad L=\begin{bmatrix} 0 & 2 & 5 & \infty \\ 2 & 0 & 3 & 4 \\ 5 & 3 & 0 & 5 \\ \infty & 4 & 5 & 0 \end{bmatrix},\qquad \lambda=\begin{bmatrix} 0 & 20 & 70 & 0 \\ 20 & 0 & 18 & 48 \\ 70 & 18 & 0 & 40 \\ 0 & 48 & 40 & 0 \end{bmatrix},$$

$$V=\begin{bmatrix} 0 & 10 & 14 & 0 \\ 10 & 0 & 6 & 12 \\ 14 & 6 & 0 & 8 \\ 0 & 12 & 8 & 0 \end{bmatrix},\qquad U=\begin{bmatrix} 0 & 9 & 5 & 10 \\ 9 & 0 & 3 & 8 \\ 5 & 3 & 0 & 2 \\ 18 & 8 & 2 & 0 \end{bmatrix}.$$

Задача №2. Дана структура



Построить матрицы **R**, **L**, **V**, **U**, **λ** Определить ранги, длину, емкость путей (любых).

$$R=\begin{bmatrix} - & 1 & 0 & 1 & 1 \\ 1 & - & 1 & 1 & 0 \\ 0 & 1 & - & 1 & 1 \\ 1 & 1 & 1 & - & 1 \\ 1 & 0 & 1 & 1 & - \end{bmatrix},\quad L=\begin{bmatrix} 0 & 17 & \infty & 7 & 19 \\ 17 & 0 & 12 & 6 & \infty \\ \infty & 12 & 0 & 8 & 18 \\ 7 & 6 & 8 & 0 & 7 \\ 9 & \infty & 18 & 7 & 0 \end{bmatrix},\quad \lambda=\begin{bmatrix} 0 & (8\times17) & 0 & (6\times7) & (8\times19) \\ \bullet & 0 & \bullet & \bullet & \bullet \\ \bullet & \bullet & 0 & \bullet & \bullet \\ \bullet & \bullet & \bullet & 0 & \bullet \\ \bullet & \bullet & \bullet & \bullet & 0 \end{bmatrix},$$

$$V=\begin{bmatrix} 0 & 8 & 0 & 6 & 8 \\ 8 & 0 & 7 & 2 & 0 \\ 0 & 7 & 0 & 5 & 9 \\ 6 & 2 & 5 & 0 & 5 \\ 8 & 0 & 9 & 5 & 0 \end{bmatrix},\quad U=\begin{bmatrix} 0 & 5 & 4 & 2 & 6 \\ 5 & 0 & 4 & 0 & 2 \\ 4 & 4 & 0 & 1 & 7 \\ 2 & 0 & 1 & 0 & 3 \\ 6 & 2 & 7 & 3 & 0 \end{bmatrix}.$$

3 Законы изменения структуры сети.

Задача изменения структуры сети ставится следующим образом.

Дано:

Число узлов в сети *N*.

Число стандартных каналов $\boldsymbol{v_{st}}$.

Все ребра являются ненаправленными. Разрешается введение обходных направлений, дополнительных узлов, организация коммутации на узлах.

Необходимо найти структуру сети, удовлетворяющую одному из критериев:

1. Min общей длины ребер: $\boldsymbol{L=\sum_{\forall i,j} l_{ij} \to min}$.

2. Min общей длины каналов: $\Lambda=\sum_{\forall i,j} v_{ij} l_{ij} \to \min$.

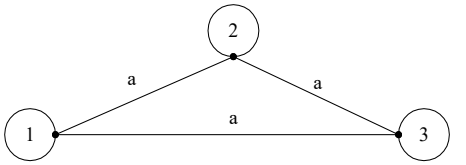
3. Min суммы расстояний для всех связей $\boldsymbol{b_{ij}}$: $\boldsymbol{D=\sum_{\forall s,t} d_{st} = \sum_{\forall i,j} l_{ij} \to min}$.

4. Min общей стоимости сети: $\boldsymbol{Ц=\alpha L+\beta\Lambda \to min}$.

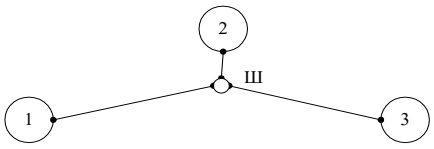
В некоторых случаях общая длина ребер односвязной сети может быть уменьшена введением дополнительного узла – точки Штейнера.

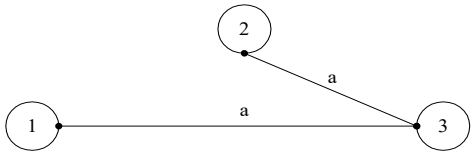
Пример 1:

полносвязная сеть



односвязная сеть

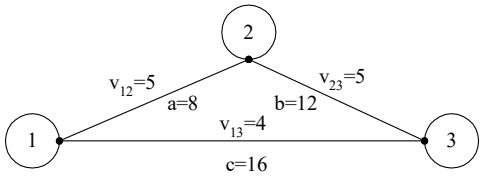




$$\boldsymbol{L_ш=\frac{a}{\sqrt{3}}3=a\sqrt{3},\ \Delta=2a-a\sqrt{3}=0.866a}.$$

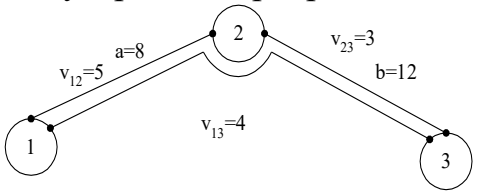
Выигрыш Δ максимален для равностороннего треугольника. Для трех узлов выигрыш от введения точки Штейнера может варьироваться от 0 до 0.866a.

Пример 2:



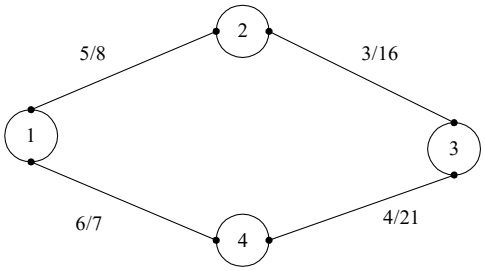
$N = 3,\ L = 8 + 12 + 16 = 36,\ \lambda 1 = 140$.

Первое преобразование – устранение ребра *c*.



$L = 8 + 16 = 20$ (расстояние уменьшилось), $\lambda 2 = 156$ (λ увеличилось).

Пример 3:

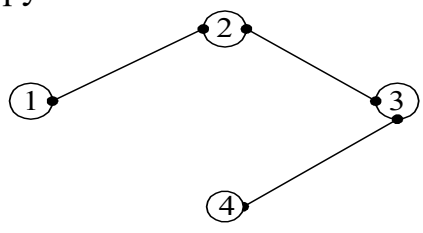


Необходимо построить односвязную сеть по критериям 1, 2.

Решение. Находим минимальный элемент в строке, остальные вычеркиваем. Отмечаем «двойники» оставшихся членов. В остальных строках осуществляется аналогичная операция.

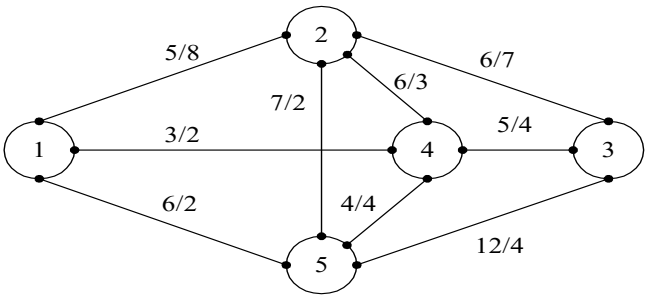
$$L = \begin{Vmatrix} 0 & 5 & 0 & 6 \\ 5 & 0 & 3 & 0 \\ 0 & 3 & 0 & 4 \\ 6 & 0 & 4 & 0 \end{Vmatrix}.$$

Получаем структуру –



$L_{min} = 12$.

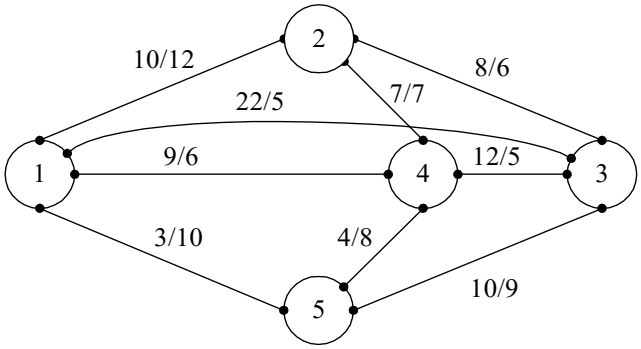
Пример 4:



Необходимо построить односвязную сеть по критериям 1, 2.
Решение.

$$L = \begin{Vmatrix} 0 & 5 & 0 & 3 & 6 \\ 5 & 0 & 6 & 6 & 7 \\ 0 & 6 & 0 & 5 & 12 \\ 3 & 6 & 5 & 0 & 4 \\ 6 & 7 & 12 & 4 & 0 \end{Vmatrix}, \ \boldsymbol{L_{min} = 18}.$$

Задача:



Необходимо построить односвязную сеть по критериям 1, 2.

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №8

Расчёт локальной вычислительной сети Ethernet

1. Построение структурной схемы сети
2. Расчет параметров трафика

Цель работы.

Привить обучающимся практические навыки по расчету параметров трафика сети Ethernet.

1. Построение структурной схемы сети

Указания по выбору варианта.

Варианты задания определяются по таблице 1 в соответствии с предпоследней цифрой номера студенческого билета, по таблице 2 - в соответствии с последней цифрой студенческого билета и по таблице 3 - в соответствии с последней цифрой текущего года.

В процессе работы необходимо произвести:

- определение среднесуточных и пиковых потоков между всеми элементами ЛВС;
- рассчитать потоки в каждой соединительной линии;
- рассчитать канальные скорости и выбрать соответствующие технологии;
- рассчитать время реакции в системе клиент-сервер для спроектированной ЛВС;
- произвести выбор оборудования;
- рассчитать стоимость выбранного оборудования.

3. Исходные данные для расчета.

Рассчитываемая ЛВС представляет собой клиент – серверную систему, объединяющую серверы и рабочие станции производственного подразделения.

Иерархическая структура сети повторяет структуру подразделения.

Исходные данные:

1. Число серверов от 3-х до 6-и в соответствии с таблицей 1. Номера серверов - $m = 1 \div 6$.

2. Число отделов – 3. Номера отделов – $i = 1 \div 3$.

3. Число рабочих групп в каждом отделе от 3-х до 6-и в соответствии с таблице 1. Номера рабочих групп – $j = 1 \div 6$.

4. Число ПЭВМ (рабочих станций, персональных компьютеров - PC) в каждой рабочей группе равно 8. Номера PC – $k = 1 \div 8$.

5. Интенсивность среднесуточных обменов для любой пары клиент – сервер одинакова и равна:

- в направлении ПЭВМ – сервер – 0,2 Кбайта/с;
- в направлении сервер – ПЭВМ – 2 Кбайт/с;
- коэффициент пульсаций трафика (отношение пиковых потоков к среднесуточным) определяется по таблице 2.

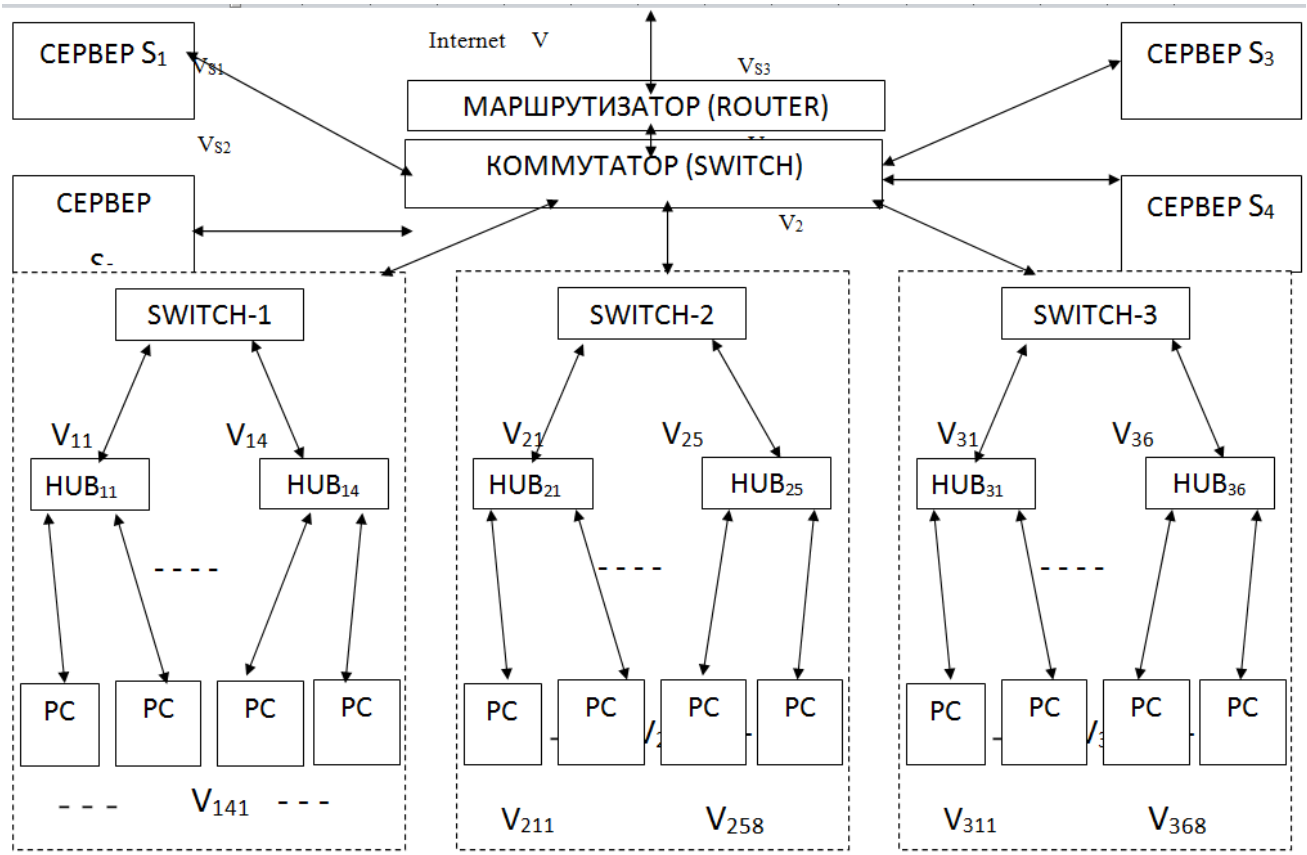


Рисунок 1 – Схема сети

6. Интенсивность среднесуточного внешнего обмена для любой ПЭВМ одинакова и равна:

- в направлении ПЭВМ -Internet - 0,05 Кбайт/с;
- в направлении Internet – ПЭВМ – 0,7Кбайт/с;
- коэффициент пульсации трафика определяется по таблице 3;

7. Интенсивность среднесуточного обмена между ПЭВМ одной рабочей группы – 0,3Кбайта/с. Коэффициент пульсации – 50:1.

8. Интенсивность среднесуточного обмена между любыми ПЭВМ подразделения не входящими в одну рабочую группу – 0,1Кбайт/с. Коэффициент пульсации – 50:1.

Таблица 1 - Число серверов и рабочих групп в отделах

Предпоследняя цифра шифра	Число серверов	Число рабочих групп		
		1 отдел	2 отдел	3 отдел
0	3	3	4	5
1	3	4	5	6
2	4	3	4	6
3	4	4	4	5
4	4	4	5	6
5	5	3	5	5

6	5	4	4	6
7	6	3	4	5
8	6	4	4	6
9	6	4	5	5

Таблица 2 - Коэффициент пульсаций трафика “клиент – сервер” и “сервер – клиент”

Последняя цифра шифра	0	1	2	3	4	5	6	7	8	9
Коэффициент пульсаций	0	5	0	5	0	5	0	5	0	5

Таблица 3 - Коэффициент пульсаций внешнего трафика

Последняя цифра текущего года	0	1	2	3	4	5	6	7	8	9
Коэффициент пульсаций	85	90	95	100	110	120	130	140	150	160

4. Порядок выполнения работы.

4.1. Составить структурную схему ЛВС, уточнив для схемы (рисунок 1) числа серверов и рабочих групп в отделах для своего варианта.

4.2. Для удобства дальнейших расчетов провести индексацию всех узлов схемы своего варианта подобно тому, как это сделано на рисунке 1. В рассматриваемой сети под узлом N (Node) понимается любое устройство типа маршрутизатор (Router), коммутатор (Switch) или (hub).

Линии связи (ветви) между узлами удобно индексировать так же, как узел, расположенный ниже по иерархии. Например, линию, соединяющую узел N_1 (в данном случае SWITCH-1) с узлом N_{11} (HUB_{11}), будем обозначать как V_{11}, а линию, соединяющую N_{36} (HUB_{36}) с узлом N_{368} (PC-368), обозначим как V_{368}. Эти обозначения будут необходимы при расчете потоков в линиях.

Самые нижние ветви (между ПЭВМ и концентратором) имеют трёхиндексное обозначение V_{ijk} , где: i – номер отдела, j – номер рабочей группы и k – номер ПЭВМ в рабочей группе.

4.3. Произвести расчет пиковых потоков для направлений обменов, указанных в пунктах 5÷8 раздела 4.3. Например, если трафик ПЭВМ – Сервер составляет 2 Кбайта/с при пульсациях 70:1, то пиковый поток составит 140 кбайт/с.

4.4. Определить суммарные пиковые потоки для каждой ветви. Это основная расчетная часть работы, требующая точности и внимательности. Ориентируясь на суммарные пиковые потоки, необходимо будет определять пропускные способности соответствующих ветвей. Фактически это сведется к выбору одного из стандартов; Ethernet (10Мбит/с), Fast Ethernet (100 Мбит/с), Gigabit Ethernet (1000 Мбит/с) или 10GBit Ethernet (10Гбит/с). Целесообразно составить таблицу со столбцами – ветвями и строками – потоками следующего вида (таблица 4).

Таблица 4 - Объёмы потоков в ветвях ЛВС

	Вид трафика	Объёмы потоков в ветвях (Мбайт/с)										
		1jk	2jk	3jk	1j	2j	3j	1	2	3	0	sm
	ПЭВМ → Сервер											
	Сервер → ПЭВМ											
	ПЭВМ → Internet											
	Internet → ПЭВМ											
	ПЭВМ → ПЭВМ одной рабоч. групы											
	ПЭВМ → ПЭВМ разных рабоч. групп											
	Суммарный трафик в ветви											
	Суммарная скорость в ветви, Мбит/с											

Структура таблицы отражает следующие особенности исходных данных:

- в каждом отделе потоки между ПЭВМ и концентратором одинаковы. Поэтому в таблице для ветвей нижнего уровня выделено только три столбца – V_{1jk} (ветви 1-го отдела), V_{2jk} и V_{3jk} ;

- в каждом отделе потоки между концентратором и коммутатором для каждой рабочей группы одинаковы. Поэтому в таблице для ветвей среднего уровня тоже выделено только три столбца – V_{1j} (ветви 1-го отдела), V_{2j} и V_{3j} .

Например, пиковый поток от сервера S_1 до $PC_{118,}$ равный 1,2 Мбайт/с, пройдет по ветвям $V_{S1,}$ $V_{1,}$ V_{11} и V_{118} и должен быть отмечен в столбцах V_{sm} , V_1 , V_{1j} и V_{1jk} . Поток между двумя ПЭВМ 1-го и 2-го отделов, равный 50 Кбайт/с, пройдет по ветвям, например, $V_{111,}$ $V_{11,}$ $V_{1,}$ $V_{2,}V_{21}$ и V_{218} и должен быть отмечен в соответствующих столбцах.

Необходимо обязательно просчитывать количество одинаковых потоков, проходящих по той или иной ветви, и указывать их суммарный объём. Например, в 5-й строке столбца V_{1jk} указывается пиковый поток от одной из ПЭВМ 1-го отдела к семи другим ПЭВМ своей рабочей группы, умноженный на 2. Удвоение потока необходимо для учёта исходящего и входящего потоков, так как по одной и той же ветви V_{131} проходит, например, поток PC_{131} → PC_{138} и поток PC_{138} → PC_{131} .

После включения в табл. 4 всех потоков можно будет подсчитать суммарный информационный поток в каждой ветви.

Замечание. Такой подсчет суммарной нагрузки является очень упрощенным, так как пики потоков носят стохастический характер и необязательно совпадают во времени. Для точных расчетов применяются методы теории телетрафика, но для сложных сетей, подобных рассматриваемой, они не разработаны. В

случае необходимости получения точных значений объёмов потоков применяют методы имитационного моделирования, которые выходят за рамки данной контрольной работы. Однако, использование такой явно завышенной оценки суммарного потока в какой-то степени оправдано, так как создает определенный запас пропускной способности каналов. Обычная практика проектирования сетей - загружать каналы изначально не более, чем на 20-30%, поскольку интенсивность информационного обмена в современных сетях непрерывно возрастает (по некоторым оценкам удваивается каждые 9 месяцев).

4.5. Определяются требуемые канальные скорости для каждой ветви (заполняется последняя строка таблицы 4. Общепринято информационные потоки измерять в байт/с, а канальные скорости в бит/с. Поэтому переход от предпоследней строки таблицы 4 к последней состоит в простом умножении ее значений на 8 (на число бит в байте).

4.6. На заключительных этапах работы студенты должны выбрать оборудование для узлов ЛВС, рассчитать время реакции в тракте “ПЭВМ – Сервер – ПЭВМ”.

5. Выбор оборудования для узлов ЛВС.

Рассмотрим ряд устройств (концентратор, коммутатор, маршрутизатор), которые использовались при построении данной ЛВС.

5.1. Концентратор – это многопортовый повторитель, который любой бит, появляющийся на любом из его портов, передает на все другие порты, независимо от адреса принятого кадра (адрес даже не анализируется). Появление сигналов одновременно на двух или более входах рассматривается как столкновение и обнаруживается источниками этих сигналов. Передача временно прекращается и возобновляется через некоторый случайный промежуток времени. Концентратор – устройство физического уровня. Он проще и дешевле коммутатора, но безадресная передача кадров на все выходы сильно перегружает соответствующий сегмент сети.

5.2. Коммутатор - любой поступающий на его порт кадр записывает в память (целиком или только заголовок), анализирует адрес получателя и передает этот кадр только в направлении к адресату. Это даёт возможность коммутатору осуществлять одновременно несколько обменов. Например, передавать кадр с порта 1 на порт 7 и одновременно с 11-го порта - на 9-й. Коммутатор – устройство второго уровня. Он производит передачу кадров в соответствии с физическими адресами портов.

5.3. Маршрутизатор – устройство 3-го уровня. В сетях, входящих в Internet, маршрутизаторы анализируют адреса IP-пакетов, поступающих на любой из его портов, и в соответствии с этими адресами направляют пакеты к другим маршрутизаторам или ПЭВМ (напрямую или через сети 2-го уровня). Другая существенная функция маршрутизатора – согласование протоколов логического уровня. Как правило, порты маршрутизатора многофункциональны (или модульны). К одному маршрутизатору могут подключаться, например, каналы ЛВС Ethernet и каналы сети ATM, Frame Relay или ISDN.

5.4. Выбор оборудования производится по следующим критериям:

- число портов. Очевидно, что приобретаемое устройство должно иметь число портов, не меньшее, чем число подходящих к нему каналов;

- наличие соответствующих физических интерфейсов: коаксиальные кабели, витая пара, оптоволоконный кабель. Для нашей схемы примем, что все каналы организованы на витых парах категории 5, кроме канала выхода в Internet (V), который целесообразно организовать на оптическом кабеле;

- наличие соответствующих логических интерфейсов. Для внутренних линий нашей схемы примем интерфейсы разновидностей Ethernet (Ethernet, Fast Ethernet , Gigabit Ethernet или 10GBit Ethernet), а для внешней линии (V) – интерфейс с сетью ATM по каналу абонентского доступа ADSL;

- пропускные способности портов должны быть не ниже канальных скоростей, рассчитанных для соответствующих линий (последняя строка таблицы 4). При этом необходимо обратить внимание на следующее обстоятельство: представленные в таблицы 4 потоки получены путём суммирования двухсторонних потоков в каждой ветви. Поэтому выбранные канальные скорости будут достаточны даже для полудуплексных режимов работы портов Ethernet, а в случае выбора аппаратуры с дуплексным режимом будет обеспечен определённый запас по пропускной способности ветви. В лучшем случае, при симметричном трафике, запас будет двукратным. Например, при дуплексной связи порты Ethernet 10 Мбит/с могут передавать данные со скоростью 20 Мбит/с – по 10 Мбит/с в каждом направлении. Отметим также, что современные стандарты Ethernet, кроме коаксиальных версий, используют, как правило, дуплексные режимы;

- суммарная пропускная способность устройства должна быть не ниже суммы рассчитанных канальных скоростей для всех линий, подключаемых к этому устройству.

5.5. Рекомендации по выбору оборудования.

При выборе оборудования из номенклатуры: маршрутизатор, коммутатор, концентратор необходимо пользоваться следующими правилами:

- маршрутизатор по сравнению с коммутатором обладает большим интеллектом (работа с IP-адресами, борьба с широковещательными штормами), а коммутатор дешевле и обладает, как правило, большим быстродействием;

- коммутатор по сравнению с концентратором более интеллектуален (работа с MAC-адресами, что позволяет существенно ограничить зону коллизий, специфичную для технологии Ethernet). Концентратор дешевле. Однако, в последние годы в связи с массовым выпуском микрочипов для коммутаторов их стоимость значительно снизилась и они считаются более предпочтительными. В данном практическом занятии выбор предоставляется студентам.

6. Расчет времени реакции системы.

В системе клиент - сервер под временем реакции понимается интервал времени между вводом запроса в ПЭВМ клиента и получением ответа на экране монитора. С большими упрощениями этот расчет можно произвести следующим образом.

В общем виде $T_р = t_{пз} + t_s + t_{по}$;

- $T_р$ - время реакции;

- $t_{пз}$ - время передачи запроса от ПЭВМ до сервера. Так как запросы, как правило, очень короткие, то можно в $t_{пз}$ учесть только задержки в узлах. Примерно по 25мкс. на каждом узле (концентраторе или коммутаторе);

- t_s - время подготовки ответа сервером. Если не учитывать возможное стояние запроса в очереди на обслуживание сервером, то можно принять t_s равным 0,5мс;

- $t_{по}$ - время передачи ответа от сервера до ПЭВМ. Здесь, кроме задержки в узлах (25 мкс.), следует учесть и время прохождения длинного ответа через самый низкоскоростной канал.

Рассмотрим цепь каналов между сервером S_1 и PC_{111} (Рисунок – 2).

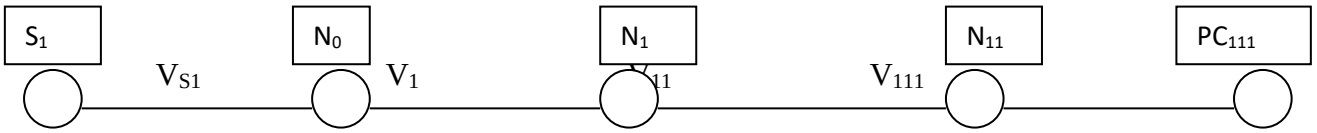


Рисунок 2 - Схема тракта сервер – рабочая станция (ПЭВМ).

Пусть в результате расчетов по пункту 5 определены следующие канальные скорости: C_{111}=10Мбит/с, C_{11}=10Мбит/с, C_1=100 Мбит/с:

Тогда, для передачи ответного файла, например, длиной в L Мбайт, по самому низкоскоростному каналу V_{111} (скорость C_{111}=10Мбит/с) потребуется время $t_{111} = \frac{L*8}{C} = \frac{0{,}01*10^6*8}{10*10^6} = 8мс$; (для L=0,01 Мбайт).

Цифра 8 в числителе соответствует числу бит в байте. Таким образом, общее время реакции составит:

$T_р$= 3*25мкс + 500мкс + 3*25мкс + 8000мкс = 8650 мкс.

Коэффициенты 3 в данной формуле соответствуют 3-м узлам, разделяющим ПЭВМ и сервер. Время - 500мкс – это продолжительность подготовки ответа сервером. Время - 8000мкс – это рассчитанная выше продолжительность передачи ответа (8мс). Длину ответного файла L студент выбирает произвольно и независимо от других студентов.

Строго говоря, расчёт времени реакции должен был учитывать и время распространения сигнала (электромагнитной волны) от компьютера до сервера и обратно. В общем случае, это время определяется как $t_p = S/v$, где: S – расстояние между двумя узлами (по кабелю);

v – скорость распространения электромагнитной волны в кабеле данного типа (можно принять v = 200000 км/с).

Тогда для S = 100 м получим t_p = 0,5 мкс. А время распространения сигнала в обе стороны определится как t_{p2} = 1мкс.

В связи с тем, что топология ЛВС (расположение серверов, компьютеров, коммутаторов) в данном практическом задании не определяется, а также в связи с

незначительностью времени распространения в пределах небольшой локальной сети, это время при расчёте времени реакции не учитывалось.

7. Заключение.

В заключении работы необходимо привести основные параметры рассчитанной системы:

- число ПЭВМ;
- время реакции;
- скорость канала доступа в Internet.

Контрольные вопросы.

1. Принцип работы сети по технологии Ethernet.
2. Основные стандарты технологии Ethernet.
3. Для каких передающих сред стандартизирована технология Ethernet.
4. Структура кадра Ethernet.
5. Какие отличия физических параметров сигналов существуют в технологиях Ethernet, Fast Ethernet и Gigabit Ethernet.

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №9

Моделирование запоминающего устройства.

Модель работы двухтактного буферного запоминающего устройства.

Идея двухтактного буфера: совместить во времени процессы сбора и записи информации в буферное запоминающее устройство (БЗУ) ограниченного объема и перезаписи информации в долговременное запоминающее устройство (ДЗУ), объем которой неограничен.

Техническое осуществление — с использованием канала прямого доступа и программного канала. Рассмотрим упрощенную модель системы двухтактного буферирования. Экспериментальная информация в процессе сбора записывается словами в буфер. При заполнении приемного БЗУ до определенного уровня, генерируется запрос к управляющему устройству (УУ) на передачу информации в ДЗУ (включается система прерываний). Между моментом возникновения запроса и переключением ключей (рисунок 1) проходит определенное время, по истечении которого приемное БЗУ становится передающим (например, подключается канал прямого доступа и информация пишется на диск), а для сбора информации в оперативной памяти выделяется новая область, которая становится приемным БЗУ.

При исследовании такой системы на модели могут ставиться раз-личные вопросы: каков должен быть уровень заполнения БЗУ, при котором в системе генерируется запрос на переключение ключей; как связан этот уровень с интенсивностью потока экспериментальных данных и вероятностью переполнения приемного БЗУ за время между запросом и переключением ключей и т.п.

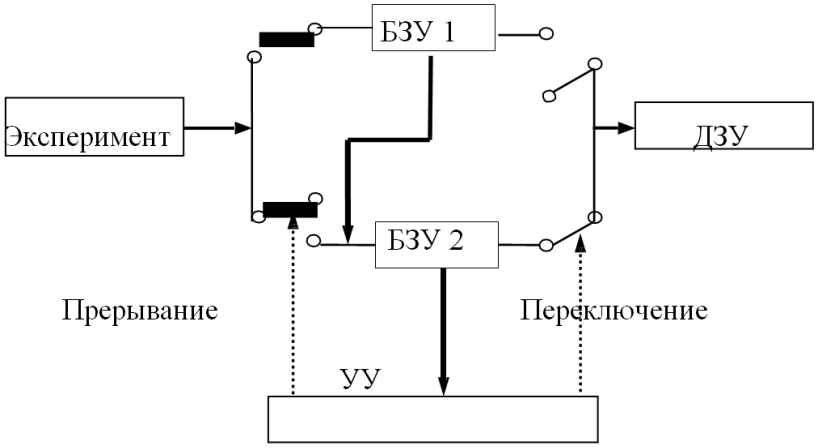


Рисунок 1 - Схема двухтактного БЗУ

Рассмотрим упрощенное описание модели на языке GPSS. Допустим, объем БЗУ — 1024 слова. Разрядность слова — 8 единиц. Тогда память БЗУ1/БЗУ2:

```
VAR2 VARIABLE 1024#8
STR1 STORAGE V$VAR2
```

Допустим, запрос на прерывание возникает при заполнении БЗУ до 1020 слов, тогда уровень заполнения:

```
VAR3 VARIABLE 1020#8
```

Еще одна переменная — описание среднего времени задержки между запросом и переключением БЗУ:

```
VAR4 VARIABLE 10/4
```

Пусть время между приходом слов в БЗУ распределено экспо-ненциально и равно в среднем 10 мин., тогда функционирование этой системы:

```
GENERATE (EXPONENTIAL(1,0,10))
GATE SNF STR1,OTKAZ ;если память заполнена — отказ
ENTER STR1,8
TEST E S$STR1,V$VAR3,ENDM
ADVANCE (EXPONENTIAL(1,0,V$VAR4)) ;переключение БЗУ
LEAVE STR1,S$STR1
TERMINATE 1
OTKAZ SAVEVALUE 1+,1 счетчик слов, потерянных из-за
переполнения БЗУ
ENDM TERMINATE
```

Здесь транзакт попадает в блок ADVANCE только тогда, когда БЗУ заполнена до необходимого уровня. Блок LEAVE полностью очищает память STR1, что равносильно переключению на новое БЗУ.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК